

Methoden zur intuitiven Modifikation und interaktiven Darstellung von großen Finite-Element-Modellen

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

Vorgelegt von

Dirc Rose

aus Heilbronn-Neckargartach

Hauptberichter: Prof. Dr. Thomas Ertl
Mitberichter: Prof. Dr. Günther Greiner
Tag der mündlichen Prüfung: 22. Februar 2006

Institut für Visualisierung und Interaktive Systeme
der Universität Stuttgart

2006

Für meine Eltern.

Inhaltsverzeichnis

Verzeichnislisten	vii
Abbildungsverzeichnis	vii
Farbabbildungsverzeichnis	xi
Abkürzungsverzeichnis	xiii
Danksagung	xv
Zusammenfassung und Kapitelüberblick	1
Abstract and Chapter Summaries	5
1 Einleitung	9
2 Berechnungsmethoden und Visualisierung strukturanalytischer Simulationen	15
2.1 Die Finite-Element-Methode	18
2.1.1 Grundelemente	18
2.1.2 Adaptive Netze	22
2.1.3 Verbindungsdefinitionen zwischen Finite-Element-Netzen	24
2.2 Finite Elemente in der Computergraphik	25
2.2.1 Die OpenGL-Graphik-Pipeline	26
2.2.2 Visualisierung von Finite-Element-Datensätzen	28
2.2.3 Pre- und Postprocessing	32
3 Optimierung der graphischen Darstellung	39
3.1 Optimierung der Qualität	39
3.1.1 Beleuchtung und Schattierung	40

3.1.2	Detektion und Hervorheben von Merkmalen	49
3.2	Optimierung der Performanz	55
3.2.1	Generelle Ansätze zur Entlastung der OpenGL-Pipeline	55
3.2.2	Adaptive Detaillierungsstufen	57
3.2.3	Optimierte Simplifizierungsverfahren	63
4	Intuitive und interaktive Modifikationsmechanismen	79
4.1	Linienförmige und flächige Verbindungselemente	80
4.1.1	Schweißnähte	81
4.1.2	Klebeverbindungen	88
4.2	Sensorpunkte	92
4.3	Modifizieren der Bauteilgeometrie	95
4.3.1	Selektieren im dreidimensionalen Raum	96
4.3.2	Editieren im dreidimensionalen Raum	99
4.3.3	Interaktive Netzvalidierung	106
4.3.4	Netzoptimierung	108
5	Virtuelle Realität am Arbeitsplatz	113
5.1	Stereosehen	113
5.1.1	Autostereoskopie	117
5.1.2	Generische Erweiterung von OpenGL-Programmen um Stereo- Funktionalität	121
5.2	Interaktion in der Virtuellen Realität	124
5.2.1	Menüs	125
5.2.2	Haptische Ein- und Ausgabe	129
6	Ergebnisse	137
6.1	Zusammenfassung	137
6.2	Bewertung	139
6.3	Ausblick	141
A	Farbabbildungen	143
B	Literaturverzeichnis	151
C	Lebenslauf	161

Verzeichnislisten

Abbildungsverzeichnis

2.1	Abstrahierter Konstruktions- und Berechnungszyklus eines neuen Fahrzeugs	16
2.2	Beispiele für statische Grundelemente mit unterschiedlicher Komplexität bzw. Anzahl an Freiheitsgraden	19
2.3	Ansatzfunktionen für die Biegefläche	20
2.4	Beispiel für ein Finite-Element-Netz	21
2.5	Strategien zur Netzverfeinerung	23
2.6	Gesamtvernetzung versus bauteilunabhängige Vernetzung	24
2.7	Darstellung dreidimensionaler Objekte in der Graphik-Pipeline	26
2.8	Detailliertere Ansicht der OpenGL-Graphik-Pipeline	27
2.9	Farbkodierung verschiedener Bauteile und Komponenten	29
2.10	Unterschiedliche Bauteilschattierungen	30
2.11	Für die Beleuchtungsberechnung benötigte Normalenvektoren	30
2.12	Halfway-Vektor	32
2.13	Repräsentation der FE-Konturen: 2-Pass vs. Textur	33
2.14	OpenGL Optimizer/Cosmo3D Szenegraph für eine Preprocessing-Anwendung .	34
2.15	Materialdurchdringung zweier Bleche	35
2.16	Schweißpunkte bilden die Verbindung zwischen benachbarten Bauteilen	35
2.17	Visualisierung der Eindringtiefe bei einem Seitenaufprall	37
3.1	Normalenverteilung und Beleuchtungsverlauf bei verschiedenen Ansätzen	40
3.2	Robuste Interpolation von Normalen	41
3.3	Dependent Lookup Textur für dritte Normalenkomponente / Normalisierung . . .	44
3.4	Beleuchtungsberechnung nach dem Phong'schen Modell für eine mit einer zwei-kanaligen Normalentextur überzogenen Oberfläche	46

3.5	Normalen können nur ungenau in den Textur-Farbkanälen abgespeichert werden .	48
3.6	Hervorhebung der Oberflächenmerkmale trägt wesentlich zum Verständnis bei . . .	50
3.7	Ermittlung der Außenkanten einer Oberfläche	51
3.8	Bestimmung der lokalen Orientierung der Elementnormalen	51
3.9	Zusammensetzen einzelner Kanten im Vergleich zu einem Dominospiel	53
3.10	Robuste Detektion flach auslaufender Knicke	55
3.11	Half Edge Collapse Operation	58
3.12	Butterfly Unterteilungsschema	59
3.13	Gleichmäßige, dyadische Unterteilung eines Dreiecks	60
3.14	Maximale lokale Distanz zwischen Netzen zweier Unterteilungstiefen	60
3.15	Saubere Übergänge zwischen unterschiedlichen Unterteilungsstufen	61
3.16	Ungleichmäßige „green“-Unterteilung eines Dreiecks	61
3.17	Simplifizierungsprozess	66
3.18	Aufspaltung eines komplexen Polygons	68
3.19	Geeignete Auflösung der Normalentextur	69
3.20	Initialisierung einer Normalentextur	71
3.21	Normalentextur-Atlas des in Abb. 3.6(b) bzw. 3.17 gezeigten Bauteils	71
3.22	Eine Gerade kann indirekt in einer gerichteten Distanztextur gespeichert werden .	72
3.23	Vollständige Berandung eines finiten Elements	73
3.24	Zweikanalige Distanzfeldtextur ermöglicht die Restauration hochaufgelöster Be- randungslinien von einem Viertel der Elemente	74
3.25	Schachbrettartige Elementanordnung	74
3.26	Distanzfeldtextur-Atlanten des in Abb. 3.6(b) bzw. 3.17 gezeigten Bauteils	75
3.27	Simplifiziertes Gesamtfahrzeug mit restaurierten Elementberandungen und Phong-Beleuchtung/-Schattierung	76
4.1	Erzeugen einer gültigen Schweißpunktlinie entlang eines Flansches	81
4.2	Unterscheidung zwischen inneren und Kantenschweißnähten	82
4.3	Bestimmung der Mittellinie von Schweißnähten entlang von Außenkanten	83
4.4	Repräsentation einer inneren und einer Kantenschweißnaht	84
4.5	Distanzanalyse von Schweißnähten im dreidimensionalen Raum	86
4.6	Beispiel für eine VDAFS-Eingabedatei	87
4.7	Aus VDAFS-Daten generierter „Edge Link“	88

4.8	Anfang bzw. Ende eines Klebestreifens werden durch eine Ebene begrenzt	89
4.9	Visualisierung einer fehlerbehafteten Klebeschicht	90
4.10	Klebeschicht wird von einem Bauteil durchdrungen	91
4.11	Vergleich zwischen realen und modellierten Sensoren	93
4.12	Visualisierung von Sensorpunkten	94
4.13	Bauteilpenetration und -perforation	95
4.14	Selektion mittels einer Freihandlinie	97
4.15	Sortierung der Berandungskanten einer Freihandselektion	98
4.16	Merkmalsbasierte Selektion	99
4.17	Dreidimensionales Translations-Widget	100
4.18	Gemeinsame Parallelverschiebung einer Menge von Knoten	101
4.19	Kegelförmige Deformation durch direkte Anwendung der Wichtungsfaktoren	102
4.20	Gleichmäßige Ausbeulung durch Anwenden einer zweidimensionalen Transferfunktion	103
4.21	Rotationsdeformationen	104
4.22	Unterschiedliche Deformationsoperationen an einem freien Bauteilende	105
4.23	Markierung fehlerhafter Elemente durch Glyphen	107
4.24	Glyphenbasierte Fehleranzeige an einer Fahrzeugkomponente	108
4.25	Relaxation eines FE-Netzes	109
4.26	Randbedingungen beim Verschmelzen kurzer Kanten zu einem gemeinsamen Knoten	110
4.27	Lokale Neuvernetzung der fehlerhaften Elementgebiete aus Abb. 4.23	111
4.28	Verlängerung des Bauteils aus Abb. 4.22(a) mit interaktiver Neuvernetzung	112
5.1	Kameraeinstellungen für die Stereoprojektion	114
5.2	Parameter eines schiefwinkligen Frustums	115
5.3	Funktionsprinzip eines autostereoskopischen Displays	118
5.4	Zeichnen einer dünnen Linie zur Wiedergabe auf einem autoster. Bildschirm	119
5.5	Aufbereitetes Gesamtfahrzeugbild zur Ausgabe auf einem autoster. Monitor	120
5.6	Einschleusen zusätzlicher Bibliotheksroutinen	121
5.7	Rot-Cyan-Anaglyph eines Fahrzeugmodells	123
5.8	Menü mit drei Ebenen auf einer kleinen Ausgabefläche	126
5.9	Angepasste Benutzeroberfläche in der Anwendung auf einem iPAQ	127

5.10	Evolutionsbaum der alternativen Menü-Interaktion	128
5.11	SpaceMouse® der Firma 3Dconnexion	129
5.12	Immersive Arbeitsplätze	131
5.13	Kollisionsflächen während der Modifikation	132
5.14	Durchdringungsproblematik bei Verformungen	133
5.15	Gegenkräfte ermöglichen eine Rückkopplung über die Netzqualität	134
5.16	Deformationsoperationen mit virtuellem Stylus	135

Farbabbildungsverzeichnis

2.9	Farbkodierung verschiedener Bauteile und Komponenten	143
2.15	Materialdurchdringung zweier Bleche	144
2.16	Schweißpunkte bilden die Verbindung zwischen benachbarten Bauteilen	144
3.3	Dependent Lookup Textur für dritte Normalenkomponente / Normalisierung	145
3.4	Beleuchtungsberechnung nach dem Phong'schen Modell für eine mit einer zwei- kanaligen Normalentextur überzogenen Oberfläche	145
3.17(c)	Gebietsgrenzen im Simplifizierungsprozess	146
3.27	Simplifiziertes Gesamtfahrzeug mit restaurierten Elementberandungen und Phong-Beleuchtung/-Schattierung	146
4.9	Visualisierung einer fehlerbehafteten Klebeschicht	147
4.12	Visualisierung von Sensorpunkten	147
5.1	Kameraeinstellungen für die Stereoprojektion	148
5.7	Rot-Cyan-Anaglyph eines Fahrzeugmodells	148
5.10	Evolutionsbaum der alternativen Menü-Interaktion	149

Abkürzungsverzeichnis

2D	zweidimensional	GUI	Graphical User Interface
3D	dreidimensional	GLUT	OpenGL Utility Toolkit
Abb.	Abbildung	HILO	High-Low (Texturformat mit 16-bit Genauigkeit pro Kanal)
AGP	Accelerated Graphics Port (Graphikbus)	i.A.	im Allgemeinen
API	Application Programming Interface (Anwendungsprogrammierschnittstelle)	ID	Identifikationsnummer
BMBF	Bundesministerium für Bildung und Forschung	Kap.	Kapitel
BMW	Bayerische Motoren Werke AG	kHz	Kilohertz
bzw.	beziehungsweise	MB	Megabyte
CAD	Computer-Aided Design	MFC	Microsoft Foundation Classes
CAE	Computer-Aided Engineering	MHz	Megahertz
CAM	Computer-Aided Manufacturing	NC	Numerical Control
CAVE	CAVE Automatic Virtual Environment	o.Ä.	oder Ähnliche(s)
CFD	Computational Fluid Dynamics (Numerische Strömungssimulation)	OpenGL	Open Graphics Language
d.h.	das heißt	PC	Personal Computer
DFP	Digital Flat Panel (digitaler Flachbildschirm)	PCI	Peripheral Component Interconnect (Bussystem)
DLL	Dynamic Link Library (dynamisch geladene Bibliothek)	Pixel	Picture(Bild)-Element
evtl.	eventuell	RGB	rot, grün und blau
FD	Finite Differenzen	RGBA	rot, grün, blau und alpha
FE	Finite(s) Element(e)	SIMD	Single Instruction, Multiple Data
FEM	Finite-Element-Methode	Texel	Textur-Element
FV	Finite(s) Volumen	TFT	Thin Film Transistor
GB	Gigabyte	u.a.	unter anderem
ggf.	gegebenenfalls	u.U.	unter Umständen
GHz	Gigahertz	VDAFS	Verband der Automobilindustrie – Flächenschnittstelle
		VR	Virtuelle Realität
		vs.	versus
		z.B.	zum Beispiel

The last ever dolphin message was misinterpreted as a surprisingly sophisticated attempt to do a double-backwards-somersault through a hoop whilst whistling the “Star Sprangled Banner”, but in fact the message was this: So long and thanks for all the fish.

Douglas Adams: The Hitch Hiker’s Guide to the Galaxy

Danksagung

Diese Arbeit wäre niemals ohne die Unterstützung und tatkräftige Mithilfe zahlreicher Personen zustande gekommen. An allererster Stelle ist hierbei mein Doktorvater Thomas Ertl zu nennen. Ihm gilt mein besonderer Dank für ein stets offenes Ohr, seine kompetenten Ratschläge sowie Führung und die Hilfe insbesondere beim Entstehen diverser Publikationen. Meinem Zweitgutachter Günther Greiner danke ich für die aufmunternden Worte im Vorfeld und die mühevoll Begutachtung der nachfolgenden Seiten. Emeritus Rul Gunzenhäuser war ebenfalls nicht unbetieilt, indem er durch diverse Ausflüge dem angenehmen Arbeitsklima stets noch das Tüpfelchen auf das „i“ setzte.

Ich hatte das Glück, stets mit sympathischen Kollegen zusammenarbeiten zu dürfen. Hier sind zum einen meine Mitstreiter in den beiden BMBF-Projekten „AutoBench“ und „AutoOPT“ zu nennen: Ich danke Ove Sommer, Norbert Frisch und Katrin Bidmon dafür, dass sie mich nicht alleine vor dem Code von „FEMod“ sitzen ließen. Den eben Genannten sowie Simon Stegmaier, Guido Reina und Daniel Weiskopf gilt außerdem mein Dank für die oft tief in die Nacht bzw. in den Morgen dauernde Zusammenarbeit bei verschiedenen Publikationen. Simon Stegmaier möchte ich an dieser Stelle zusätzlich für die Erkenntnis danken, dass Singen nicht alles im Leben ist.

Viele Mitarbeiter waren für mich nicht nur Kollegen oder Zimmernachbarn, sondern sind auch über die Zeit am Institut hinaus noch gute Freunde. Ich möchte mich bei Katrin Bidmon, Matthias Hopf, Guido Reina, Ulrike Ritzmann und ganz besonders Marcelo Magallón bedanken, dass sie nicht nur nette und hilfsbereite Kollegen waren, sondern auch über die Arbeit hinaus mir zur Seite standen. Meinen „Auslandsbegleitern“ Simon Stegmaier und Manfred Weiler danke ich für unvergleichliche Erlebnisse in den USA und Marcelo Magallón für eine unvergessliche Führung durch seine zweite Heimat. Außerdem möchte ich mich bei Jana Tenner und Karin Ganslmayer für das Verständnis bedanken, dass sie mich vor Abgabeterminen meist nur sehr selten zu Gesicht bekamen. Alle Kollegen und Freunde, welche ich hier nicht explizit erwähnt habe und welche ihren Teil zum Gelingen dieser Arbeit oder erinnerungswürdigen Momenten beigetragen haben, schließe ich hiermit in meinen Dank ein. Hierzu zähle ich auch diejenigen, für die der Montag nicht einfach nur ein Wochentag ist.

Im Laufe der Arbeit entstanden mehrere Studien- bzw. Diplomarbeiten, deren Ergebnisse in diese Dissertation eingeflossen sind. Für die Umsetzung der darin verwirklichten Ideen möchte ich mich bei Martin Kada, Reinhold Zöllner und Daniel Mohr bedanken. Ebenfalls zur erfolgreichen Realisierung der im Folgenden präsentierten Forschungsergebnisse haben die Kooperationspart-

ner am Forschungs- und Innovationszentrum der BMW Group beigetragen. Für entsprechende Anregungen aus der Praxis bedanke ich mich daher bei Sven Kuschfeldt, Horst-Uwe Mader und Christoph Lübbling.

Mein Dank geht auch an diejenigen, welche die Mühe auf sich genommen haben, die Tippfehler in diesem Werk zu finden und die mir wertvolle Ratschläge zur Verbesserung erteilt haben. Dies sind neben Thomas Ertl im Einzelnen: Ingrid und Manfred Rose, Katrin Bidmon, Guido Reina, Daniel Weiskopf sowie Fredrik Svensson.

Last but not least möchte ich mich bei Maria Haase bedanken. Ohne ihre Unterstützung und Ermutigung im Vorfeld hätte die Suche nach einer geeigneten Promotionsstelle mit Sicherheit nicht einen solch positiven Ausgang genommen. Sie hat – wie auch meine Eltern – immer an mich geglaubt, wobei ich Letzteren ebenfalls für die Unterstützung und auch ihre Geduld danke.

Those who wish to know should read on. Others may wish to skip on to the last chapter which is a good bit and has Marvin in it.

Douglas Adams: So long, and thanks for all the fish

Zusammenfassung und Kapitelüberblick

Die Entwicklung neuer Fahrzeugtypen im Automobilbau unterliegt zunehmend einem steigenden wirtschaftlichen und damit zeitlichem Druck. Um diesem Druck entgegenzuwirken, wurde unter anderem mit der Entwicklung und Einführung virtueller Simulationstechniken vor einigen Jahren ein wesentlicher Schritt hin zu kürzeren Evolutionszyklen getätigt. Um die dabei eingesetzten Daten – in der Regel Finite-Element (FE)-Modelle – visualisieren und bearbeiten zu können, wurden von verschiedenen Anbietern teilweise spezialisierte Werkzeuge erstellt. Allerdings wurde beim Entwurf der meisten dieser Werkzeuge lediglich auf die reine Funktionalität geachtet, so dass eine intuitive und somit schnell zu erlernende, einfache Bedienbarkeit dieser Anwendungen in der Regel nicht gegeben ist. Zudem werden im Allgemeinen die Möglichkeiten moderner Graphikhardware überhaupt nicht oder nur in geringem Maße ausgenutzt, so dass aufgrund der konventionellen Darstellungsmethoden ein interaktives Arbeiten mit großen Modellen nicht durchführbar ist bzw. mit einer Performanz erfolgt, die hinter aktuellen Möglichkeiten zurückbleibt.

Die vorliegende Dissertation befasst sich daher sowohl mit Methoden zur Optimierung der Qualität als auch der Steigerung der Interaktionsrate bei der Modelldarstellung. Dazu wurden Verfahren entwickelt, welche durch speicherfreundliches Ablegen von Oberflächeninformationen in Texturen auf der Graphikkarte den Einsatz beliebiger Beleuchtungsmodelle und eine pixelechte Auswertung derselben erlauben. Da bei technischen Komponenten Knicke in der Struktur sich entscheidend auf die Stabilität eines Fahrzeugs auswirken können, werden solche Merkmale gesondert behandelt und hervorgehoben. Bei den im weiteren Verlauf präsentierten Simplifizierungsverfahren zur Performanzsteigerung bei der interaktiven Visualisierung großer FE-Modelle werden spezialisierte Methoden vorgestellt, welche hohe Simplifizierungsraten mit dem Erhalt dieser Merkmale verbinden. Dabei tragen neue Graphiktechnologien und darauf zugeschnittene Algorithmen dazu bei, auch feine Details konservieren zu können – wie z.B. die Anzeige der Umrandung der einzelnen finiten Elemente. Somit ist zwischen Original und simplifiziertem Modell kein visueller Unterschied erkennbar.

Diese interaktiven Visualisierungstechniken bilden die Grundlage für intuitive Modifikationsmechanismen, welche es erlauben, benutzerfreundlich und innerhalb kürzester Zeit Standardaufgaben durchzuführen. Dazu zählen die schnelle und bequeme Definition von Bauteilverbindungen und das Erzeugen virtueller Sensorpunkte, welche zur Überprüfung auftretender Kräfte in der

experimentellen Simulation unerlässlich sind. Neben diesen vor allem für strukturmechanische Untersuchungen wichtigen Themen wurden Methoden zur interaktiven Verformung bestehender Modellgeometrie entwickelt. Dabei wurde besonderen Wert auf die Flexibilität der Deformationsoperationen unter Anwendung einer intuitiv zu bedienenden Steuerung gelegt. Während der Interaktion ständig durchgeführte Kontrollen der Netzqualität und entsprechende visuelle Rückkopplungsmechanismen sowie auf die Bedürfnisse von FE-Netzen zugeschnittene Optimierungsalgorithmen zur automatischen Behebung der dabei ggf. entstehenden FE-Vernetzungsfehler unterstützen den Anwender bei der Durchführung dieser Aufgaben. Um ein optimales Verständnis zu garantieren, wurden zudem spezielle dreidimensionale Glyphen erstellt, welche die Bearbeitung der erläuterten Aufgaben sinnvoll ergänzen und erleichtern.

Räumliche Darstellungs- und Interaktionstechniken können dabei das Verständnis und den Umgang mit einem dreidimensionalen Modell erheblich verbessern. Deshalb wurde der im Laufe der Arbeiten entstandene, mittlerweile kommerziell vertriebene Prototyp um verschiedene Ein- und Ausgabemöglichkeiten erweitert, welche die Einbindung in eine arbeitsplatzbasierte, virtuelle Umgebung ermöglichen. Neuartige Hardware erlaubt zu diesem Zweck sowohl autostereoskopische Visualisierungen als auch haptisch unterstützte Eingabe. Im Hinblick auf die im vorangegangenen Abschnitt vorgestellten Methoden ergeben sich in solch einer immersiven Arbeitsplatzumgebung neue Anforderungen und neue Möglichkeiten, welche entsprechend untersucht und eingebunden wurden. Im Vergleich zu konventionellen bzw. verbreiteten FE-Werkzeugen erlauben die im Rahmen dieser Arbeit entwickelten Verfahren somit ein intuitiveres, angenehmeres, direkteres und damit schnelleres bzw. effektiveres Arbeiten im Umgang mit dreidimensionalen Modellen.

Kapitelüberblick

Diese Abhandlung ist in insgesamt sechs Kapitel gegliedert. *Kapitel 1* gibt einen kurzen Einblick in bestehende Arbeitsabläufe sowie Anwendungsszenarien und erläutert die dabei auftretenden Problemstellungen, welche die Motivationsgrundlage für die durchgeführten Forschungen bilden.

Kapitel 2 schließt mit der Erklärung im Automobilbau gebräuchlicher Simulationsmethoden an. Hierbei wird insbesondere detailliert auf die Methode der finiten Elemente eingegangen. Neben einem Überblick über dieses Berechnungsverfahren und aktuell eingesetzter Simulationsmethoden und -optimierungen ist das Kapitel hauptsächlich auf bestehende Algorithmen und Methoden zur Visualisierung solcher Datensätze ausgerichtet. Am Beispiel der Graphiksschnittstelle OpenGL werden zunächst einige Grundlagen erklärt und im Verlauf auf höherer Ebene angesiedelte Szenengraph APIs übertragen. Abschließend wird auf generelle Unterschiede in der Darstellung und Handhabung von Pre- und Postprocessingdaten eingegangen.

Die nachfolgenden Kapitel erläutern die erzielten Ergebnisse, welche entsprechend ihres Forschungsgebietes in drei Hauptthemen aufgegliedert wurden. *Kapitel 3* befasst sich mit Optimierungen in der graphischen Darstellung der Fahrzeugmodelle. Diese umfassen sowohl Verbesserungen in der Qualität als auch in der Geschwindigkeit. Zu ersteren zählen unter ande-

rem eine hochwertige Beleuchtungsberechnung und die damit verbundene Schattierung von FE-Oberflächen oder geeignete Mechanismen zur Hervorhebung von besonderen Merkmalen dieser gitterbasierten Oberflächen durch Analyse von Krümmungen und dem Verlauf von Unstetigkeiten. Trotz des Hauptziels der Qualitätsverbesserung wird hierbei, wie auch im Folgenden, besonders auf die Performanz der eingesetzten Algorithmen geachtet. Im zweiten Teil dieses Kapitels wird zudem gesondert auf schnelle Darstellungsmethoden unter Ausnutzung moderner Graphikhardware eingegangen. Das Hauptaugenmerk liegt dabei auf Simplifizierungsmethoden, welche die bei FE-Modellen häufig auftretende Geometrielast auf die Rasterisierungsleistung der Graphikkarte umverteilen. Mittels spezieller Texturen wird – trotz einer deutlich reduzierten Geometrie – eine augenscheinlich exakte Visualisierung des FE-Modells wiedergegeben.

Kapitel 4 befasst sich mit der Benutzerinteraktion für Anwendungen im FE-Umfeld. In Anlehnung an bereits bekannte und in anderen Arbeiten veröffentlichten Ansätzen beschreibt der erste Abschnitt des Kapitels Erweiterungen auf dem Gebiet der Bauteilverbindungen zwischen unabhängig vernetzten Fahrzeugkomponenten. Ebenfalls in diesen Bereich fällt die Definition virtueller Sensorpunkte, welche zum einen die Auswertung bestimmter Parameter erlauben, zum anderen aber auch zum Abgleich zwischen der Simulation und realer Versuche dienen. Der letzte Kapitelabschnitt behandelt die manuelle und interaktive Deformation von FE-Gittern zur einfachen und schnellen Erzeugung von Bauteilvarianten oder zur Behebung von fehlerhaften Bereichen. Parallel dazu werden Methoden vorgestellt, welche dem Benutzer bereits während der Modifikation eine instantane Rückkopplung über die aktuelle Netzqualität liefern und diese ggf. automatisch optimieren.

In *Kapitel 5* wird schließlich auf die Anwendbarkeit der in den vorangegangenen Kapiteln präsentierten Verfahren in einer VR-Umgebung eingegangen. Hierbei werden im speziellen neuere Interaktions- und Wiedergabegeräte und hier insbesondere Hardware-Lösungen für Arbeitsplatzumgebungen, adressiert. Dies sind zum einen autostereoskopische Monitore, welche dreidimensionales Stereosehen ohne Zusatzgeräte wie z.B. einer Shutter-Brille erlauben, und zum anderen haptische Eingabegeräte, welche – zusätzlich zu den üblichen Mechanismen Bild und Ton – mittels Kraftübertragung einen weiteren Rückkanal zum Benutzer bieten. Dabei werden sowohl die Interaktionsmöglichkeiten unter Ausnutzung der zusätzlichen dritten Dimension als auch die sich u.U. daraus ergebenden Problematiken diskutiert.

Abschließend folgt in *Kapitel 6* eine Zusammenfassung und Bewertung der erzielten Ergebnisse sowie ein kurzer Ausblick, wie und wo die präsentierten Methoden in naher Zukunft zum Einsatz kommen können. Zukünftige Entwicklungen können so durch die Anwendung der erarbeiteten Methoden noch weiter verbessert werden.

In den Anhängen findet sich das Literaturverzeichnis und der Farbteil, in welchem von ausgesuchten Bildern aus dem Hauptteil der Dissertation jeweils hoch aufgelöste, farbige Versionen der entsprechenden Abbildungen abgedruckt sind. Abbildungen, für die eine zugehörige Farbabbildung existiert, sind am Rand mit einem zusätzlichen Symbol wie nebenstehend gekennzeichnet.



“What’s this fish doing in my ear?”

“It’s translating for you. It’s a Babel fish.”

Douglas Adams: The Hitch Hiker’s Guide to the Galaxy

Abstract and Chapter Summaries

The development of new automotive prototypes is increasingly subject to economic and temporal pressure. To counteract this pressure, simulation methods have been developed and introduced into the early design process for several years now, leading to considerably shorter evolution cycles. Various software providers developed specialized tools to visualize and handle the data necessary for such simulations. For structural analysis, such simulation data usually consists of finite element models. However, most of the commercially available software packages only provide bare functionality, and no intuitive and easy-to-learn handling of the variety of finite element components. In general, the capabilities of modern graphics hardware is hardly exploited. Therefore, interactive work on large models is limited or not possible at all. This is due to the application of conventional rendering methods, with a performance that falls short of today’s possibilities.

This PhD thesis addresses methods for both the optimization of the visualization quality and the improvement of the interaction rate for the rendering of large finite element models. For this purpose, various algorithms have been developed, which enable the adoption of arbitrary illumination models combined with pixel-exact evaluation of their lighting terms and a memory-efficient storage of the required surface information in textures. Because of the severe impact of sharp bends and corrugations on the stability of structural components, such features are separately examined and accentuated in the graphical representation. A simplification approach tailored for finite element models and for the preservation of detailed features is presented. These methods offer high simplification ratios for large FE models and make interactive visualization with high quality possible. Techniques employing modern graphics technologies contribute to this rise in quality and special algorithms are able to reproduce fine details, e.g. the rendering of the wire frame lines of finite elements, out of a small footprint of allocated texture memory. No visual distinction between the original and the simplified, highly interactive version of a model is possible, and the latter often appears superior due to the improved lighting calculation.

These interactive visualization techniques establish the foundation for intuitive modification mechanisms which permit to accomplish standard tasks in a user-friendly way and in short time. Subtasks include the comfortable definition of component assembly and the creation of virtual sensor points, which are essential for the verification of forces and accelerations occurring in experimental simulations. Besides these processes – which are vital for modern structural analysis – several methods for interactive deformation of the geometry of existing models have been developed and evaluated. Attention has been paid to the flexibility of the deformation operations

while the user takes advantage of an intuitive control system. When performing an editing task the user is supported by concurrent monitoring of quality attributes of the finite element mesh and by appropriate feedback. Additional optimization algorithms assist the user in repairing or removing erroneous elements manually, semi-automatically, or completely autonomously. Special three-dimensional glyphs and widgets ensure optimal comprehension of the modifications and facilitate the appropriate handling of the tasks described above.

The understanding of the topology and the handling of three-dimensional models can definitely be enhanced by introducing stereographic display and spatial interaction techniques. Therefore, the prototype developed in the course of this PhD thesis – which is marketed commercially in the meantime – has been extended by various input and output facilities. These permit the finite element tool to be integrated into a virtual environment located at the work place. For this purpose, novel hardware contributes autostereoscopic visualization as well as haptically supported input. Particularly with regard to the methods mentioned in the preceding paragraph, new possibilities and also challenges arise within an immersive work place environment. These new immersive techniques have been evaluated and integrated into the prototypical tool. In contrast to conventional and widespread FE tools the techniques developed within this research project offer a more intuitive, more pleasant, and more direct way of working, and therefore permit a faster and more effective way of handling three-dimensional models.

Chapter Summaries

This PhD thesis consists of six chapters. *Chapter 1* gives a short introduction into current work flows and application scenarios in automotive engineering. It describes the problems arising while performing typical tasks. These problems provide the motivation for the accomplished research efforts.

Chapter 2 explains common simulation techniques in the automotive industry, focusing in detail on the finite element method. Besides an overview of this analysis method and of currently applied simulation techniques as well as optimizations, this chapter gives a review of existing algorithms and methods for visualizing such datasets. Some basic principles are explained by means of the low-level graphics API OpenGL and also carried over to high-level scene graph APIs. Finally, general distinctions of visualization and handling of pre- and postprocessing data are discussed.

The main contribution of this work is presented in the following three chapters. The optimization of the graphical presentation of car models is addressed in *Chapter 3*. It comprises an improvement in both rendering quality and performance. A high-quality illumination calculation belongs to the enhancements concerning the visual quality just as the associated shading of finite element surfaces. Appropriate mechanisms for highlighting particular features of the grid-based surfaces by analyzing their curvature and determining the progression of discontinuities belong to this specific field as well. Despite the main goal of quality improvement in this context the introduced algorithms aim at optimal performance. The second part of this chapter describes details of fast rendering methods exploiting the capabilities of modern graphics hardware. The main

focus is set on simplification algorithms which redistribute the geometry load – often problematic with complex finite element models – onto the rasterization capacity of the graphics card. Special textures and appropriate shaders can be utilized to reproduce a visually indistinguishable representation of the original finite element model, despite a significant geometry reduction.

Chapter 4 deals with user interaction in finite element applications. The first section of this chapter describes enhancements for the assembly of independently meshed FE components which is a consequential continuation of previous approaches already published and established within the workgroup. The definition of virtual sensor points belongs to this domain as well. Sensor points open up the possibility of examining certain parameters and permit the comparison of simulation and experimental results. The last section of this chapter covers manual and interactive deformation of finite element meshes. The main purpose of such manipulation methods is the creation of model variants as well as the correction of erroneous element regions in an easy and fast manner. In parallel, methods are presented that provide the user with instantaneous feedback about the current mesh quality concurrently to the manual modification process. Auxiliary algorithms facilitate an automatic optimization of the distorted mesh and recover a finite element model valid for simulation.

The last chapter – *Chapter 5* – details the adaptability of the methods presented in the preceding chapters for virtual reality environments. In particular, novel interaction and display devices are addressed. Specific focus is set on hardware solutions for work place environments. For one these are autostereoscopic monitors that enable three-dimensional stereo vision without any additional devices like shutter glasses. Other immersive equipment consists of haptically enhanced input devices, which provide a further feedback channel by transmitting forces besides the usual mechanisms such as vision and sound. The potential of interaction techniques exploiting the extra dimensions is discussed. Problems arising from the support of such devices are addressed and appropriate solutions are presented.

A summary including an evaluation of the achieved results is given in *Chapter 6*. A short outlook is given on future prospects. In the concluding remarks it is discussed where and how the presented methods can be employed in the near future and how they might be further improved within the scope of ongoing development.

The bibliography can be found in the appendixes as well as the color plates, which reproduce colored high resolution versions of selected pictures from the main part. Monochrome figures with a corresponding colored counterpart in Appendix A are denoted by an additional symbol at the outer page margin as shown next to this paragraph.



Kapitel 1

Einleitung

“So we’re actually going to land in a minute?”

“Well not so much land, in fact, not actually land as such, no ... er ...”

“What are you talking about?” said Ford sharply.

“Well,” said the Captain, picking his way through the words carefully, “I think as far as I can remember we were programmed to crash on it.”

“Crash?” shouted Ford and Arthur.

“Er, yes,” said the Captain, “yes, it’s all part of the plan I think. There was a terribly good reason for it which I can’t quite remember at the moment. It was something to with ... er ...”

Douglas Adams: The Restaurant at the End of the Universe

Neue Designs und Modelle werden zunehmend am Rechner entworfen. Die Begriffe Design und Modell sind hierbei bewusst zunächst nicht näher spezifiziert, denn diese Aussage trifft im Wesentlichen auf fast alle Industriezweige zu. Sowohl im Maschinen- und Anlagenbau, in der Metallindustrie, im Fahrzeugbau, aber auch in der Textil- und Bekleidungsindustrie hat der Computer das Reißbrett verdrängt. Wegen dem enormen Konkurrenz- und damit bedingten Preis- und Zeitdruck kam dieser Trend besonders früh im Fahrzeugbau und hier vornehmlich im Automobilbau zum Tragen. Durch konsequenten Einsatz von Computern konnten die Entwicklungs- und auch Fertigungsdauern im Vergleich zu früher stark reduziert werden. So wird heutzutage ein neues Fahrzeugmodell komplett mit Hilfe von CAD-Techniken entworfen und konstruiert. Daran angeknüpft lassen sich, durch entsprechende Eingliederung in einen alle wichtigen Bereiche – wie Entwicklung, Simulation, Produktion, Zulieferung – umfassenden CAE-Prozess, sowohl im Prototypen- als auch Produktionsstatus, z.B. automatisiert CAM-Verfahren ansteuern. So ist es in der Regel ohne weiteres möglich, aus CAD-Daten die entsprechende CAM- bzw. NC-Umgebung einer Fräsmaschine mit den gewünschten Daten zu versorgen. Dies ermöglicht es, in viel kürzerer Zeit als bisher, einen Prototypen zu erstellen, da die im Großen und Ganzen redundanten Daten nicht mehrmals – in der Regel von Hand – erzeugt werden müssen.

Diese Abläufe und Techniken erreichen mittlerweile einen Zustand, den man als ausgereift bezeichnen kann, und haben daher auch eine sehr große Verbreitung erlangt. Natürlich gibt es auch hier weiterhin Entwicklungsarbeit, so zum Beispiel bei der in jüngster Zeit zunehmend eingesetzten assoziativ parametrischen Konstruktion, bei der nicht mehr vergleichbar zum Reißbrett konstruiert wird, sondern ein Bauteil komplett mittels seiner definierenden Parameter beschrieben wird. Diese Parametrisierung verlangt zwar gegebenenfalls etwas mehr Disziplin und Aufwand bei der Modellierung, hat aber den Vorteil, dass sehr schnell Varianten erzeugt werden können. Letztendlich sind aber auch diese neueren Verfahren im Grunde vergleichbar mit der herkömmlichen Vorgehensweise und verlangen daher keine gravierenden Änderungen im Entwicklungsprozess.

Aus der Sicht des CAE-Umfelds hat sich daher in den vergangenen Jahren ein mehr oder weniger gleich bleibender Ablaufprozess eingestellt, der zwar immer mehr verfeinert und optimiert wurde, aber trotzdem nicht ohne weiteres Platz für große Einsparungen ließ. Dies hat sich erst

mit der verbreiteten Einführung von numerischen Simulationen – anstelle des zeit- und extrem kostenintensiven Prototypenbaus und realer Versuche – geändert. Zwar ist die Grundidee für solche numerischen Simulationen, wie z.B. die Methode der finiten Elemente, schon relativ alt – je nach Definition zwischen 60 und 300 Jahre [Ode87] – deren praktische Umsetzung ist jedoch bis vor einigen Jahren noch nicht für sehr große und komplexe Modelle mit vertretbarem Aufwand möglich gewesen. Vor allem im Bereich der nicht-linearen Dynamik, wie der Simulation von Crashes, sind langwierige Berechnungen über hunderttausende von Zeitschritten durchzuführen. Erst seit den Neunziger Jahren ist es daher möglich, solche Berechnungen im großen Stil für ein hinreichend detailgetreues Simulationsmodell innerhalb weniger Tage zu bewältigen.

Zur Erstellung, Bearbeitung und Auswertung dieser numerischen Simulationsmodelle sind zur Zeit lediglich wenige allgemein anwendbare Werkzeuge vorhanden. Die bereits existierenden Programme sind meist auf spezielle Anwendungsfelder beschränkt und großteils kryptisch und umständlich zu bedienen. Teilweise wird versucht, sich an die bestehenden Prozesse anzulehnen, indem man sich Strukturen aus Bereichen des CAD bedient. Dies hat den Vorteil, dass der Benutzer sich nicht komplett neu orientieren muss, gelingt aber bisher leider nur für sehr einfache Aufgaben. Ebenso wird versucht, sich der bereits bekannten Algorithmen zu bedienen. Dies funktioniert jedoch nur bedingt im Bereich der Modelldarstellung und Ergebnispräsentation und hat zugleich auch den Nachteil, dass man sich auf nicht optimierte Methoden aus der Computergraphik festlegt. In diesem Feld hat sich in den vergangenen Jahren sehr viel getan, so dass sich in einem handelsüblichen PC in der Regel eine deutlich leistungsfähigere und funktionsvielfältigere Graphikkarte befindet als in einer teuren Workstation von vor zwei Jahren. Gerade für diese speziellen Workstations sind die Algorithmen aus dem Bereich des CAD zugeschnitten, und diese lassen sich zwar auch zur Darstellung von Simulationsmodellen verwenden, die Möglichkeiten werden aber bei weitem nicht ausgeschöpft. Denn mit der wachsenden Modellkomplexität und der steigenden Anzahl an Freiheitsgraden – im Fall der Struktursimulation also dementsprechend der Anzahl der finiten Elemente – stellen sich neue Anforderungen an die Interaktivität der Visualisierung. Verwendet man hier herkömmliche Visualisierungsansätze, so ist es schwierig, ein aktuelles Gesamtfahrzeugmodell – bestehend aus etwa einer Million finiter Elemente – mit hohen Interaktionsraten darzustellen. Nutzt man hingegen die speziellen Möglichkeiten der aktuellen Graphikkarten aus – wie Multi-Texturierung und Fragment-Programmierung – so kann man entsprechend in anderen Bereichen des Darstellungsprozesses Einsparungen vornehmen und dadurch Engpässe in der Graphikhardware entlasten. Eine solche, gleichmäßigere und performantere Ausnutzung moderner Hardware macht diese Verfahren auch für andere Bereiche interessant und sie könnten umgekehrt den Weg zurück zur Darstellung von CAD-Geometrien finden.

Andererseits lassen sich auch die Bearbeitungsmethoden für CAD und numerische Modelle wie z.B. FE-Oberflächen vergleichen. Dabei ergibt es jedoch keinen Sinn, die verwendeten Algorithmen direkt zu übertragen, da die Anforderungen und Aspekte zu unterschiedlich sind. Stattdessen müssen komplett neue Methoden entwickelt werden. Dies bedeutet nicht, dass sich die Bedienung auf Benutzerseite zwingend ändern muss – wie bisher leider üblich – sondern vielmehr, dass die Algorithmen speziell an die Gegebenheiten der numerischen Modelle angepasst werden müssen und hilfreiche Informationen an den Benutzer zurückgegeben werden. Die Interaktion kann und sollte dabei ähnlich einem CAD-Tool erfolgen oder gegebenenfalls wegen des

höheren Informationsgehalts – d.h. zusätzlichen Anforderungen aufgrund z.B. Elementgestalt, Bauteilverbindungsart, FE-Topologie – von FE-Modellen sogar eher einfacher als bei den teils komplexen Zeichenprogrammen zu handhaben sein.

Wegen der mangelnden Unterstützung der Berechnungsingenieure durch solche Werkzeuge bedeutete das bisher meist einen Bruch im ansonsten sehr effizient abgestimmten Entwicklungsprozess. Zwar hat man nun durch die numerischen Simulationen – und dadurch große Einsparung an realen Prototypen – sehr viel Zeit gewonnen, aber beim Erstellen von Varianten wird immer noch sehr umständlich vorgegangen. Wird bei der Auswertung der Simulationsergebnisse ein Versagen oder ein ungünstiges Verhalten festgestellt, so werden in der Regel kleinere Änderungen an sehr wenigen Bauteilen vorgenommen, welche das Problem beheben sollen. Diese Änderungen müssen dann zuerst an die Konstruktionsabteilung zurückgegeben werden, welche die CAD-Daten entsprechend modifiziert. Hieraus wird anschließend ein neues Simulationsmodell erzeugt, das unter Umständen nochmals von Hand optimiert werden muss. Erst dann kann ein neuer Berechnungsdurchgang gestartet werden. Es liegt auf der Hand, dass durch entsprechende Werkzeuge zur direkten Modifikation der entsprechenden Bauteile auf Basis des Simulationsmodells sehr viel Zeit gespart werden kann, da man den langwierigen Umweg über CAD und Neuvernetzung einspart.

Im Bereich der Darstellung und Visualisierung kann man sich außer der Möglichkeiten moderner Graphikkarten auch Methoden bedienen, die hauptsächlich nur beim Design und nicht bei der Konstruktion Anwendung finden. Auch hier ist es jedoch nicht sinnvoll, diese direkt zu übernehmen. So ist es z.B. nicht sehr hilfreich, wenn ein FE-Gitter in Lebensgröße auf einer Großbildprojektion wie einer Powerwall zu sehen ist. Jedoch die dreidimensionale Stereodarstellung kann bei kompliziert geformten oder nicht direkt offensichtlich gelegenen Bauteilen durchaus hilfreich sein. Hier gilt es neue Wege im Bereich der virtuellen Realität am Arbeitsplatz zu gehen, zum Beispiel durch Einsatz von autostereoskopischen Displays. In Verbindung mit speziellen mehrdimensionalen Eingabegeräten – gepaart mit entsprechenden Modifikationsmethoden – ergeben sich effiziente und intuitive Werkzeuge zur schnellen Generierung von Bauteil- und Fahrzeugvarianten.

Aufbauend auf bereits früher in der Arbeitsgruppe unternommene Anstrengungen, die erwähnten Problematiken zu lösen und neue Ansätze für eine performante Modellvisualisierung zu entwickeln, wurden im Rahmen zweier durch das BMBF finanzierter Projekte weiterführende Methoden und Algorithmen erarbeitet. Innerhalb des ersten Projekts „AutoBench“ wurden sowohl Pre- als auch Postprocessing-Aufgaben, also sowohl Anwendungsgebiete vor und nach der Simulationsrechnung, adressiert. Im Postprocessing-Bereich wurden innerhalb der Forschungsgruppe Visualisierungsverfahren für die Ergebnispräsentation entwickelt, welche die aktuell stetig wachsenden Möglichkeiten moderner Graphiktechnologie konsequent nutzen [Kus98]. Daraus wurden ebenfalls Kenntnisse im Bereich des Preprocessing gewonnen, so dass auch dort u.a. Texturen für die Darstellung skalarer Parameterverläufe oder Geometrieoptimierungsverfahren für einen schnelleren Bildaufbau zum Einsatz kommen. Daneben wurden Methoden für eine schnelle und bequeme Verbindung bzw. Assemblierung unabhängig vernetzter Bauteilkomponenten entwickelt [Som03, Fri04], welche aufgrund der dadurch erzielten Erleichterung der Modellaufbereitung eine entsprechende Verkürzung in der Entwicklungszeit bewirken. Durch die

projektbegünstigte enge Kooperation mit der Karosserieberechnungsabteilung der BMW Group konnten die daraus hervorgegangenen Interaktionsmethoden optimal auf Anwenderbedürfnisse abgestimmt werden.

Darauf aufbauend wurden im Rahmen dieser Arbeit und im Verlauf des Nachfolgeprojekts „AutoOPT“ die Methoden zur interaktiven Generierung von Bauteilverbindungen weiter verfeinert und erweitert. Innerhalb dieses Projekts lag das Hauptaugenmerk auf Preprocessingaufgaben, so dass unter dieser Voraussetzung die graphischen Beschleunigungsverfahren entsprechend angepasst und optimiert werden konnten. Durch Anwendung der durch aktuelle Graphikhardware gegebenen Freiheiten konnten so bereits bekannte Ansätze zur Geometriesimplifizierung komplett überarbeitet und weiterentwickelt werden, so dass sowohl Qualität als auch Leistungsgehalt deutlich gesteigert werden konnten. Die performanten Darstellungsalgorithmen bildeten dabei die notwendige Grundlage für die interaktive Manipulation von Bauteilgeometrien. Auch hier konnte teilweise auf bestehende Arbeiten zurückgegriffen werden [Som03, Fri04], und die anfänglich halbautomatischen Methoden wurden zu einem interaktiv bedienbaren Modifikationswerkzeug ausgebaut, welches den Anwender bei seiner Arbeit mit intuitiv begreifbaren Mitteln und Vorgehensweisen unterstützt.

Die innerhalb des Projekts „AutoBench“ implementierten Algorithmen wurden dabei in die prototypische Applikation „crashViewer“ integriert, welche später im Laufe des Projekts „AutoOPT“ in „FEMod“ umbenannt wurde und unter der Bezeichnung „scFEMod“ mittlerweile kommerziell vertrieben und weiterentwickelt wird. Letzteres unterstreicht dabei die Anwenderorientiertheit und den Nutzen der u.a. im Rahmen dieser Arbeit implementierten Methoden für eine Erleichterung und Beschleunigung des Fahrzeugentwicklungskreislaufs.

Nach einem Überblick über aktuell verfügbare Berechnungsmethoden und in diesem Umfeld eingesetzte Visualisierungs- und Interaktionsmethoden werden in den nachfolgenden Kapiteln die spezifischen Erweiterungen näher erläutert und diskutiert. Dabei sind die Ausführungen im Wesentlichen auf drei Bereiche fokussiert. Zunächst werden Ansätze für eine – sowohl in qualitativer, wie auch leistungsbezogener Hinsicht – optimale Darstellung präsentiert. Dies umfasst Algorithmen zur Verbesserung der limitierten OpenGL Beleuchtungsmöglichkeiten, die schnelle und zuverlässige Detektion von charakteristischen Bauteilmerkmalen sowie deren performante Darstellung und neuartige Simplifizierungsverfahren, welche trotz der – unter Anwendung speziell auf FE-Netze zugeschnittener Strategien – hohen Geometriereduktion eine nahezu identische Bildgebung auch bei zugeschalteter Visualisierung des FE-Gitters erlauben.

Ohne die Durchführung dieser Vorarbeiten wären die später entstandenen Interaktionsmechanismen in der präsentierten Form nicht umsetzbar gewesen. Die im Ingenieursumfeld alltäglich anfallenden Editieraufgaben wurden um intuitive Methoden zur Erzeugung linien- und flächenförmiger Verbindungen erweitert, wobei auf Fehlersicherheit Wert gelegt wurde. Daneben können Sensorpunkte generiert, überprüft und angepasst werden. Als universell einsetzbare Ergänzung zu bestehenden, jedoch im Funktionsumfang relativ limitierten Gittermanipulationsmethoden, wurden Interaktionsmechanismen zu einer bequemen, mausgesteuerten Gittermanipulation geschaffen, welche parallel eine Validierung der Gitterqualität erlauben und auf Wunsch automatisch entsprechend darauf reagieren können.

Sowohl die verbesserten Darstellungstechniken als auch die Gittermanipulationswerkzeuge können durch den Einsatz neuartiger Hardware im Bereich der virtuellen Realität besonders eng verknüpft werden. Daher wird im letzten Bereich auf die Integration entsprechender Geräte – wie autostereoskopische Displays und haptische Eingabegeräte – und die daraus entstehenden Möglichkeiten nochmals gesondert eingegangen. Bei der Umsetzung wurden dabei auch Probleme adressiert, welche sich im Umgang mit konventionellen Arbeitsplätzen nicht ergeben, wie z.B. die sinnvolle Darstellung von Menüs auf derartigen Ausgabegeräten.

“Bistromathics,” he said. “The most powerful computational force known to parascience. Come to the Room of Informational Illusions.”

Douglas Adams: Life, the Universe, and Everything

Kapitel 2

Berechnungsmethoden und Visualisierung strukturanalytischer Simulationen

Die numerische Simulation realer Sachverhalte hat in den vergangenen Jahren immens an Bedeutung gewonnen. Vor allem innerhalb des vergangenen Jahrzehnts hat die Leistung der Computer soweit zugenommen, dass die numerische Simulation großer und komplexer Modelle und Sachverhalte mit vertretbarem Zeitaufwand überhaupt erst möglich wurde. Obgleich die Simulation schon seit längerem meist kostengünstiger war als entsprechende reale Versuche, so wäre ohne leistungsfähige Computer die Wartezeit für ein entsprechend genaues Ergebnis ähnlich lang gewesen wie die Dauer für den Bau eines Prototypen und anschließend damit durchgeführter Versuche. Eine Zeiteinsparung erhielt man nur durch eine angemessen starke Vereinfachung der Simulationsmodelle, was zur Folge hatte, dass man trotzdem parallel dazu viele reale Experimente durchführen musste, um die Berechnungsergebnisse zu verifizieren. Mit zunehmender Rechenleistung, der Verwendung von Parallelrechnern und damit verbundener Parallelisierung der Algorithmen ist die Genauigkeit und Verlässlichkeit soweit gestiegen und gleichzeitig die Rechenzeit soweit gesunken, dass reale Experimente nahezu ersetzt werden können. Somit lassen sich sehr viele teure und zeitaufwändige Versuche vermeiden und durch eine nur wenige Tage andauernde Berechnung auf einem Mehrprozessorsystem ersetzen.

Dieser Trend gliedert sich hervorragend in den bestehenden Entwurfsprozess ein, da auch ein Großteil oder sogar das gesamte Design eines neuen Produktes am Rechner durchgeführt wird und mittlerweile ohne diesen nicht mehr denkbar ist. Das mittels CAD entworfene Modell kann in der Regel halbautomatisch – in einfachen Fällen auch gänzlich automatisiert – durch geeignete Verfahren in ein Simulationsmodell umgewandelt werden. Einen vereinfachten Entwicklungskreislauf zeigt Abb. 2.1. Hier kann man erkennen, dass sowohl beim Design als auch bei den Berechnungsmethoden entweder das Wissen und die Daten vorausgegangener Modelle und Erfahrungen einfließen oder aber die Natur als Beispiel dient. Daraus entsteht meist ein CAD-Modell des neuen Prototyps, welches dann – je nach Simulationsabsicht – ein entweder für Fluidsimulationen oder für Strukturuntersuchungen geeignetes Rechenmodell generiert. Diese sogenannte

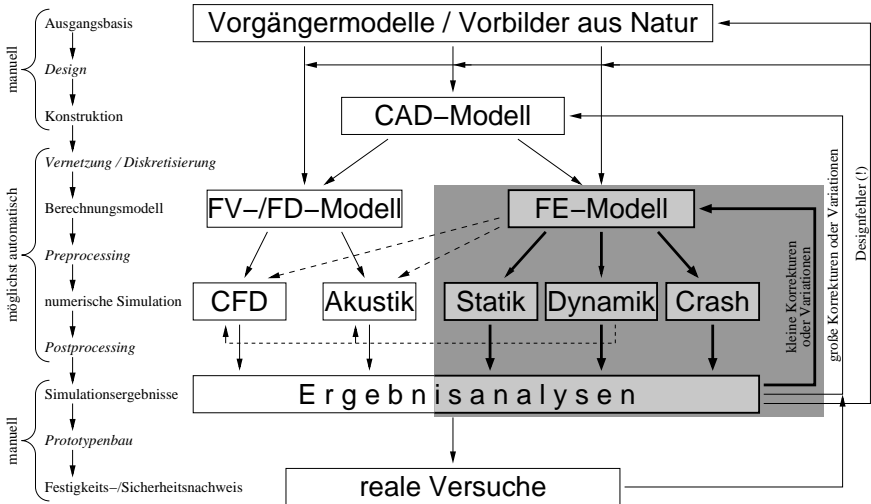


Abbildung 2.1: Abstrahierter Konstruktions- und Berechnungszyklus eines neuentwickelten Fahrzeugs. Der FE-Simulationen betreffende Bereich ist grau hinterlegt.

Vernetzung bedarf zur Zeit noch der manuellen Unterstützung, da es trotz jahrzehntelanger Forschung [BS91] teilweise noch keine perfekten Werkzeuge gibt, die ein optimal rechenfähiges Simulationsmodell erzeugen. Neben den in Abb. 2.1 aufgeführten Methoden „Finite Volumen (FV)“, „Finite Differenzen (FD)“ und „Finite Elemente (FE)“, welche alle diese angesprochene Vernetzung – also die Umwandlung eines Volumens oder einer Oberfläche in eine geeignete Gitterstruktur – voraussetzen, gibt es auch noch andere Verfahren, wie z.B. die netzfreien Methoden oder Monte-Carlo Ansätze. Diese sind jedoch noch relativ jung und noch nicht ausgereift und sollen nicht Gegenstand dieser Arbeit sein. Da man in der Regel aus den CAD-Daten nicht ohne manuelle Nachbearbeitung ein qualitativ hochwertiges¹ FV/FD/FE-Netz erhält, versucht man in den sehr frühen Projektphasen zunächst ohne CAD-Modelle auszukommen und direkt ältere Simulationsmodelle entsprechend den neuen Vorstellungen zu modifizieren. Unabhängig vom Weg, für den man sich letztendlich entscheidet, erfordert die Modellerstellung jedoch immer noch sehr viel Handarbeit, und es herrscht somit ein großer Bedarf an Werkzeugen, die diese Arbeiten unterstützen.

An die Modellerstellung schließen sich verschiedene Simulationsmöglichkeiten an. So kann man im Fall von Fahrzeugen mit Hilfe eines FV- oder FD-Modells aerodynamische Berechnungen durchführen oder die Ausbreitung von akustischen Wellen z.B. im Fahrzeuginnenraum simu-

¹„qualitativ hochwertig“ bedeutet in diesem Fall, dass man bei der numerischen Simulation aussagekräftige und realitätsnahe Ergebnisse erzielt

lieren. In beiden Fällen werden die das Simulationsvolumen begrenzenden Oberflächen meist durch finite Elemente beschrieben und besonders im Bereich der Akustik fließen auch die dynamischen Eigenschaften des Fahrzeugs ein – z.B. in Form der Eigenfrequenzen verschiedener Bauteile. Die Haupteinsatzgebiete der finiten Elemente bilden jedoch strukturmechanische Untersuchungen wie die Berechnung statischer und dynamischer Belastungen oder die Simulation eines Crashverlaufs. Die beiden Erstgenannten lassen sich fast immer durch lineare Ansätze beschreiben und sind daher numerisch vergleichsweise anspruchslos. Die Crashberechnung hingegen erfordert nichtlineare Verfahren und ist dadurch aufwändiger zu handhaben und hat zum Teil sehr strenge Anforderungen an die Beschaffenheit der einzelnen finiten Elemente. Auch hier sind somit Tools gefragt, die den Ingenieur bei seiner Arbeit unterstützen und direkte Aussagen über die Güte eines Finite-Element-Modells ermöglichen.

Der Simulation folgt die Ergebnisanalyse. Bei diesem sogenannten Postprocessing begutachten die Ingenieure verschiedene Eigenschaften und Verformungen des Simulationsmodells. Verschiedene Werkzeuge können die in der Simulation berechneten Kräfte, Spannungen oder z.B. einen Film eines Crashverlaufs darstellen und interessante Bereiche – in der Regel die mit hohen Druck-, Zug- oder Scherspannungen – farblich hervorheben. Fällt das Ergebnis nicht zufriedenstellend aus, so kann entsprechend reagiert werden, und je nach Umfang der Änderung kann man an verschiedenen Stellen wieder in den oberen Teil des Simulationskreislaufs einsteigen. Das Ziel ist in jedem Fall, möglichst wenig Schritte zurückzugehen, die Simulationsschleife also klein zu halten. Die Simulation an sich muss jedoch stets neu durchlaufen werden. Dies ist kein gravierender Nachteil, da dies zum einen im Batchbetrieb – ohne menschliche Interaktion – erfolgen kann und zum anderen in absehbarer Zukunft Mehrgitter- und Kondensierungsverfahren [Bra03] die Wiederverwertbarkeit vorangegangener Ergebnisse ermöglichen sollten, was die Neuberechnung deutlich beschleunigen kann. Mit der entsprechenden Erfahrung lassen sich dadurch reale Versuche auf ein Minimum reduzieren und sie dienen in der Regel nur noch zur Bestätigung der numerischen Rechnung. Sollten erst zu diesem Zeitpunkt gravierende Versagensfälle auftreten, so muss in der Regel wieder ganz von vorne begonnen werden. In diesem Fall sollte man weitreichende Gedanken über das Design oder das der Simulation zu Grunde liegende Berechnungsmodell anstrengen.

Die vorliegende Arbeit entstand in direkter Kooperation mit Ingenieuren der Abteilung *Simulation Passive Sicherheit* der *BMW Group* und konzentriert sich daher hauptsächlich auf Anwendungen im spezifischen Bereich von Crashsimulationen. Dennoch können viele der präsentierten Ansätze und Algorithmen ebenfalls im benachbarten Simulationsumfeld und zum Teil sogar im CAD-Bereich ihre Anwendung finden. Im Folgenden wird eine kurze Einführung in das Gebiet der finiten Elemente gegeben mit einem besonderen Augenmerk auf ihre derzeitigen und zukünftigen Einsatz- und Verwendungsmöglichkeiten bei Crashsimulationen. Dabei werden auch Gemeinsamkeiten und Unterschiede zu anderen Simulationsproblemen angeschnitten.

2.1 Die Finite-Element-Methode

Wie bereits erwähnt, finden numerische Simulationen im großen Stil erst seit einer guten Dekade eine verbreitete Anwendung. Dies bedeutet jedoch nicht, dass die Methoden für solche Simulationen nicht bereits schon deutlich früher existierten. Die Entwicklung und Mathematik der Berechnungsmethoden gehen teilweise sehr weit zurück, noch vor die Erfindung des Computers. So kann man die Geburtsstunde der finiten Elemente auf das Jahr 1851 oder – je nach Definition – sogar bis 1696 zurückdatieren [Ode87]. Die Motivation für die Einführung solcher numerischen Methoden ist damals wie heute die schwierige oder unmögliche Lösbarkeit eines Problems auf analytischem Weg. Ein bekanntes und zugleich komplexes Beispiel hierfür ist die Lösung der Navier-Stokes-Gleichungen [CM93] in der Strömungssimulation mittels Finiter-Differenzen- oder Finiter-Volumen-Verfahren. Bei statischen und dynamischen Struktursimulationen hingegen kommt in der Regel die Finite-Element-Methode zum Einsatz. Bei diesen Festigkeits- und Schwingungsuntersuchungen stellt weniger die Analyse an sich das Problem dar sondern die meist sehr komplizierte Geometrie der modellierten Strukturen. Für einfache Objekte wie Balken oder eine rechteckige Platte lassen sich zum Beispiel Verformungen auch analytisch berechnen, für komplexere Bauteile bietet es sich an, das Problem in mehrere kleine Probleme zu unterteilen. Im Prinzip beschreibt genau dieses Vorgehen die Methode der finiten Elemente anschaulich: Ein komplexer Körper wird in viele kleine, einfache Elemente zerteilt, welche für sich gesehen überschaubar und analytisch oder zumindest näherungsweise lösbar sind. Bei dieser Unterteilung – der sogenannten Vernetzung oder Diskretisierung – verliert eine Oberfläche oder ein Körper die kontinuierliche Struktur und wird durch eine endliche Anzahl von Elementen ersetzt. Es leuchtet ein, dass mit zunehmender Anzahl und somit kleineren Elementen die diskretisierte Geometrie sich der Originalstruktur annähert. Dies bedeutet jedoch gleichzeitig auch einen höheren Rechenaufwand. Die Berechnung komplexer und fein aufgelöster Modelle wurde daher erst durch den Einsatz von Computern ermöglicht.

Zunächst entwickelt und eingeführt im Flugzeugbau [Arg54, Arg55] wurde die Methode der finiten Elemente auch relativ bald für die Lösung einfacherer statischer Belastungsprobleme im Fahrzeugbau verwendet. Die numerische Abbildung eines Crashes hingegen benötigt jedoch hunderttausende einzelner Simulationsschritte, welche den Verlauf eines Crashes mittels Momentaufnahmen im Abstand von wenigen Millisekunden auflöst. Dieser enorme Anspruch an Rechenleistung hat den breiten Einsatz der Finite-Element-Methode für virtuelle Crashtests bis ins letzte Jahrzehnt verzögert.

2.1.1 Grundelemente

Abhängig von der Problemstellung und der Bauteilstruktur kommen bei der Finite-Element-Methode die unterschiedlichsten Grundelemente zum Einsatz. Allen gemeinsam ist die relativ einfache Beschaffenheit und die Möglichkeit, die gewünschten Daten – wie z.B. Verformungen oder Spannungen – analytisch oder näherungsweise zu berechnen. Für die Untersuchung eines Seiles oder eines dünnen Stabes bietet sich das in Abb. 2.2(a) dargestellte Element an. Bei dickeren Stäben oder z.B. einer Radachse kommen in der Regel Balkenelemente wie in

Abb. 2.2(b) zum Einsatz. Für die Modellierung von sehr dünnen Blechen oder Folien werden meist Membranelemente verwendet, welche zweidimensional sind und wie der eindimensionale Zugstab keine Biegemomente aufnehmen können. Bei weniger dünnen Blechen gebraucht man die in Abb. 2.2(c) dargestellten Schalen- oder Plattenelemente. Diese können zusätzlich zu den Membranelementen Biegemomente aufnehmen und übertragen und sind die zur Zeit mit Abstand am häufigsten eingesetzten Elemente im Fahrzeugbau. Hier abgebildet ist die Version mit viereckiger Grundfläche. Flexibler und einfacher für die Vernetzung beliebiger Geometrien sind Elemente mit dreieckiger Grundfläche, diese haben jedoch den Nachteil, dass sie zwar in der Regel Verformungen genauso verlässlich vorhersagen wie viereckige Elemente, jedoch sind die Aussagen bei anderen Werten, wie Spannungen im Elementquerschnitt, weniger zuverlässig. Aus diesem Grund werden – wo möglich – viereckige Schalenelemente gegenüber dem dreieckigen Pendant bevorzugt.

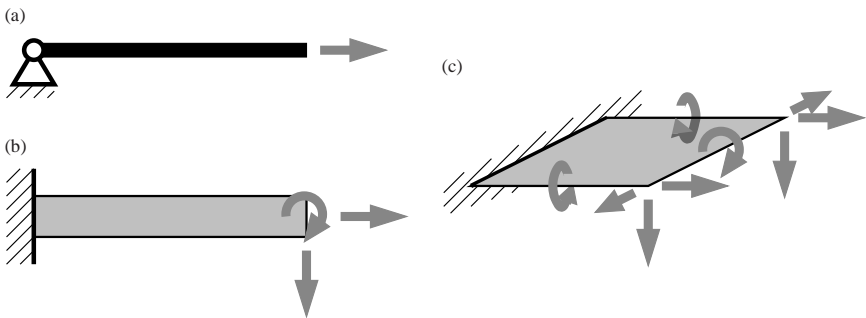


Abbildung 2.2: Beispiele für statische Grundelemente mit unterschiedlicher Komplexität bzw. Anzahl an Freiheitsgraden: (a) Zug-/Druckstab oder Seil, (b) einseitig eingespannter Biegebalken, (c) einseitig eingespannte dünne Platte. Alle Elemente sind entsprechend ihrer Freiheitsgrade auf der linken Seite fest verankert.

Wie man anhand der in Abb. 2.2 an den freien Enden der jeweiligen Elemente eingezeichneten, möglichen Kräfte und Drehmomente erkennen kann, steigt die Anzahl der Elementfreiheitsgrade mit wachsender Dimensionalität. Daher wird an dieser Stelle auch auf die Abbildung eines volumetrischen Elements verzichtet (in der Regel Tetraeder, Prismen oder Quader). Zudem spielten volumetrische Elemente in der Vergangenheit bei der Crashberechnung lediglich eine eher untergeordnete Rolle. Sie hatten ihr Hauptanwendungsgebiet bei der Modellierung von dicken Schäumen, z.B. innerhalb von Türen. Da mit wachsender Anzahl der Freiheitsgrade auch die benötigte Rechenzeit zunimmt, wick man bei der Modellierung von dickeren Blechen fast immer – auf Kosten der Genauigkeit – auf die niederwertigeren Schalenelemente aus. Es zeichnet sich jedoch ein Trend ab, auch zunehmend die dicken Blechteile aus volumetrischen Elementen nachzubilden.

Am Beispiel der häufig eingesetzten Platten- bzw. Schalenelemente soll die Idee zur Lösung eines Problems mit Hilfe der Methode der finiten Elemente kurz erläutert werden. Vereinfachend wird von einer statischen Belastung des Finite-Element-Netzes ausgegangen, es sind also keine kinetischen Energien zu berücksichtigen. In diesem Fall kann man die Verschiebungen und Dehnungen der einzelnen finiten Elemente relativ einfach herleiten. Zugrunde liegt der Arbeitssatz, der besagt, dass die Summe aller inneren und äußeren Arbeiten verschwindet:

$$\sum A = \sum (A_i + A_a) = 0 \quad (2.1)$$

Das heißt, die inneren Spannungen leisten Arbeit auf den Dehnungen ε des Materials und wirken den von außen aufgebracht Lasten – welche Arbeit auf den Verschiebungen verrichten – entgegen. Zu den äußeren Lasten zählt zum Beispiel auch die Schwerkraft, welche jedoch der Einfachheit halber in dieser Einführung vernachlässigt wird. Somit ergibt sich für die Arbeitsgleichung:

$$\sum A = -\frac{1}{2} \int N \varepsilon \, dx + \frac{1}{2} \int p u \, dx = 0 \quad (2.2)$$

Je nach Elementtyp ergeben sich unterschiedliche Terme für die inneren Kräfte N und für die Verformung u unter der aufgebracht Last p . Die Verformungen eines Elements lassen sich allgemein mit Hilfe von – in der Regel polynomiellen – Ansatzfunktionen annähern. Für Son-

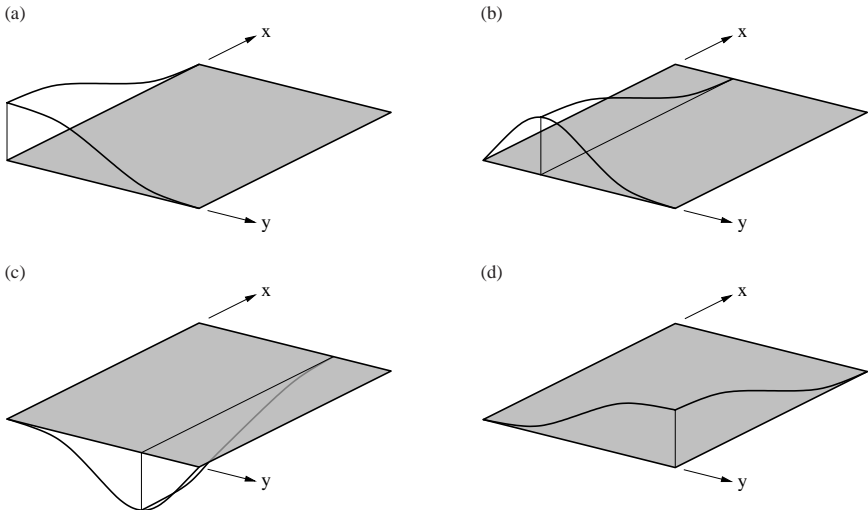


Abbildung 2.3: Ansatzfunktionen für die Biegefläche.

derfälle, d.h. unter bestimmten Belastungen, kann die durch die Ansatzfunktionen beschriebene Verformung deckungsgleich mit der analytischen Lösung sein. Bei Schalenelementen bieten sich die Hermite-Polynome als Ansatzfunktionen an, da sich mit diesen sehr einfach Verschiebungen und Steigungen in den einzelnen Elementknoten wiedergeben lassen. Durch Tensorproduktbildung lässt sich durch diese Basispolynome die Deformation eines solchen zweidimensionalen Elements ausdrücken. Abbildung 2.3 zeigt einen Teil der möglichen Kombinationen. Hier wird jeweils immer das gleiche Hermite-Polynom der ersten Dimension mit vier verschiedenen Hermite-Polynomen der zweiten Dimension verknüpft. Berücksichtigt man alle Varianten, so ergeben sich 16 unterschiedliche Kombinationsmöglichkeiten. Skaliert man diese 16 Grundformen mit entsprechenden Koeffizienten, so kann man damit die Verformung einer Oberfläche näherungsweise so beschreiben, dass auch die Anschlussbedingungen an Nachbarelemente oder Lagerbedingungen berücksichtigt werden können. Soll zudem noch die Verschiebung in der Plattenebene berücksichtigt werden, so kommen weitere Ansatzterme hinzu. Im allgemeinen Fall ergeben sich somit bis zu 24 Freiheitsgrade für ein einziges Schalenelement mit 4 Knoten. Entsprechend komplexer verhalten sich volumetrische Elemente [Wri04]. Der numerische Aufwand wird deutlich und größere Probleme sind somit nur mit leistungsstarken Computern lösbar.

An den Knotenpunkten können unterschiedliche Elementtypen miteinander gekoppelt werden. Hierbei sollte jedoch darauf geachtet werden, dass die Übergänge zwischen den einzelnen Elementen glatt verlaufen, d.h. gleiche Verschiebungen und gegebenenfalls gleiche Torsion. Die Verschiebungs- und Torsionswerte addieren sich dann an den Knoten jeweils zur entsprechenden, gemeinsamen Verformung. Ein Beispiel für ein FE-Netz, gebildet aus dreieckigen und viereckigen Schalenelementen, zeigt Abb. 2.4. An dieser Stelle sei noch erwähnt, dass Knoten nicht

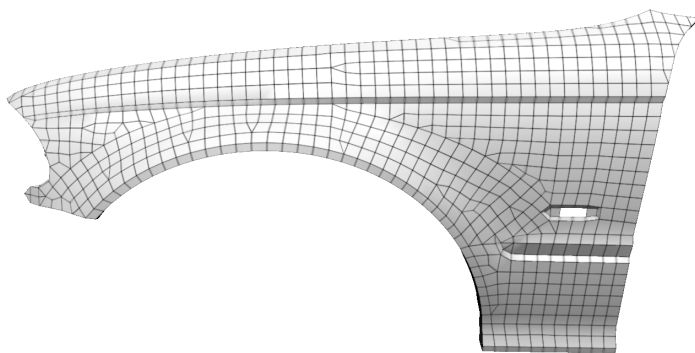


Abbildung 2.4: Beispiel für ein Finite-Element-Netz bestehend aus drei- und viereckigen Schalenelementen: linker Kotflügel eines BMWs.

immer nur an den Eckpunkten eines Elements platziert sein müssen. Es gibt auch Elementtypen, bei denen weitere Knoten auf den Elementkanten oder im Inneren des Elements liegen. Die Arbeitsgleichung (2.2) für ein solches FE-Gitter lässt sich z.B. nach der Methode der virtuellen

Arbeit bzw. dem Prinzip der virtuellen Verschiebungen lösen [GHSW04], welche sich die Eigenschaft zunutze machen, dass im Gleichgewicht bei kleinen Verschiebungen eines beliebigen Knotens der Arbeitssatz (2.1) weiterhin gültig sein muss. Unabhängig vom gewählten Lösungsweg gilt es jedoch letztendlich immer ein – meistens sehr großes – Gleichungssystem zu lösen.

2.1.2 Adaptive Netze

Bei der Erstellung eines FE-Netzes – also der Vernetzung – sollte auf mehrere Faktoren geachtet werden. Um eine annähernd genaue Abbildung der Bauteilgeometrie zu gewährleisten, sollten die Elemente nicht zu groß gewählt werden. Andererseits sollten sie auch nicht zu klein gewählt werden, da dies sowohl die Anzahl der Freiheitsgrade als auch die benötigten Zeitschritte und somit die Rechenzeit immens in die Höhe treibt. Ebenso sollten Elemente nicht zu sehr deformiert werden. In der Regel bedeutet eine Verletzung dieser Faustregeln eine Verschlechterung der Rechenergebnisse, so dass eventuell keine sinnvollen Aussagen mehr über das Verformungsverhalten getroffen werden kann. Will man abgeleitete Größen wie Spannungen betrachten, so ist den oben genannten Regeln sogar noch stärker Beachtung zu schenken, da solche Werte noch empfindlicher auf eine schlechte Vernetzung reagieren.

Unter ungünstigen Umständen kann die Lösung eines großen FE-Gleichungssystems numerisch instabil werden. Vor allem an scharfen Kanten oder im Bereich von Punktlasten kann es zu unrealistisch hohen Spannungen kommen, welche eventuell das Simulationsergebnis des gesamten Bauteils negativ beeinflussen. Treten solche Spannungsspitzen auf, sollte das Netz verfeinert werden oder auf Elemente höherer Ordnung, d.h. Elemente mit mehr Knoten und Freiheitsgraden, zurückgegriffen werden. In der Regel wird jedoch der Konsistenz wegen die erste Variante bevorzugt. Abbildung 2.5(b) zeigt ein, im Vergleich zum Ausgangsnetz in Abb. 2.5(a), verfeinertes Netz, so dass potenzielle Spannungsspitzen im schraffierten Bereich – und ebenso in den drei weiteren Ecken der Vierkantaussparung – vermindert werden. Im äußeren Bereich dieses einfach geformten Bauteils hingegen ist die hohe Auflösung des FE-Gitters sinnlos und erhöht die zur Lösung benötigte Rechenzeit. Eine Methode zur Beschleunigung der Simulation sind adaptive Netze, die das FE-Netz nur in Bereichen hoher Spannung feiner auflösen. Abbildung 2.5(c) zeigt ein mögliches adaptives Netz. Diese Variante hat den Nachteil, dass in den äußeren Gebieten die Elemente stark in die Länge gezogen werden, was unter Umständen – z.B. in Verbindung mit hohen Spannungen – zu verfälschten Simulationsergebnissen führen kann. Eine intelligenter Art der Vernetzung ist in Abb. 2.5(d) zu sehen. Hier kommen sowohl dreieckige als auch viereckige Elemente zum Einsatz. Die Verzerrung der einzelnen Elemente wird dadurch verringert und die Anzahl der benötigten Elemente nimmt im Vergleich zur vorangegangenen adaptiven Vernetzung noch einmal deutlich ab. Dennoch ist die Elementgröße im schraffierten Bereich für alle drei verfeinerten Gitter die gleiche. Nachteil ist, wie bereits angeschnitten, dass die Abbildung der Spannungswerte innerhalb eines dreieckigen Elements weniger genau erfolgt als in einem viereckigen.

Um – in der Wirklichkeit nicht vorhandene – Spannungsspitzen zu vermeiden, kann es gegebenenfalls vorteilhaft sein, sehr kleine Details, wie z.B. Bohrlöcher überhaupt nicht zu modellieren.

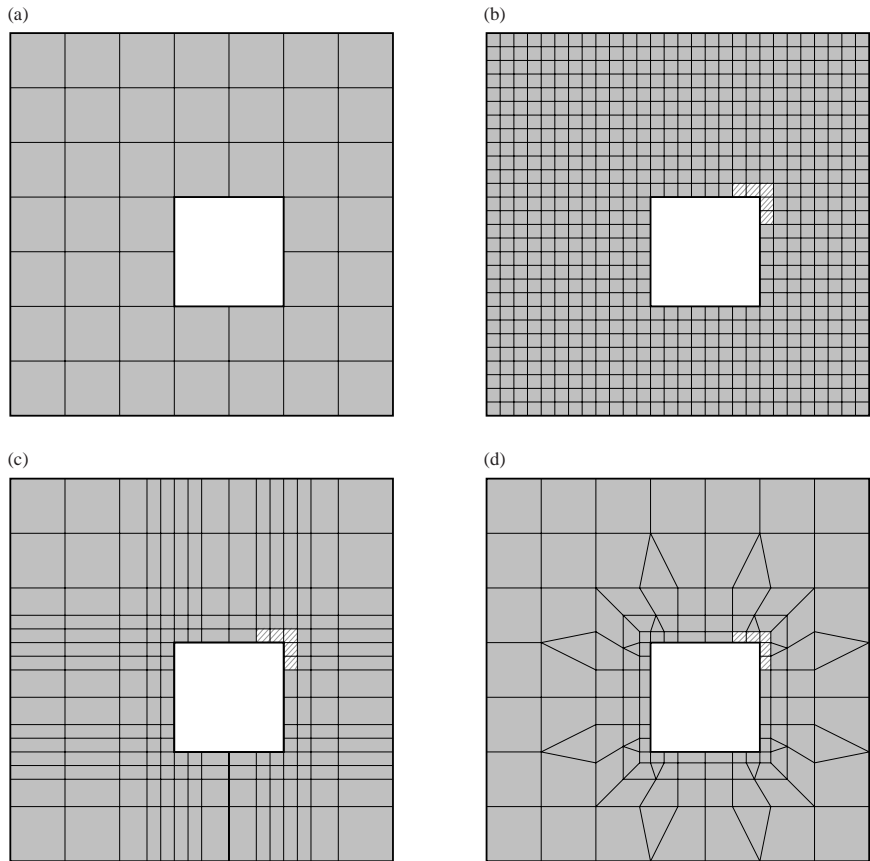


Abbildung 2.5: Strategien zur Netzverfeinerung: (a) grobes FE-Netz (45 Vierecke), (b) uniform verfeinertes Netz (720 Vierecke), (c) adaptiv verfeinertes Netz (459 Vierecke), (d) adaptiv verfeinertes Netz, Mischung zwischen Vierecks- und Dreieckselementen (109 Vierecke und 16 Dreiecke).

Dadurch entfällt auch eine adaptive Verfeinerung des FE-Netzes in der Umgebung des Details. Trotz des minimalen Fehlers, den man durch das Weglassen des Details in Kauf nimmt, ist das Simulationsergebnis meist näher an der Wirklichkeit, da – neben den Spannungsspitzen – die Bildung potenziell entarteter Elemente umgangen wird.

Eine andere Möglichkeit zur Einsparung von Rechenzeit bieten Verbesserungen der Rechenmethoden. Sogenannte Mehrgitter-Verfahren ermöglichen zunächst eine ungenaue Lösung auf einem groben FE-Gitter und verwenden dann spezielle Koppelungstechniken, um das grobe Ergebnis auf das feine Netz zu übertragen und auf diesem das genauere Ergebnis zu berechnen. Die Lösungsfindung erfolgt bei diesem Algorithmus also auf mehreren Skalen [Bra03].

Adaptive Strukturen – sowohl im FE-Gitter als auch in den Lösungsstrategien – können somit zu einer deutlichen Reduzierung der für die Simulation benötigten Anzahl an Elementen führen. Die Rechenzeiten werden folglich ohne nennenswert negative Beeinträchtigung der Simulationsergebnisse reduziert.

Bei vielen Simulationen – z.B. Strömungssimulationen – ist der Versuchsaufbau spiegelsymmetrisch zur vertikalen Mittelebene längs durch das Fahrzeug. Daher genügt es in diesen Fällen meist, lediglich eine Hälfte der Fahrzeugumströmung zu simulieren. Bei Crashsimulationen hingegen ist diese Optimierung wegen der asymmetrischen Massen- und Steifigkeitsverteilung in der Regel nicht möglich.

2.1.3 Verbindungsdefinitionen zwischen Finite-Element-Netzen

In der klassischen FE-Methode wird die Verbindung von benachbarten Bauteilen – die sogenannte Bauteilassemblierung – dadurch hergestellt, dass beide Bauteile an den Verbindungsstellen dieselben Knoten gemeinsam verwenden. Abbildung 2.6(a) zeigt ein Beispiel für solch eine bauteilübergreifende Vernetzung an einem Flansch zweier Bauteile (übereinstimmende Knoten sind weiß markiert). Diese Vorgehensweise bringt mehrere Probleme mit sich. Zum einen ist es schwieriger, einen Kompromiss zwischen Ergebnisgenauigkeit und Berechnungsdauer zu finden, wenn die Gitterauflösung benachbarter Bauteile nicht unabhängig gewählt werden kann. Hier müssen – je nach geometrischer Beschaffenheit der Bauteile in der Nähe der Verbindungsflansche – häufig sehr viel kleinere Elemente zum Einsatz kommen als bei entsprechenden, separat vernetzten Bauteilen. Zum anderen muss, wegen der einzuhaltenden Übereinstimmung der Knoten an *allen* Verbindungsstellen, bei der Netzgenerierung immer das komplette Fahrzeug-

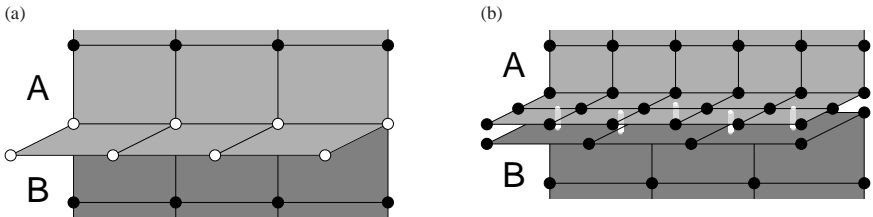


Abbildung 2.6: (a) Gesamtvernetzung versus (b) bauteilunabhängige Vernetzung mit hell dargestellten Verbindungselementen.

modell berücksichtigt werden. Das heißt, die Vernetzung eines Gesamtfahrzeuges ist ein relativ umständlicher Prozess und der Aufwand für den Austausch oder die Veränderung eines einzelnen Bauteils ist daher in der Regel nicht vertretbar. Außerdem können die realen Festigkeitseigenschaften an einer solchen Verbindungsstelle nur sehr grob abgebildet werden.

Abhilfe schaffen neue Methoden der Verbindungsdefinition. Angelehnt an die real gegebenen Techniken ist es seit wenigen Jahren möglich, z.B. Schweißpunkte, Schweißnähte oder Klebeverbindungen analytisch zu repräsentieren und in einer zufriedenstellenden Genauigkeit auf ein Rechenmodell abzubilden [RSD⁺02]. Dies bedeutet, dass die Fahrzeugkomponenten unabhängig – ggf. sogar von unterschiedlichen Dienstleistern – vernetzt werden können (siehe Abb. 2.6(b)) und anschließend mittels dieser neuen, hier hell dargestellten, Elemente virtuell verbunden werden. Die Position dieser Verbindungsstellen ist im Allgemeinen frei innerhalb eines Flanschgebietes wählbar. Wegen der realitätsnahen mathematischen Modellierung sind sie jedoch auch nach wirklichkeitsnahen Gesichtspunkten zu setzen. Ein Schweißpunkt hat somit einen Einfluss auf seine direkte Nachbarschaft und mehrere Schweißpunkte sollten daher, ähnlich der Realität, nicht zu dicht beieinander platziert werden. Der Aufwand für die Einhaltung dieser Randbedingungen ist jedoch eher als Vorteil denn als Nachteil zu werten, da dadurch ggf. spätere Überraschungen beim Bau eines Prototypen vermieden werden können.

Der größte Vorteil bei dieser Verbindungsmethode ist jedoch die sehr dynamische Austauschbarkeit der einzelnen Fahrzeugkomponenten, da nicht auf Deckungsgleichheit des Gitters an Verbindungsflanschen geachtet werden muss. Es können sehr einfach verschiedene Varianten eines Bauteils eingesetzt und getestet werden. Solange die Flanschbereiche sich nicht zu stark ändern, ist sogar eine Neudefinition der Verbindungselemente unnötig, da die alten Positionen weiterhin ihre Gültigkeit haben. Für bestimmte Anwendungsfälle – z.B. bei einer Steifigkeitsberechnung – lassen sich außerdem die FE-Matrizen der unveränderten Nachbarbauteile zu kleiner dimensionierten Steifigkeitsmatrizen kondensieren, so dass die nachfolgende Simulation mit dem ausgewechselten Bauteil sehr viel schneller durchgeführt werden kann.

2.2 Finite Elemente in der Computergraphik

Computergraphik ist mittlerweile ein unumgängliches Mittel geworden, um direkt oder indirekt z.B. das Aussehen und die Qualität eines FE-Modells darzustellen und dementsprechend eventuell Modifikationen daran vorzunehmen. Nach erfolgreichem Simulationsverlauf kann Computergraphik sehr hilfreich in der Beurteilung des Gesamtergebnisses – wie dem Verlauf eines Fahrzeugcrashes – sein. Verschiedene Visualisierungstechniken erleichtern die Begutachtung abstrakter Werte, wie z.B. Farbskalen zur Wiedergabe eines Spannungsverlaufs.

Zum Verständnis der diesen Visualisierungsmethoden zu Grunde liegenden Graphiktechnologien soll an dieser Stelle eine kurze Einführung in die sogenannte Graphik-Pipeline gegeben werden.

2.2.1 Die OpenGL-Graphik-Pipeline

OpenGL, die 1992 veröffentlichte und seitdem stetig weiterentwickelte Open Graphics Language, stellt dem Programmierer eine relativ hardwarenahe Schnittstelle zur Graphikkarte bereit und hat gegenüber anderen Schnittstellen, wie Direct3D, mehrere Vorteile. Der größte ist die Verfügbarkeit auf einer Vielzahl an Plattformen, unter anderem Windows, Linux und IRIX. Außerdem ist es ein bestehender Industriestandard, der in Zusammenarbeit zwischen Graphikkartenherstellern und Anwendern kontinuierlich erweitert und den Bedürfnissen angepasst wird.

Bei OpenGL erzeugt der Programmierer eine zweidimensionale, graphische Ausgabe eines dreidimensionalen Objektes, indem er dessen Daten – wie z.B. Gestalt, also Eckpunkte (Vertizes) und Flächenbeschreibungen, Farbe, Lichtverhältnisse, etc. – an die Graphik-Pipeline übergibt. Der Verlauf der Daten in dieser Pipeline und die Art, wie sie dort aufbereitet werden, lässt sich dabei mittels bestimmter Kommandos beeinflussen. Der grobe Datenfluss ist in Abb. 2.7 skizziert. Ausgehend von einer entsprechend aufbereiteten Szenenbeschreibung der Objekte, Licht-

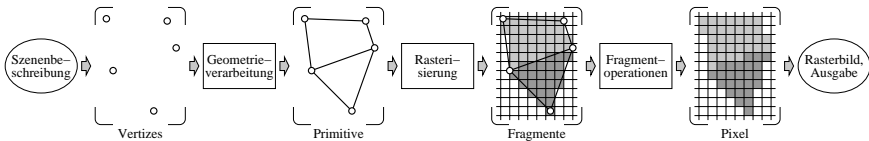


Abbildung 2.7: Darstellung dreidimensionaler Objekte in der Graphik-Pipeline.

quellen und Kamerapositionen wird dabei zunächst die Geometrie verarbeitet. Das heißt, die Vertizes der Objekte werden an die gewünschte Position transliert, rotiert oder skaliert, und die komplette Szenerie wird so vor die Kamera bewegt, dass sie deren Einstellungen Genüge leistet. Anschließend wird die Szene entsprechend der Lichtverhältnisse beleuchtet. Dabei ist anzumerken, dass bei Verwendung von Standard-OpenGL-Befehlen aus Performanzgründen die Beleuchtungsberechnung nur an den Eckpunkten erfolgt und die Helligkeit bzw. Farbe auf den dazwischen aufgespannten Flächen lediglich interpoliert wird (Gouraud Shading). Ebenso findet eine Berechnung des Schattenwurfs in der Regel nicht statt.

Die daraufhin erhaltenen Primitive, meistens Drei- oder Vierecke, werden ggf. durch vom Benutzer eingefügte Schnittflächen abgetrennt. Die komplette Szenerie wird perspektivisch projiziert, so dass der durch die Kamera aufgespannte Stumpf der Sichtpyramide – das sogenannte Frustum – auf einen Einheitswürfel abgebildet wird. Da lediglich Primitive innerhalb des Sichtfrustums gesehen werden können, werden außerhalb liegende Primitive entfernt und die Szene somit auf den Einheitswürfel zurechtgestutzt. Diese Koordinaten können direkt in den zweidimensionalen Viewport abgebildet werden, indem der Tiefenwert ignoriert und die nun zweidimensionalen Koordinaten in die Gerätekoordinaten transformiert werden. Einen Überblick über diese detailliertere Beschreibung der Geometrieverarbeitung bietet der obere Ausschnitt in Abb. 2.8. Diese Geometrieoperationen können dabei alternativ auch durch ein Vertexprogramm abgewickelt werden, welches u.a. eine programmgesteuerte Änderung der Vertexkoordinaten und -farben ermöglicht.

Allerdings genügt es nicht, die gewünschte Manipulation z.B. der Vertexkoordinaten zu implementieren, sondern es müssen auch sämtliche Transformationen und die Beleuchtungsberechnung nachimplementiert werden, da durch die Anwendung eines Vertexprogramms der gesamte Zweig ersetzt wird.

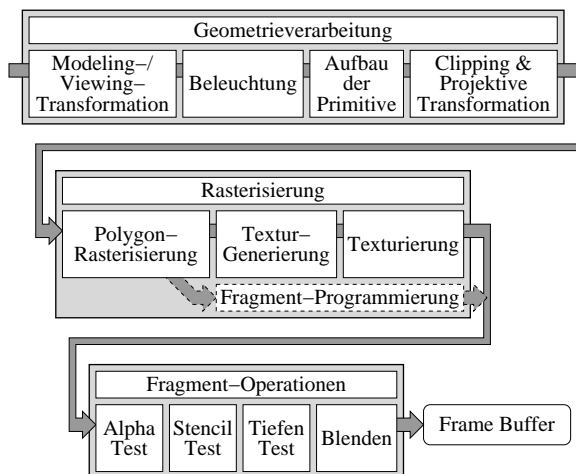


Abbildung 2.8: Detailliertere Ansicht der OpenGL-Graphik-Pipeline.

Mit diesen Informationen können die Primitive nun rasterisiert werden, wobei gleichzeitig die bereits oben erwähnte, zeilenweise Gouraud-Interpolation stattfindet. Ist das gezeichnete Primitiv texturiert, d.h. mit zusätzlichen 1-, 2- oder 3-dimensionalen Bildinformationen versehen, so wird anhand der interpolierten Texturkoordinaten das entsprechende Texel aus dem Texturspeicher geholt und auf das Fragment an dieser Pixelposition angewandt. Alternativ können bei neueren Graphikkartengenerationen auch sogenannte Fragmentprogramme diese Texturierung übernehmen. Zusätzlich bieten diese eine, in gewissem Maße, freie Programmierbarkeit an, wodurch sich der Farb- und Tiefenwert des aktuell rasterisierten Pixels bzw. Fragments im Prinzip beliebig modifizieren lässt. Abbildung 2.8 (Mitte) zeigt diese Wahlmöglichkeit zwischen konventioneller Texturierung und Fragmentprogramm. Ein Fragmentprogramm eröffnet – aufgrund der auf einer aktuellen Graphikkarte mehrfach vorhandenen Pixelpipelines – eine parallele Programmausführung. Durch die gegebene SIMD-Architektur kann dabei eine Instruktion auf viele Fragmente gleichzeitig angewandt werden, so dass innerhalb kurzer Zeit sehr viele, gleichartige Berechnungen durchgeführt werden können. In der Regel dienen Fragmentprogramme dazu, einer Menge an Pixelfragmenten – dem Eingangsdatenvektor – jeweils bestimmte RGB(A)-Farben, d.h. einen entsprechenden Ausgangsdatenvektor derselben Dimension, zuzuweisen. Allerdings wird die SIMD-Programmierbarkeit in jüngster Zeit zunehmend auch für primär nichtgraphische Anwendungen genutzt [KW03].

Bei den abschließenden Fragmentoperationen handelt es sich zunächst um verschiedene Tests, durch die entschieden wird, ob das Fragment auch tatsächlich in den Bildschirmspeicher geschrieben wird. Der wichtigste davon ist der Tiefentest, welcher den Tiefenwert des Fragments mit der bereits an dieser Stelle vorhandenen Tiefeninformation vergleicht. In der Regel wird ein Pixel nur dann gesetzt, wenn es nicht bereits von den im Bildschirmspeicher gezeichneten Objekten überdeckt wird, d.h. also, wenn seine Tiefe geringer als der bestehende Wert ist. Das Speichern der aktuellen Tiefeninformation pro Pixel benötigt zusätzlichen Platz, der sich aber durch die Schlichtheit und Performanz dieses Verfahrens leicht rechtfertigen lässt, da dadurch – zumindest bei opaken Objekten – eine ggf. aufwändige Sortierung im Objektraum zur korrekten Berücksichtigung der Verdeckung wegfallen kann.

Besteht ein Fragment alle Tests, so kann im abschließenden, sogenannten Blending seine Farbe entsprechend seiner Lichtdurchlässigkeit (oder anhand verschiedener anderer Rechenregeln) mit der bereits an dieser Stelle bestehenden Pixelfarbe kombiniert werden, bevor es dann letztendlich in den Bildschirmspeicher geschrieben wird.

Die vorgestellte Abfolge von Berechnungen und Operationen in der Graphik-Pipeline stellt nur einen sehr kleinen Ausschnitt aus dem Funktionsumfang von OpenGL dar. Für weitere Informationen zum Standardumfang von OpenGL sei auf [NDW93] verwiesen. Die zahlreichen Erweiterungen und Neuerungen finden sich auf den Webseiten der Graphikkartenhersteller oder bei [Silb] bzw. [Len03].

Auf der hardwarenahen OpenGL-Schnittstelle bauen verschiedene, komfortablere Schnittstellen auf, welche die Darstellung von Objekten durch einen sogenannten Szenengraphen umsetzen. Ein Szenengraph ist ein gerichteter, azyklischer Graph, bei welchem in den Blättern die Darstellungselemente gespeichert werden. Dabei werden in der Regel auch Lichtquellen, Kameras oder Transformationen in einem entsprechenden Szenengraph-Knoten abgespeichert. Durch die sich daraus ergebende Anordnung in einem Baum (siehe Abb. 2.14) können entsprechende Hierarchien erstellt werden, welche einen gut strukturierten und einfachen Modellaufbau erlauben. Je nach Szenengraph-API werden zusätzlich unterschiedliche Arten der Szenenoptimierung angeboten, so dass die Bildwiederholrate gesteigert werden kann, indem Objekte, welche z.B. die gleiche Textur verwenden, gruppiert werden, oder momentan nicht im Sichtbereich liegende Teile automatisch ausgeblendet werden und somit nicht an die Graphikkarte übermittelt werden müssen.

2.2.2 Visualisierung von Finite-Element-Datensätzen

Um FE-Datensätze zu rendern, also darzustellen, müssen diese zunächst in einem für die Graphik-Pipeline geeigneten Format aufbereitet werden. Dabei ist auf mehrere Faktoren zu achten. Zum einen sollten alle wichtigen Informationen auf einen Blick zur Verfügung stehen, zum anderen sollte die Darstellung dennoch nicht zu überladen oder detailliert sein, da ansonsten das Verständnis sowie unter Umständen die Performanz – und damit die Interaktivität – leidet.

Was dies bedeutet, soll am Beispiel von dünnen Blechen, welche man in der Regel als Schalenelemente modelliert, verdeutlicht werden. Solche Schalenelemente sind einem bestimmten Bauteil zugeordnet und haben als Daten eine gewisse Blechdicke, Materialkennwerte wie Dichte oder

Festigkeitseigenschaften wie z.B. maximale Zugspannung oder E-Modul. Ihre Form beschränkt sich üblicherweise auf Drei- oder Vierecke.

Es ist nicht sinnvoll, alle diese Daten auf einen Blick zu visualisieren, da viele davon die meiste Zeit uninteressant sind und in der Regel eher bauteil- denn elementspezifisch sind. So variieren bei Crashesimulationen normalerweise die Blechdicke, Dichte oder Festigkeitseigenschaften nicht von Element zu Element sondern sind konstant für ein Bauteil². Bauteilweit geltende Werte, bei denen es zudem evtl. auf den genauen Zahlenwert ankommt, sollten daher von vornherein üblicherweise ausgeblendet oder zumindest nicht abstrakt visualisiert werden und ggf. nur auf ausdrücklichen Wunsch des Anwenders eingeblendet werden.

Im Fall von Crashesimulationsmodellen sind für die Interaktion und das direkte Verständnis hauptsächlich die Bauteilzuordnung der Elemente sowie die Form und Lage der Elemente von Interesse. Vor allem in Hinblick auf die Performanz wird meist darauf verzichtet, die Blechdicke explizit darzustellen, obwohl im Fall von unzulässigen Durchdringungen der Bleche auch dieser Wert von Bedeutung sein kann. Man beschränkt sich statt einer korrekten Repräsentation der Blech-Ober- und -Unterseite darauf, lediglich die Mittelebene des Bleches bzw. der Elemente zu zeichnen, was in erster Näherung eine Halbierung des Zeichenaufwands bedeutet. Hierbei ist jedoch darauf zu achten, dass diese Mittelebene von beiden Seiten gesehen werden kann, d.h. Beschleunigungsverfahren wie Backface-Culling können nicht zur Anwendung kommen und die Beleuchtungsberechnung für die Fläche sollte beidseitig erfolgen. Außerdem sollte dem An-

²In anderen Simulationsgebieten, wie z.B. Tiefziehsimulationen, können diese Werte hingegen unter Umständen zwischen benachbarten Elementen stark schwanken.

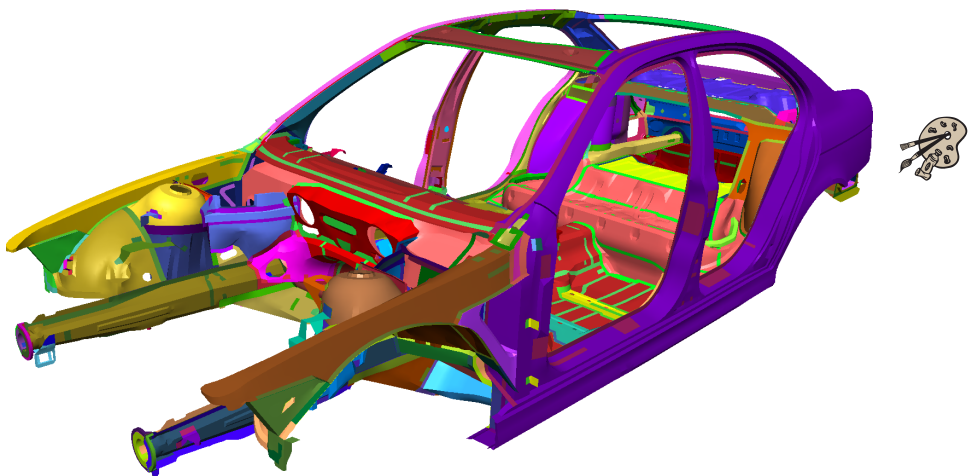


Abbildung 2.9: Farbkodierung verschiedener Bauteile und Komponenten.

wender stets bewusst sein, dass sich benachbarte Bleche – obwohl sich ihre Mittelebenen nicht durchdringen – wegen ihrer endlichen Dicke ggf. penetrieren können.

Die Repräsentation der Bauteilzugehörigkeit erfolgt über eine Farbkodierung, d.h. derselben Komponente zugeordnete Elemente bekommen dieselbe Farbe zugewiesen, welche sich wegen der besseren Wahrnehmung im Idealfall stark von allen benachbarten Bauteilen unterscheidet (siehe Abb. 2.9). Die Darstellung erfolgt, wie bereits erwähnt, schattiert bzw. beleuchtet, wobei sich eine Mischung zwischen Flat- und Smooth-Shading bewährt hat, da hier – in Abb. 2.10(c) gut zu erkennen – scharfe Bauteilkanten besser hervorgehoben werden. Die Beleuchtungsbe-

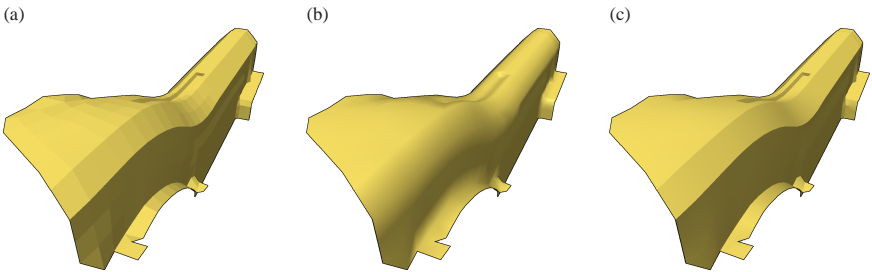


Abbildung 2.10: Unterschiedliche Bauteilschattierungen: (a) Flat, einzelne finite Elemente treten relativ gut hervor; (b) Smooth, glatt aussehende Oberfläche, aber verschwimmende Kanten; (c) Mischung aus beiden Schattierungsvarianten, scharfe Kanten sind klar erkennbar.

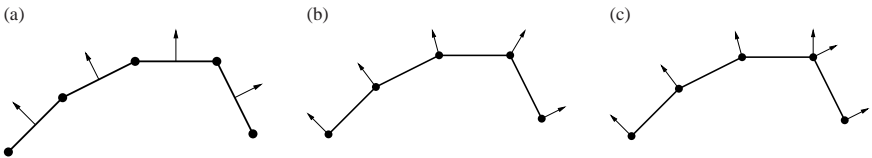


Abbildung 2.11: Für die Beleuchtungsberechnung benötigte Normalenvektoren: (a) Flat, elementweise Normalen; (b) Smooth, gemittelte Normale an jedem Vertex; (c) gemischter Ansatz, doppelte Normalen entsprechend der Elementnormalen an scharfen Kanten.

rechnung erfolgt beim reinen Flat-Shading elementweise (Abb. 2.10(a) und 2.11(a)), beim reinen Smooth-Shading hingegen an jedem einzelnen Vertex (Abb. 2.10(b) und 2.11(b)). Der gemischte Ansatz führt ebenfalls die Beleuchtungsberechnung an jedem Vertex durch, verwendet an einer scharfen Kante jedoch zwei verschiedene Normalenvektoren für Knoten, die auf solch einer Kante liegen. Dabei wird einem Normalenvektor die Normale des Elements auf der einen Halbseite

der Kante zugewiesen, der andere Vektor wird auf die anderseitig benachbarte Elementnormale gesetzt (Abb. 2.11(c)). Da in Standard-OpenGL Vertices keine doppelten Normalenvektoren zugewiesen werden können, müssen solchen Kantenvertices dupliziert werden. Diese Verdopplung kann eine geringe Geschwindigkeitseinbuße bedeuten, wenn man die Daten in einer optimierten Form – z.B. Vertexarrays oder Dreiecks-/Vierecks-Streifen – für eine performante Darstellung aufbereiten möchte. Der in Abb. 2.10(c) klar zu erkennende Gewinn an Darstellungsqualität spricht jedoch für diese etwas aufwändigere Repräsentation. An spitzen Ecken, an denen in der Regel drei scharfe Kanten aufeinandertreffen (ggf. nur zwei, oder sehr selten vier und mehr Kanten) tritt entsprechend eine Normalen- und Vertex-Verdreifachung (oder ggf. Vervielfachung) auf.

Die Beleuchtungsberechnung selbst wird im Allgemeinen nach dem Phong-Blinn'schen Modell durchgeführt, da dieses zusätzlich zum rein diffusen, Lambert'schen Beleuchtungsmodell Glanzlichter bzw. spekulare Reflexionen beinhaltet. Diese spekularen Glanzlichter ergeben zum einen einen realistischeren Gesamteindruck und können zum anderen – ähnlich Reflexionslinien [YK04] – zum besseren Verständnis des genauen Oberflächenverlaufs bzw. der Oberflächenkrümmung dienen.

Die Gesamthelligkeit für eine Farbe λ berechnet sich für das Phong'sche Beleuchtungsmodell [Pho75] unter Berücksichtigung von l verschiedenen Lichtquellen L_i , den diffusen bzw. spekularen Objektfarben O_d und O_s sowie den ambienten und spezifischen Lichtintensitäten I_a bzw. I_{L_i} nach (2.3). Die Reflexionseigenschaften eines Objekts werden entsprechend mittels einem ambienten, einem diffusen sowie einem spekularen Koeffizienten k_a , k_d , bzw. k_s berücksichtigt, die Streuung bzw. „Schärfe“ der spekularen Spiegelung wird durch den Exponenten s geregelt.

$$I_\lambda = \underbrace{I_{a\lambda} k_a O_{d\lambda}}_{\text{ambianter Anteil}} + \sum_{i=1}^l \underbrace{f_{AD_i}}_{\text{atmosph. Dämpfung}} I_{L_i\lambda} \left(\underbrace{k_d O_{d\lambda} (\vec{N} \cdot \vec{L}_i)}_{\text{diffuser Anteil}} + \underbrace{k_s O_{s\lambda} (\vec{R}_i \cdot \vec{V})^s}_{\text{spekulärer Anteil}} \right) \quad (2.3)$$

Mit der Blinn'schen Vereinfachung des Phong-Modells [Bli77] lässt sich das Skalarprodukt aus reflektiertem Lichtvektor \vec{R} und Betrachtervektor \vec{V} durch ein Skalarprodukt aus der Oberflächennormalen \vec{N} und dem sogenannten Halfway-Vektor \vec{H} annähern:

$$\vec{R}_i \cdot \vec{V} \approx \vec{N} \cdot \vec{H}_i \quad (2.4)$$

Hierbei ist der Halfway-Vektor der gemittelte (und anschließend normierte) Vektor zwischen der Lichtrichtung \vec{L} und dem Betrachtervektor \vec{V} , wie in Abb. 2.12 skizziert. Vernachlässigt man zusätzlich die atmosphärische Dämpfung f_{AD} und beschränkt sich auf eine Lichtquelle, so erhält man letztendlich:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + I_{L_\lambda} (k_d O_{d\lambda} (\vec{N} \cdot \vec{L}) + k_s O_{s\lambda} (\vec{N} \cdot \vec{H})^s) \quad (2.5)$$

Gängige Graphikkarten können das Ergebnis dieser Gleichung – auch für mehrere Lichtquellen – für die an jedem Vertex gegebenen Normalenvektoren hardware-beschleunigt berechnen. In der Regel kommt jedoch nur eine Lichtquelle zum Einsatz, da mehrere Lichtquellen zwar

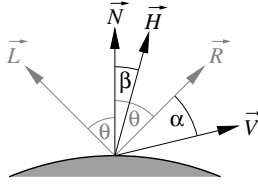


Abbildung 2.12: Der Halfway-Vektor \vec{H} liegt in der Mitte zwischen Lichtrichtung \vec{L} und Betrachtungsrichtung \vec{V} .

realistischere Beleuchtungsverhältnisse schaffen, im Kontext einer möglichst nicht überladenen Repräsentation des FE-Netzes den Betrachter jedoch eher verwirren.

Da bei FEM nicht nur die Gestalt der Oberfläche sondern auch die Form der Elemente selbst sich sehr entscheidend auf ein aussagekräftiges Simulationsergebnis auswirkt, möchte ein Ingenieur zusätzlich die Konturen der einzelnen finiten Elemente einblenden können. Nach konventionellen Vorgehen bedeutet dies, dass im Anschluss an das Rendern des beleuchteten Modells die einzelnen Elemente erneut als schwarzes Drahtgitter darübergezeichnet werden. Durch Anwendung des Tiefentests³ werden dabei nur die sichtbaren Gitterlinien gezeichnet. Dennoch resultiert das zweimalige Traversieren des FE-Modells ungefähr in einer Halbierung der Bildwiederholrate. Einen schnelleren Ansatz zeigt [KHSE98], welcher den zweiten Zeichendurchgang durch die Verwendung einer geeigneten Drahtgittertextur für die Elemente im ersten Durchgang ersetzt. Heutige Graphikhardware kann solch eine Texturierung mit einer zudem vergleichsweise kleinen Luminanz-Textur sehr schnell durchführen und die Bildwiederholrate ist daher deutlich höher als beim konventionellen Zwei-Pass-Verfahren. Abbildung 2.13(b) unterstreicht die zudem hochwertige Qualität des Texturansatzes im Vergleich zum für z-Fighting anfälligen separaten Zeichnen des Drahtgitters in Abb. 2.13(a).

2.2.3 Pre- und Postprocessing

Der vergangene Abschnitt hat grundlegende Methoden für die Darstellung von FE-Datensätzen vorgestellt. Für die weitergehenden Betrachtungen muss zwischen zwei Anwendungsgebieten unterschieden werden. Auf der einen Seite steht das sogenannte Preprocessing. Dabei handelt es sich um die Aufbereitung eines FE-Modells für die Simulation. Preprocessing findet somit vor dem Rechenlauf statt und umfasst Aufgaben wie das Detektieren und Beheben von Materialdurchdringungen, das Verbinden verschiedener Bauteile z.B. mittels Schweißpunkten und Klebeflächen oder das Anbringen von speziellen Sensorpunkten zur Überprüfung der Ergebnisse.

³ggf. in Kombination mit `glPolygonOffset` zur Vermeidung des sogenannten z-Fightings, d.h. eines Flackerns der Drahtgitterlinien, hervorgerufen durch die u.U. nicht exakte Zuordnung desselben Tiefenwerts wie bei der schattierten Darstellung

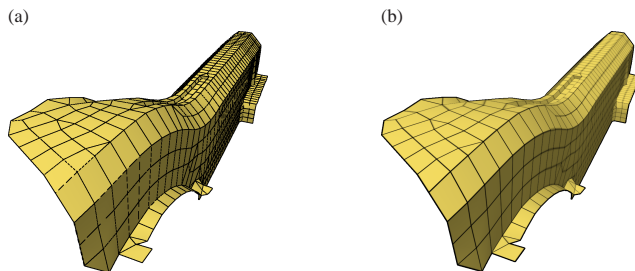


Abbildung 2.13: Repräsentation der Konturen der finiten Elemente mittels (a) Drahtgitter in einem zweiten Zeichendurchgang; (b) Luminanztextur.

Auf der anderen Seite steht das Postprocessing, welches sich an die Simulation anschließt. Hier sind hauptsächlich geeignete und performante Visualisierungstechniken für die animierte Darstellung des Crashverlaufs und verschiedener Ergebnisparameter wie z.B. Spannungsverläufe gefordert.

Preprocessing

Beim Preprocessing ist vorrangig eine gute Bildwiederholrate von Bedeutung, da der Anwender hier direkt mit dem Modell interagiert und Änderungen daran vornehmen möchte, während er beim Postprocessing eher eine Betrachterrolle einnimmt. Bei der heutigen Datensatzgröße von ca. 1 Million meist viereckiger Elemente und somit ca. 2 Millionen zu zeichnender Dreiecke kann es bereits bei der reinen Darstellung – also noch ohne die Durchführung einer Modifikation – zu Performanzproblemen kommen, sofern man keine High-End Graphikkarten einsetzt. Die bereits erwähnte Verwendung von Texturen für die Elementkontur-Darstellung ist an dieser Stelle jedoch bereits ein erster Ansatz für die Lösung dieses Problems.

Für die bessere programmiertechnische Handhabung empfiehlt sich der Einsatz eines Szenengraphen. Die meisten Szenengraphen stellen bereits Methoden für die Beschleunigung der Darstellung zur Verfügung, z.B. View-Frustum-Culling⁴ oder eine geschickte Traversierung der Objekte zur Minimierung von aufwändigen Zustandsänderungen in der OpenGL-Pipeline. In dem innerhalb der Arbeitsgruppe entwickelten Prototyp zur Bearbeitung von FE-Datensätzen kam OpenGL Optimizer/Cosmo3D [Sila] zum Einsatz. Die Gründe für die Auswahl dieses Szenengraph-APIs sind in [Som03] aufgeführt.

Abbildung 2.14 zeigt ein für Preprocessing gut geeignetes Layout eines solchen Szenengraphen. Hierbei wird jedem Bauteil ein eigener *Shape-Knoten* zugeteilt. Über untergeordnete *Appearance-Knoten* wird u.a. festgelegt, in welcher Farbe das jeweilige Bauteil gezeichnet wird

⁴Beschränken der Szene auf im Blickfeld liegende Teile

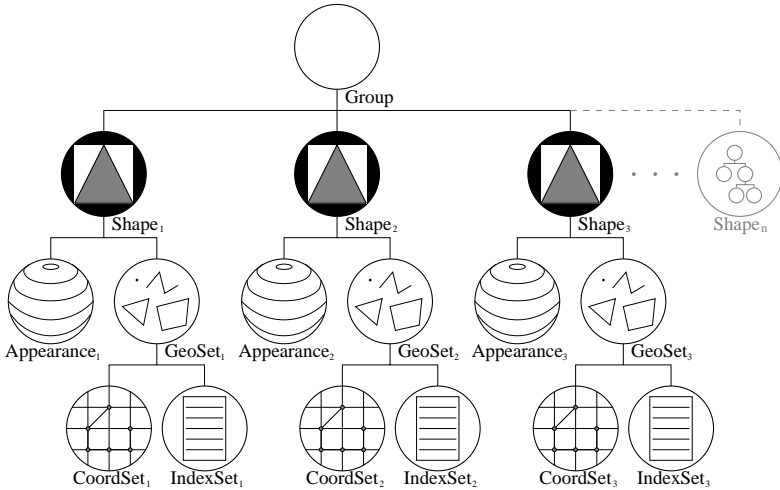


Abbildung 2.14: OpenGL Optimizer/Cosmo3D Szenegraps für eine Preprocessing-Anwendung.

oder ob – und falls ja, wie – ein Drahtgitter der Elementkonturen darübergelegt werden soll. Die *GeoSet-Knoten* enthalten die geometrische Form der Fahrzeugkomponenten, aufgespalten in eine Liste von Vertexkoordinaten (*CoordSets*) und eine Indexliste (*IndexSets*), welche die Vertices adressiert und somit die Konnektivität zwischen diesen herstellt. Diese Art der Entkoppelung spart Speicher und Zeit, da jede Vertexposition nur einmal⁵ an die Graphikkarte übertragen und gespeichert werden muss, obwohl ein Vertex im Schnitt von vier angrenzenden viereckigen Schalelementen verwendet wird. Die ebenfalls zugehörige Ablage der Vertexnormalen in einem *NormalSet* geschieht analog zur Speicherung der Vertexkoordinaten und wurde der Einfachheit halber in Abb. 2.14 nicht aufgeführt.

Bei vielen Preprocessing-Aufgaben können zudem Texturen zur Hervorhebung oder zur Darstellung wichtiger Sachverhalte dienen. Sie können z.B. verwendet werden, um Materialpenetrationen oder -perforationen zu markieren [Som03]. Bei letzteren durchdringen sich die Blechmittelebenen zweier benachbarter Bauteile wie in Abb. 2.15(a) zu sehen. Bei Penetrationen hingegen durchdringen sich die Mittelebenen nicht, jedoch – bedingt durch die endliche Dicke – die beteiligten Bleche (Abb. 2.15(b)). Die Behebung von Perforationen ist meist kritischer, da hier nicht immer eine eindeutige und vollkommen automatische Auflösung des Problems erfolgen kann. In Regelfällen lassen sich solche Probleme dennoch halbautomatisch gut beheben [Kü01].

Im Anschluss an die Nachbesserung invalider Blechdurchdringungen folgt im allgemeinen Arbeitsablauf die Definition von Verbindungselementen in Flanschbereichen. Auch hier können Texturen zur Markierung geeigneter Flanschbereiche den Benutzer unterstützen. Auf Methoden

⁵sofern es sich nicht um einen zu duplizierenden Vertex handelt, siehe Kap. 2.2.2

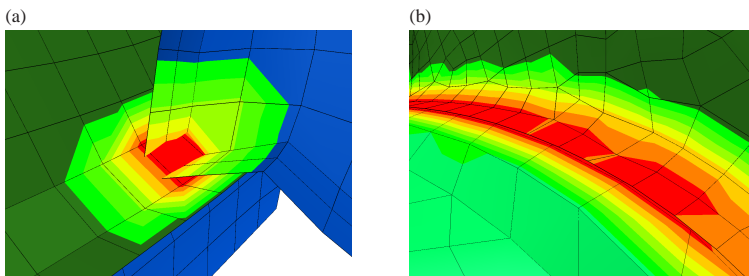


Abbildung 2.15: Materialdurchdringung zweier Bleche: (a) Perforation der Blechmittelebenen; (b) Penetration: Regionen, in denen der Abstand der Mittelebenen geringer als die gemittelte Blechdicke ist, sind rot texturiert.

zur Visualisierung und interaktiven Gestaltung von Schweißpunktnähten wird näher in [FRSE02] eingegangen, Abb. 2.16 soll an dieser Stelle lediglich einen Einblick in die Darstellung von – teilweise als fehlerhaft detektierten – Schweißpunkten geben.

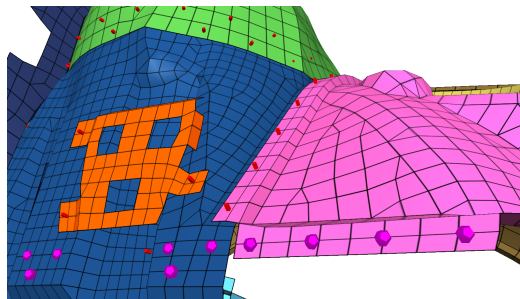


Abbildung 2.16: Schweißpunkte bilden die Verbindung zwischen benachbarten Bauteilen: Rote Quaderstifte stellen korrekte Schweißpunkte dar, violette Dodekaeder markieren das Fehlen einer Bauteilkomponente.

Postprocessing

Wie schon beim Preprocessing ist bei der Darstellung von Postprocessing-Datensätzen ein gewisses Maß an Performanz erforderlich. Jedoch besitzt die Interaktivität der Bildwiederholrate nicht denselben Stellenwert wie beim Preprocessing. Beim Postprocessing kommt es vielmehr auf eine gute und intuitive Präsentation von Ergebniswerten an, die Animation eines Crashverlaufs

muss dabei nicht unbedingt flüssig erfolgen. Dies wäre ohnehin mit einem durchschnittlichen Computersystem nur sehr schwer zu realisieren, da hier ein typischer, aktueller Crash-Datensatz mit ca. 5-8 GB oder mehr nicht ohne weiteres in den Hauptspeicher passt. Möchte man solche Datensätze dennoch visualisieren, so muss eine entsprechende Aufbereitung des Datensatzes erfolgen, so dass möglichst alle redundanten Informationen nur einmal gespeichert werden müssen. Da die Topologie eines Crashmodells sich im Simulationsverlauf nicht ändert (außer man lässt das Reißen von Blechen o.Ä. zu) ist es sinnvoll, diese getrennt nur einmal für den Startzustand abzuspeichern. Bei den nachfolgenden, verformten Zeitschritten müssen die veränderten Knotenkoordinaten zusätzlich gespeichert werden. Für die Topologiedaten, d.h. die Zuordnung welche Elemente aus welchen Vertizes aufgebaut sind, werden jedoch nur Referenzen auf den Ausgangszustand angelegt.

Das Szenengraphlayout aus Abb. 2.14 muss daher leicht abgewandelt werden. Für jeden Zeitschritt bzw. Crashzustand wird eine Struktur ähnlich dem Preprocessing-Szenengraphen angelegt. Allerdings werden nur für den ersten Schritt – den Ausgangszustand – entsprechende *IndexSets* zur Beschreibung der Konnektivität angelegt, in allen folgenden Schritten werden – wie oben erwähnt – lediglich Referenzen auf diese *IndexSets* verwendet. Die Untergraphen der einzelnen Zeitschritte werden durch einen *Switch-Knoten* zu einem einzigen Szenengraph zusammengefasst. Durch ein inkrementelles Weiterschalten dieses *Switches* erfolgt dadurch eine animierte Darstellung des simulierten Crashverlaufs.

In einem Postprocessing-Datensatz sind mit jedem Element bzw. jedem Knoten unterschiedliche Simulationsergebnisse verknüpft. Bei diesen Ergebniswerten kann es sich um Tensoren, meist Spannungen oder Momente, dreidimensionale Vektoren, z.B. Hauptspannungen, oder Skalarwerte wie Intrusionstiefen oder Vergleichsspannungen [Mis13] handeln. Diese Vielzahl an Werten haben einen entscheidenden Anteil an der Größe der Datensätze und stellen ein Problem sowohl beim Einlesen, als auch beim interaktiven Rendern auf Rechnern mit begrenztem Hauptspeicher dar. In den meisten Fällen beschränkt man sich daher auf eine vorher definierte Auswahl an Datentypen. Dabei handelt es sich meist um Skalarwerte, da diese sich besonders einfach durch Farbskala-Texturen direkt auf das Modell abbilden lassen, wie das Beispiel in Abb. 2.17 illustriert.

Auch für die Darstellung vektorieller Daten gibt es bereits interaktive Ansätze [KHSE98], wobei hierbei eine übersichtliche und intuitive Aufbereitung der meist dreidimensionalen Vektoren relativ schwierig ist. Ebenso ist die Wahl einer angemessenen Repräsentation der Datenwerte von komplexeren finiten Elementen nicht trivial. Volumetrische Elemente könnten z.B. halbtransparent dargestellt werden, so dass der Benutzer in den Datensatz hineinschauen kann. Stattdessen beschränkt man sich jedoch üblicherweise auf eine opake Darstellung der Elementoberflächen und der Anwender muss sich mit geeignet gewählten Schnittebenen durch das Innere des Datensatzes navigieren.

Das nachfolgende Kapitel befasst sich mit der Lösung einiger in diesem Kapitel angerissener Problemstellungen. Insbesondere werden Möglichkeiten zur Produktion qualitativ hochwertiger Bildschirmausgabe und intuitivere, schneller zu verstehende Darstellungsmethoden vorgestellt. Parallel dazu wird auf Techniken zur Steigerung der Renderleistung eingegangen.

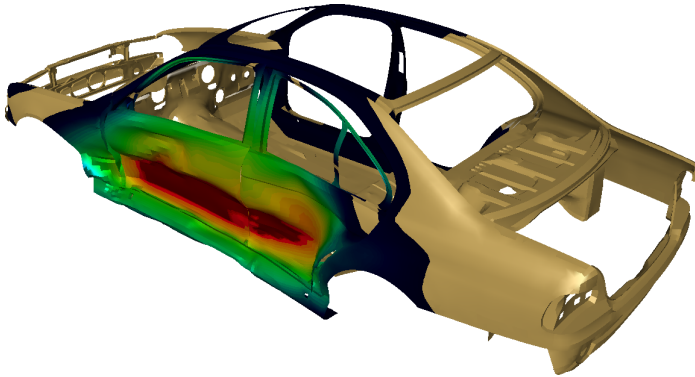


Abbildung 2.17: Visualisierung der Eindringtiefe bei einem Seitenaufprall (aus [KHSE98]).

Its surfaces seemed to be completely smooth, unbroken and featureless.

It was doing nothing. Then Ford noticed that there was something written on it. Strange. There hadn't been anything written on it a moment ago and now suddenly there was. There just didn't seem to have been any observable transition between the two states.

All it said, in small, alarming letters was a single word:

PANIC

A moment ago there hadn't been any marks or cracks in its surface. Now there were. They were growing.

Douglas Adams: Mostly Harmless

Kapitel 3

Optimierung der graphischen Darstellung

Die sinnvolle Aufbereitung von Information und eine gute Qualität ihrer Darstellung tragen entscheidend zum Verständnis eines FE-Datensatzes bei. Dies gilt gleichermaßen für Post- wie Preprocessingdaten. Allerdings sollte beachtet werden, dass die Interaktivität dabei nicht vernachlässigt wird. Insbesondere beim Bearbeiten von Preprocessingdaten wird daher meist ein Kompromiss zwischen Qualität und Performanz eingegangen. Teilweise lassen sich jedoch auch beide Ziele miteinander verbinden. Neue Graphikhardware ermöglicht es mittels ihrer Programmierbarkeit und paralleler Verarbeitung mehrerer Texturen den Flaschenhals des begrenzten Geometriedurchsatzes zu umgehen. Typische FE-Datengrößen liegen zur Zeit bei etwa einer Million Elemente mit einer stark steigenden Tendenz. In diesem Kapitel werden Methoden zur Entlastung der diese Datenmengen verarbeitenden Geometrie-Einheiten vorgestellt und Möglichkeiten präsentiert, wie moderne Graphikhardware sowohl eine qualitativ hochwertige als auch eine schnelle Darstellung mit Hilfe von Texturen und spezieller Programmierung erzielen kann.

3.1 Optimierung der Qualität

Die Darstellungsgüte einer FE-Preprocessing-Anwendung wird im Wesentlichen durch zwei Bereiche geprägt: Die Qualität der Darstellung an sich und die Anschaulichkeit bzw. Wahrnehmungsqualität der in den Datensätzen enthaltenen Informationen. Letztere ist eng verknüpft mit intuitiven Interaktionsmechanismen auf die in Kapitel 4 näher eingegangen wird. Zunächst erfolgt daher die Vorstellung von Methoden zur Verbesserung der Renderqualität.

3.1.1 Beleuchtung und Schattierung

Wie bereits kurz angeschnitten wurde, kommen bei Crashsimulationen im Automobilbau hauptsächlich viereckige Schalelemente zum Einsatz. Sie weisen zwar eine bestimmte Blechdicke auf, ausschlaggebend ist jedoch lediglich die Lage der Blechmittelebene. Daher wird auch nur diese Mittelebene visualisiert. Da ein Blech eine Ober- und eine Unterseite besitzt, sollte die Beleuchtung dieser Mittelebene zweiseitig erfolgen, so dass eine gewisse Ähnlichkeit zur Wirklichkeit erhalten bleibt. Aktuelle Graphikkarten unterstützen solch eine zweiseitige Beleuchtung ohne nennenswerten Einbruch in der Bildwiederholrate und sie wird somit in der Regel auch eingesetzt. Dahingegen ist die Wahl der Beleuchtungs- und Schattierungsmodelle bei OpenGL sehr stark begrenzt. Nativ wird bestenfalls ein Phong-Blinn-Beleuchtungsmodell in Verbindung mit Gouraud-Interpolation unterstützt, wie es bereits in Kapitel 2 vorgestellt wurde.

Für viele Anwendungsfälle ist diese vergleichsweise simple Darstellungsart völlig ausreichend. Abbildung 3.1(a) zeigt jedoch ein Beispiel, bei dem deutlich wird, dass unter Umständen bei Anwendung dieser einfachen Lichtberechnung wichtige Details verloren gehen können, sofern man bestimmte Strukturen – wie das skizzierte wellblechartige Bauteil – unter ungünstigen Winkeln betrachtet. In diesem Fall bringt auch eine fortgeschrittene Interpolationsmethode keine Verbesserung, wie die Anwendung der Phong-Schattierung in Abb. 3.1(b) demonstriert. Bei dieser Schattierungsart werden zwar die Normalen innerhalb eines Elements aufgrund der an dessen Ecken gegebenen Elementnormalen interpoliert und die Beleuchtungsberechnung erst im Anschluss daran durchgeführt, jedoch führt dies in solchen Fällen nicht zum erwarteten Ergebnis.

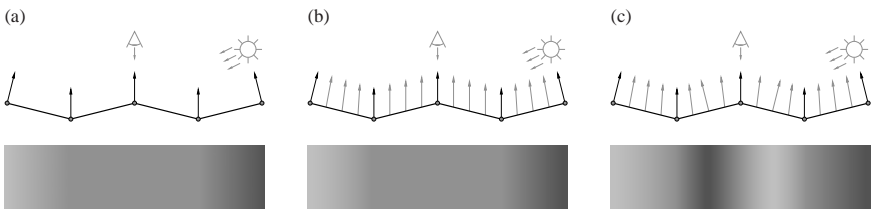


Abbildung 3.1: Normalenverteilung und Beleuchtungsverlauf (direktionales Licht von rechts oben) bei (a) Gouraud-Schattierung; (b) Phong-Schattierung; (c) Pro-Pixel-Normale und -Beleuchtung.

An dieser Stelle stellt sich die Frage, wie Normalen im allgemeinen Fall robust und sinnvoll interpoliert werden können. Abbildung 3.2(a) verdeutlicht den naheliegenden Ansatz, die an den Eckpunkten eines Elements gegebenen Normalen abhängig von einer Interpolationsvariablen $a \in [0; 1]$ gewichtet aufzusummieren. In diesem Beispiel wird zum besseren Verständnis eine Dimension unterschlagen und linear anstatt bilinear interpoliert. Wie man sieht, ist zu beachten, dass die interpolierte Normale im Allgemeinen nicht die Einheitslänge besitzt und diese

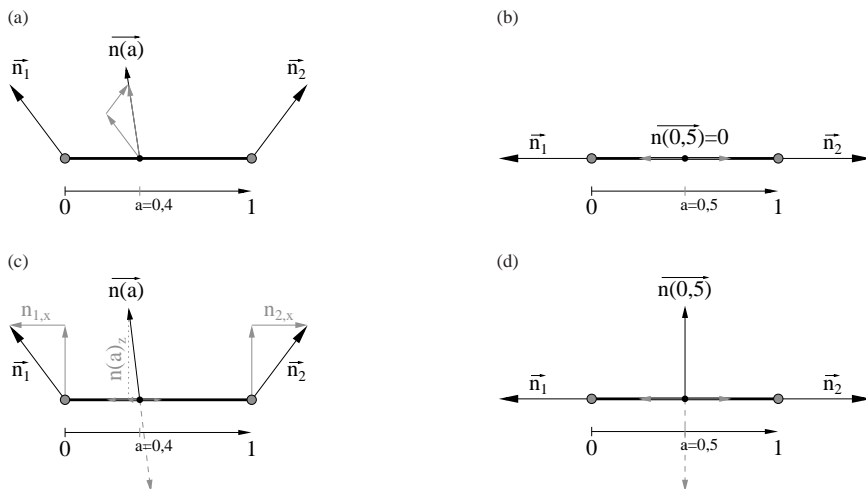


Abbildung 3.2: Robuste Interpolation von Normalen: (a) anteiliges Aufaddieren der Normalen und anschließende Normalisierung; (b) Ggf. kann das Vorgehen aus Abb. 3.2(a) zu Nullvektoren führen; (c) Interpolation der in die Tangentenebene projizierten Anteile und explizite Normalisierung durch entsprechende Berechnung der dazu orthogonalen Komponente $n(a)_z$; (d) Sonderfall resultiert nicht in einem Nullvektor – in welchen Halbraum die Normale zeigt bleibt jedoch zunächst unbestimmt.

somit zusätzlich normalisiert werden muss. Die allgemeine Interpolation von dreidimensionalen Normalen mit anschließender Normierung geschieht bei diesem Ansatz somit nach (3.1).

$$\vec{n}(a) = \frac{(1-a) \begin{pmatrix} n_{1,x} \\ n_{1,y} \\ n_{1,z} \end{pmatrix} + a \begin{pmatrix} n_{2,x} \\ n_{2,y} \\ n_{2,z} \end{pmatrix}}{\left| (1-a) \begin{pmatrix} n_{1,x} \\ n_{1,y} \\ n_{1,z} \end{pmatrix} + a \begin{pmatrix} n_{2,x} \\ n_{2,y} \\ n_{2,z} \end{pmatrix} \right|} \quad (3.1)$$

Diese Vorgehensweise hat jedoch zwei Nachteile. Zum einen ist sie sehr aufwändig, da sehr viele Multiplikationen und eine Wurzel berechnet werden müssen. Zum anderen kann es bei einem Sonderfall – falls zwei Vertex-Normalen in die entgegengesetzte Richtung weisen – vorkommen, dass die Interpolation einen Nullvektor generiert (siehe Abb. 3.2(b)). Somit ist sowohl eine Richtungsgebung der Normalen als auch eine Normalisierung an diesem Punkt nicht möglich.

Einen weniger rechenintensiven Ansatz skizziert Abb. 3.2(c). Hier werden in einem Vorverarbeitungsschritt die Vertexnormalen eines Elements so aufgespalten, dass zwei vektorielle Anteile in dessen Tangentenebene liegen und die dritte Komponente orthogonal dazu bzw. parallel zur Normalen der Elementebene verläuft [Bli78]. Bei der Interpolation werden lediglich die Anteile in der Tangentialebene berücksichtigt. Die dritte Komponente lässt sich anhand der Bedingung, dass die Normale die Einheitslänge besitzen muss, ermitteln. Das Vorzeichen der dritten Komponente ist jedoch unbekannt, lässt sich aber anhand der Orientierung der Elementnormalen oder aufgrund der Traversierungsreihenfolge der Elementvertizes¹ leicht definieren. Bei beidseitiger Beleuchtung ist der Halbraum, in den die Normale zeigt, ohnehin unbedeutend, da es für das Beleuchtungsergebnis gleichgültig ist, ob die Normale in den oberen Halbraum zeigt oder in die entgegengesetzte Richtung, wie in Abb. 3.2(c) gestrichelt angedeutet. Im Weiteren wird davon ausgegangen, dass die Windung der Elemente positiv orientiert ist, d.h. dass die Elementknoten im Gegenuhrzeigersinn angeordnet sind. Entsprechend werden ggf. die Vertexnormalen umgedreht, falls sie in den unteren Halbraum weisen. Somit ist auch die Ausrichtung der interpolierten Normalen klar festgelegt.

Wie (3.2) zu entnehmen ist, kann man auch bei diesem Ansatz die Berechnung einer Wurzel nicht umgehen,

$$\vec{n}(a) = (1-a) \begin{pmatrix} n_{1,x} \\ n_{1,y} \\ 0 \end{pmatrix} + a \begin{pmatrix} n_{2,x} \\ n_{2,y} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \sqrt{1 - ((1-a)n_{1,y} + an_{2,y})^2} \end{pmatrix} \quad (3.2)$$

jedoch werden im Vergleich zu (3.1) einige Multiplikationen und Additionen eingespart. Außerdem ist der in Abb. 3.2(b) vorgestellte Sonderfall robust lösbar, da die orthogonale Komponente erst im Anschluss an die eigentliche Interpolation berechnet wird (siehe Abb. 3.2(d)). Nichtsdestotrotz bleibt die Interpolation von Normalen vergleichsweise aufwändig, weshalb sie auch von OpenGL in der derzeitigen Version nicht standardmäßig unterstützt wird.

Obgleich sie in den meisten Fällen eine erhebliche Verbesserung des Beleuchtungseindrucks erzielen kann, so zeigt auch die Phong-Schattierung nicht immer den gewünschten Beleuchtungsverlauf, wie bereits oben zu Abb. 3.1(b) diskutiert wurde. Der Nachteil der Phong-Schattierung ist, dass die eigentliche Primitiv- bzw. Elementnormale vernachlässigt wird und nicht in das Ergebnis eingeht. Abbildung 3.1(c) zeigt eine Methode, bei der die Normale für jeden Pixel unter Berücksichtigung dieser Elementnormalen neu berechnet wurde. Bei der Interpolation wurde dabei im Prinzip analog zu dem eben erläuterten Algorithmus vorgegangen. Lediglich die Elementnormale wurde als zusätzliche Interpolationsbasis hinzugenommen.

Im oberen Abschnitt von Abb. 3.1(c) erkennt man deutlich, wie auch im mittleren Bereich des Wellblechbauteils die Normalen korrekt den Verlauf der Oberfläche wiedergeben. Aufgrund dieser an jedem Pixel neu berechneten Normalen ist der im unteren Abschnitt angedeutete Beleuchtungsverlauf deutlich aussagekräftiger als in Abb. 3.1(a) oder 3.1(b).

¹d.h. je nachdem ob die Vertizes im Uhrzeiger- oder im Gegenuhrzeigersinn durchlaufen werden

Hardware-beschleunigte pixelbasierte Beleuchtung

Die im obigen Abschnitt erläuterte Normaleninterpolation mit pixelexakter Beleuchtungsberechnung lässt sich auf mehrere Arten unter Ausnutzung unterschiedlicher OpenGL-Erweiterungen umsetzen. Allen im Folgenden präsentierten Lösungen gemein ist die Speicherung der einzelnen Normalen in Texturen. Hierbei werden die 3 Komponenten eines Normalenvektors im RGB-Farbtupel eines Texels abgelegt. Das Konzept ist, jedes Element mit solch einer Normalentextur zu überziehen. Die Auflösung der Textur bestimmt dabei die Abtastdichte des Normalenfeldes auf solch einem Element.

Es hat sich gezeigt, dass es – in Anlehnung an das Nyquist-Shannon Theorem [Sha49] – ausreichend ist, wenn in jede Richtung mindestens zwei, d.h. insgesamt in der Regel vier Normalen pro Element gegeben sind. Die zwischen den Abtastpunkten liegenden Normalen können mittels bilinearer Texturinterpolation gewonnen werden. Bei Verwendung von dreikanaligen Texturen entspricht dieses Vorgehen somit der Interpolation nach Abb. 3.2(a) ohne abschließende Normalisierung. Setzt man zweidimensionale Texturen ein, welche die Normalenkomponenten im Tangentenraum des zu zeichnenden Elements enthalten, so ist die bilineare Texturinterpolation analog zu Abb. 3.2(c) ohne Berechnung der orthogonalen Komponente.

Die Werte der drei Normalenkomponenten liegen im Bereich $[-1; 1]$. Um sie in einer RGB-Textur speichern zu können, müssen diese auf den Integerbereich $[0; 255]$ (bzw. $[-128; 127]$ bei Verwendung einer vorzeichenbehafteten Farbskala) abgebildet werden. Die 8-Bit Genauigkeit der Farbkanäle ist für diesen Zweck gerade noch ausreichend. Bei weniger Bits sind deutliche Qualitätsverluste, bedingt durch eine zu ungenaue Wiedergabe der Normalen, zu erkennen. Mehr Bits hingegen bringen nur kaum sichtbare Verbesserung, meistens bei hochfrequenten Normalenänderungen um relativ kleine Winkel, d.h. bei einer nahezu – jedoch nicht völlig – ebenen Oberfläche.

Unter Verwendung von OpenGL 1.2 Funktionalität können dreikomponentige Normalentexturen pixelgenau beleuchtet werden. Allerdings ist eine Normalisierung der Vektoren mittels des – im Vergleich zum aktuellen OpenGL-Umfang – sehr begrenzten Befehlssatzes nur sehr umständlich und nicht performant realisierbar. Variieren benachbarte Normalen nicht sehr stark, so kann jedoch auf diesen Normalisierungsschritt ohne größere Qualitätseinbußen verzichtet werden. Die Idee dieses ersten Ansatzes ist, die Beleuchtung mittels der `ColorMatrix` und Nachschlagetabellen (`PostColorMatrixColorTable`) zu berechnen. Die genaue Vorgehensweise ist in [WE98] bzw. [RE00] beschrieben und soll hier nicht weiter vertieft werden, da der Ansatz lediglich für ältere Graphikkartengenerationen reizvoll und in Relation zu aktuellen Möglichkeiten vergleichsweise langsam ist.

Eine attraktivere Variante bieten `RegisterCombiners` [Die99] und `TextureShaders` [Silb], welche durch den Graphikkartenhersteller NVIDIA eingeführt wurden. Die meisten von NVIDIA vorgeschlagenen Ansätze zur Beleuchtungsberechnung [Die99] gehen ebenfalls davon aus, dass alle drei Komponenten der Normalen in einer Textur gespeichert werden. Bedenkt man jedoch die große Anzahl an finiten Elementen in einem aktuellen Crashmodell und die damit verknüpfte Dichte von Normalen, so erscheint es sinnvoll, lediglich zwei Komponenten abzuspeichern.

Sind lediglich zweidimensionale Vektoren gegeben, so stellt die Rückgewinnung der notwendigen dritten Komponente in der Graphikkarte das Hauptproblem dar. Eine ebenfalls von NVIDIA vorgestellte Methode, aus zweidimensionalen Normalen diesen dritten Anteil zu berechnen, sind HILO Texturen [DS01]. Diese können zwei Komponenten eines Vektors in einer Auflösung von jeweils 16 Bit speichern und sind für hochqualitative Beleuchtungseffekte z.B. mit sehr großen spekularen Exponenten² gedacht. Sie bieten den Vorteil, dass intern bereits die dritte Komponente – analog zu (3.2) – berechnet wird. Aufgrund des doppelt so hohen Platzbedarfs der 16-Bit-Kanäle ist diese Variante jedoch ungeeignet.

Die bereits genannten TextureShaders bieten jedoch die Möglichkeit, zwei Farbkkanäle als Texturkoordinaten zu interpretieren und mit diesen in einer weiteren Textur nachzuschlagen (*Dependent Lookup*). Ausgehend von den in der ersten Textur gegebenen Normalenanteilen n_x und n_y kann man also in der zweiten Textur die an der entsprechenden Stelle abgelegte dritte Komponente auslesen. Besonders attraktiv wird dieser Ansatz vor allem dadurch, dass man in dieser statischen Nachschlagetextur nicht nur einen, sondern vier Farbkkanäle (RGBA) zur Verfügung hat und somit nicht nur die dritte Komponente nachschlagen kann, sondern stattdessen den bereits normalisierten Vektor. Eine geeignete Vorab-Initialisierung solch einer Textur ist trivial. Es muss lediglich für alle möglichen Kombinationen der beiden Grundkomponenten der aus (3.2) resultierende, normalisierte Vektor eingetragen sein. Solch eine 256×256 große Textur, die für alle n_x, n_y den normalisierten Vektor \vec{n} in RGB-Kodierung zurückliefert, zeigt Abb. 3.3. Da zwischen benachbarten Texeln bilinear interpoliert werden kann, genügt in der Regel auch eine geringere Auflösung dieser Textur von 128×128 oder sogar 64×64 , ohne dass nachfolgende Rechenoperationen sichtbar verfälscht werden. Die schraffierten Bereiche markieren Regionen, die nicht adressiert werden, da die Länge des Vektors niemals größer als 1 werden kann. Entsprechend ist an den Randgebieten des Kreises und außerhalb davon die Komponente $n_z = 0$. Knapp jenseits des Kreises sollte dennoch eine Initialisierung stattfinden, um Seiteneffekte – bedingt durch die bilineare Texturinterpolation – zu vermeiden.

²Bei extrem hohen spekularen Exponenten werden Genauigkeitsverluste aufgrund der ansonsten gegebenen 8-Bit-Auflösung sichtbar.

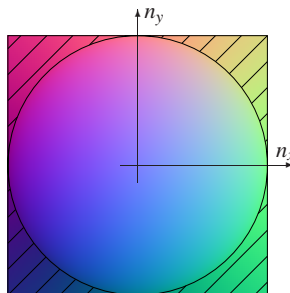


Abbildung 3.3: Dependent Lookup Textur zur Bestimmung der dritten Normalenkomponente bei gleichzeitiger Normalisierung.



Mittels dieser *Dependent Lookup Textur* ist somit eine effiziente Speicherung von zweikomponentigen Normalen und deren performante Normalisierung möglich. Mittels der in der OpenGL-Erweiterung „RegisterCombiners“ vorhandenen Funktionalität können diese Normalenvektoren nun mit weiteren Farbeinträgen kombiniert werden. Hier ist vor allem die Möglichkeit der komponentenweisen Multiplikation zweier Farben, verbunden mit einer anschließenden Aufsummierung, interessant. Kodiert man in der zweiten Farbe die Lichtrichtung \vec{l} , so lässt sich dadurch für jeden Pixel bzw. jedes Fragment eines Elements das Skalarprodukt mit dem dort gegebenen Normalenvektor bilden, und man erhält als Ergebnis dieser DOT_PRODUCT-Operation [Die99] die Helligkeit dieser Fragmente aufgrund diffuser Beleuchtung (siehe (2.5)).

Für die lichtabgewandte Seite eines Elements wird dieser diffuse Beleuchtungswert mit einem negativen Vorzeichen behaftet sein. Bei der Beurteilung, welche Seite der Lichtquelle tatsächlich abgewandt ist, sollte beachtet werden, dass die Normale – wie diskutiert – so gespeichert ist, dass sie immer auf der Oberseite eines Elements steht, wobei Ober- und Unterseite eines Elements durch die Windungsrichtung seiner Vertices definiert wird. Außerdem soll die Beleuchtung beidseitig erfolgen, also muss ggf. auch bei lichtabgewandter Normalen eine positive diffuse Illumination stattfinden. Ob ein Fragment beleuchtet wird oder nicht, hängt somit von der Betrachterrichtung, der Beleuchtungsrichtung, der Normalen und der Elementorientierung ab.

Kombiniert man diese Forderungen, so lassen sich diese relativ einfach erfüllen. OpenGL kann zwischen Ober- und Unterseite bzw. Vorder- und Rückseite eines Primitivs unterscheiden und den beiden Seiten unterschiedliche Farben zuweisen. Die Entscheidung, welche Seite dem Betrachter zugewandt ist, wird dabei ebenfalls anhand der Windungsorientierung der Vertices getroffen. Eine einfache Überlegung zeigt, dass eine Normale genau dann invertiert werden muss, wenn die Unterseite eines Elements vom Betrachter zu sehen ist. Belegt man die Unterseite nun mit schwarz (die farbkodierte Repräsentation des Vektors $\vec{m} = (m\ m\ m)^T = (-1\ -1\ -1)^T$) und die Oberseite mit weiß – entsprechend $\vec{m} = (1\ 1\ 1)^T$ – so kann man durch eine weitere RegisterCombiners-Operation die Normalen komponentenweise mit diesem „Seitenvektor“ multiplizieren und somit im Falle einer sichtbaren Rückseite invertieren.

Der durch die pixelgenaue diffuse Beleuchtung erhaltene Helligkeitswert wird durch die RegisterCombiners-Operationen automatisch jedem Farbkanal zugewiesen und kann dabei auf den Wertebereich $[0;1]$ beschnitten werden, so dass das abschließende Aufmultiplizieren der bauteilweit konstanten Objektfarbe ebenfalls sehr einfach in der sogenannten FinalCombinerStage erfolgen kann. Einen Überblick über den Ablauf dieser hardwarebeschleunigten, pixelgenauen, diffusen Beleuchtung gibt der obere Ausschnitt von Abb. 3.4.

Der ambiente Beleuchtungsanteil ist ebenfalls bauteilübergreifend gleich bleibend und kann dementsprechend als konstante Farbe vorberechnet und zur bestehenden Fragmentfarbe addiert werden.

Analog zur diffusen Beleuchtung könnte die Helligkeit des spekularen Glanzlichts ermittelt werden, indem man die Lichtrichtung \vec{l} durch den Halfway-Vektor \vec{h} ersetzt. Das Problem stellt jedoch die anschließende Potenzierung mit dem spekularen Exponenten s dar. Diese Potenzierung ist zwar für ausgewählte Exponenten $s = \{1, 2, 4, 8, 16, 32, 64\}$ durch entsprechend geschaltete RegisterCombiners realisierbar, die resultierende Darstellungsqualität ist jedoch aufgrund

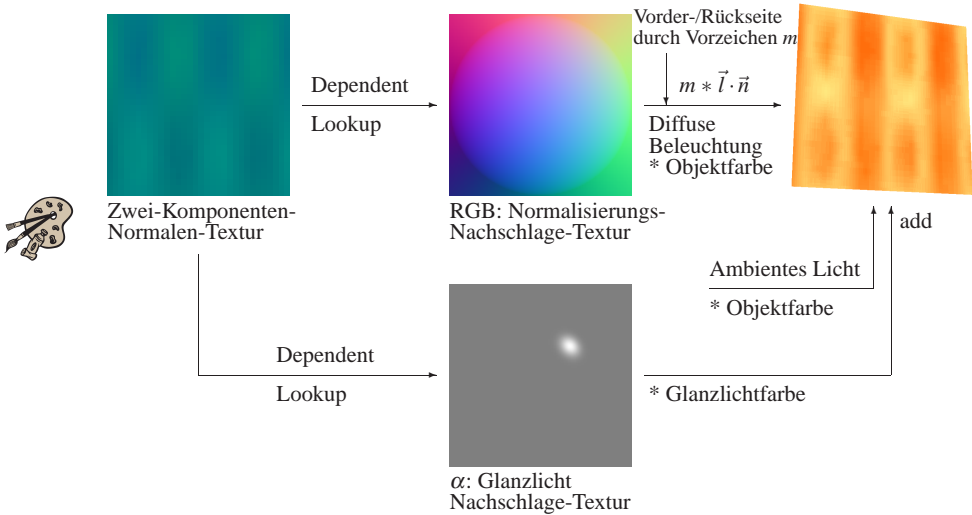


Abbildung 3.4: Beleuchtungsberechnung nach dem Phong'schen Modell für eine mit einer zwei-kanaligen Normalentextur überzogenen Oberfläche.

der geringen internen Bit-Tiefe – und dadurch mit zunehmenden Potenzen stärker ins Gewicht fallenden Rundungsfehlern – nicht zufriedenstellend.

Die Lösung kann erneut eine *Dependent Lookup Texture* sein, welche zu jedem der 256 möglichen Ergebniswerte des Terms $\vec{n} \cdot \vec{h}$ das Ergebnis der mit einem beliebigen, fixen Exponenten s versehenen Potenz enthält. Dies impliziert jedoch einen zweiten – zudem vom ersten abhängigen – *Dependent Lookup*, was teilweise gravierende Einbußen in der Performanz zur Folge haben kann. Von einigen älteren Graphikkarten, welche sich ggf. bei verschiedenen Anwendern des Crash-Preprocessing-Werkzeugs noch im Einsatz befinden, werden solche mehrfach abhängigen Lookups außerdem noch nicht unterstützt.

Unter der Annahme von directionalem Licht und einer relativ parallel-perspektivischen Betrachtung, d.h. bei nicht allzu weitwinkligen Frusta, ist der Halfway-Vektor für die Momentaufnahme einer Szene in erster Näherung konstant. Somit ist das obige Skalarprodukt nur noch vom örtlich gegebenen Normalenvektor abhängig und man kann das Ergebnis inklusive der Potenzierung mit dem spekularen Exponenten für alle auftretenden Normalen vorberechnen. Die gleichermaßen dimensionierte und adressierte Texturen für die Normalisierung der ursprünglich zweikomponentigen Normalenvektoren bietet im verbleibenden Transparenzkanal den Platz, diese vorkalkulierten Werte zu speichern. Somit erhält man im selben *Dependent Lookup* sowohl die normalisierte Normale für die diffuse Beleuchtung als auch die komplett ermittelte Helligkeit des spekularen Glanzlichts.

Diese Glanzlicht-Lookup-Textur muss für jedes Bild neu initialisiert werden, da sich potenziell sowohl Betrachter- als auch Lichtrichtung ändern können. Diese Initialisierung kann auf der Graphikkarte unter Verwendung von geeigneten HILO-Texturen erfolgen, da diese eine ausreichende Genauigkeit besitzen, um auch bei höheren spekularen Exponenten noch zufriedenstellende Resultate zu liefern. Die Ergebnisse können dann in den Alphakanal der bereits bekannten *Dependent Lookup Textur* geschrieben werden. Überraschenderweise ist auf aktuellen Prozessoren die Bereitstellung dieser Alphakomponente schneller zu realisieren als mit den vergleichsweise langsamen HILO-Texturen. Trotz des hinzukommenden Zeitverlusts durch das vor jedem Bild erneute Laden der RGBA-Textur – inklusive des statischen RGB-Anteils für die Normalisierung – in den Graphikspeicher ist die Software-Variante in der Regel schneller als die hardware-basierte Lösung auf der Graphikkarte. Bei Verwendung eines Dualprozessor-Systems kann die benötigte Zeit außerdem nahezu halbiert werden.

Darüberhinaus bietet der software-basierte Ansatz die Möglichkeit, komplexere Beleuchtungsmodelle umzusetzen. So ist es ohne großen Aufwand möglich, das Phong'sche Beleuchtungsmodell anstatt der Phong-Blinn'schen Näherung über den Halfway-Vektor zu realisieren. Laut (2.3) bzw. Abb. 2.12 ist es dazu nötig, den an der Oberfläche reflektierten Lichtvektor zu berechnen. Unter der erneuten Annahme einer – oder ggf. sogar mehrerer – gerichteten Lichtquelle(n) und annähernder Parallelprojektion lässt sich das in (2.3) benötigte Skalarprodukt aus Reflexionsvektor \vec{R} und Blickrichtung \vec{V} , wie eben beim Halfway-Vektor, im Voraus für jedes zweikomponentige Normalentupel mitsamt der anschließenden Potenzierung ermitteln.

Theoretisch ist es analog möglich, die diffuse Beleuchtung in jedem Bild vorzuberechnen. Allerdings ist hierfür die Graphikkarten-basierte Lösung deutlich schneller – die Berechnung des Skalarprodukts $\vec{n} \cdot \vec{l}$ geschieht (auch für mehrere Lichtquellen) nahezu ohne Zeitverlust – und ist in diesem Fall auch nicht so empfindlich gegenüber Genauigkeitsverlusten.

Der untere Zweig in Abb. 3.4 zeigt das parallel zum Normalisierungslookup verlaufende Nachschlagen des spekularen Anteils verknüpft mit der anschließenden Farbgebung durch die Lichtfarbe. Addiert man alle drei Beleuchtungsanteile, so erhält man die exemplarisch mit stark verzerrten Normalen versehene orangene Oberfläche im Bild rechts.

Ein Nachteil ist die Herstellerabhängigkeit dieses Verfahrens. Zwar sind die Graphikkartenproduzenten teilweise bemüht, den Funktionsumfang ihrer Treiber und ihrer Hardware anzupassen und zu erweitern, jedoch müssen für solche Speziallösungen in der Regel für die unterschiedlichen Graphikkartenanbieter verschiedene Programmkerne mit entsprechend zugeschnittenen und optimierten Algorithmen implementiert werden. Einen Ausweg aus der Plattformabhängigkeit bieten `FragmentProgramme`, deren grundlegender, maschinencode-ähnlicher Befehlsatz weitestgehend standardisiert ist. Gegebenenfalls kann man auch an Hochsprachen angelehnte Compiler wie `NVIDIAS Cg` verwenden, um ein höheres Maß an Programmierkomfort bei der Erstellung solcher `FragmentProgramme`, sowie Plattformunabhängigkeit bei nahezu optimaler Ausnutzung der gegebenen Graphikhardware zu erreichen.

Mit Hilfe dieser relativ frei programmierbaren Funktionalität³, welche aber nur von aktuellen Graphikkarten angeboten wird, lassen sich die obigen, etwas umständlich durchgeführten Be-

³konditionale Verzweigungen und Schleifen werden zur Zeit nur sehr rudimentär unterstützt

leuchtungsberechnungen viel direkter umsetzen. Allerdings ist zu beachten, dass z.B. die Wurzelfunktion lediglich in Form einer Näherungsfunktion zur Verfügung steht, welche darüberhinaus recht langsam berechnet wird. Somit behält die Lösungsstrategie, verschiedene Ergebnisse im Voraus zu kalkulieren und in einer Nachschlagetextur bereitzuhalten, auch beim Einsatz von FragmentProgrammen weiterhin ihre Berechtigung.

Abschließend sei hier noch eine Methode vorgestellt, um die Genauigkeit bei der Speicherung von 2D-reduzierten Normalenvektoren zu erhöhen. Wie Abb. 3.5 zeigt, sind vor allem recht flache bzw. fast in der XY-Ebene liegende Normalen durch Diskretisierungsfehler – aufgrund der limitierten Auflösung von 8 Bit (im Bild für lediglich 4 Bit schematisiert) in den Texturfarbkanälen – betroffen. Bei denselben diskreten Abständen in n_x (bzw. n_y) ist die Winkelabweichung ϕ_B bei diesen deutlich größer als die Winkeldifferenz ϕ_A bei entlang n_z ausgerichteten Normalen.

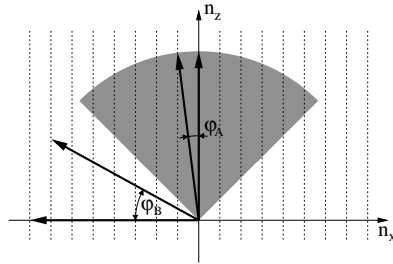


Abbildung 3.5: Die Normalen können nur ungenau in den Textur-Farbkanälen abgespeichert werden, da deren Auflösungsvermögen begrenzt ist.

Um das Auflösungsvermögen zu optimieren liegt es daher nahe, die Normalen immer so zu speichern, dass die beiden betragsmäßig kleinsten Komponenten des Vektors in der Textur gespeichert werden. Die größte Komponente lässt sich dann aus diesen beiden anhand (3.2) ableiten. Für das skizzierte Beispiel in Abb. 3.5 bedeutet dies, dass lediglich Normalen, die innerhalb des grau hinterlegten Bereichs liegen, in dem eingezeichneten Koordinatensystem gespeichert werden. Die beiden hier außerhalb liegenden Vektoren hingegen werden in einem um 90° gedrehten System gespeichert, in dem n_z parallel zur entsprechend größten Komponente orientiert ist. Es gibt somit drei unterschiedliche Koordinatensysteme, je nach Hauptausrichtung des dreidimensionalen Normalenvektors. Durch entsprechende Definition der Nachschlagetextur in Abb. 3.3 kann man anhand dem vorgegebenen Fall die korrekte Normale rekonstruieren. Um ein häufiges Neudefinieren bzw. Laden dieser Nachschlagetextur zu vermeiden, sollten Elemente mit in gleicher Hauptrichtung orientierten Normalen zusammengefasst werden, so dass lediglich dreimal für die drei Orientierungsgruppen ein Wechsel der Nachschlagetextur erfolgen muss.

Auf die sinnvolle Gruppierung der einzelnen, pro Element gegebenen Normalentexturen und deren Packung in gemeinsam genutzte, größere Texturen zur Erhöhung der Performanz wird näher in Abschnitt 3.2.3 eingegangen.

3.1.2 Detektion und Hervorheben von Merkmalen

Für den Wiedererkennungswert eines Bauteils ist nicht nur eine realistische Beleuchtung wichtig, sondern vor allem seine Silhouette in Verbindung mit besonderen Merkmalen wie Löchern oder scharfen Kanten ausschlaggebend. In diesem Abschnitt sollen Algorithmen zur möglichst schnellen und robusten Bestimmung solcher Merkmale, deren sinnvollen Zusammenfassung sowie Methoden zur performanten und artefaktfreien Darstellung diskutiert werden.

Robuste Detektion und Zusammenfassung von Merkmalen

Im Wesentlichen sind zwei Arten von Merkmalen wichtig. Zum einen sind dies die Außenkanten eines Bauteils, wozu auch Ränder von Löchern zählen. Zum anderen prägen vor allem scharfe Kanten das Bild einer Blechoberfläche. Mit diesen beiden Linieninformationen lässt sich in der Regel ein Bauteil bereits sehr gut erkennen, auch ohne eine schattierte Repräsentation, wie in Abb. 3.6(a) demonstriert ist.

Änderungen im Krümmungsverlauf einer Oberfläche hingegen mögen zwar beim Design einer Fahrzeugkarosserie eine gewisse Relevanz besitzen, für das Crashverhalten und die tragenden Fahrzeugkomponenten sind sie jedoch vergleichsweise unwichtig. Zudem sind bei der diskreten Struktur von FE-Netzen Krümmungsänderungen nur sehr schwer genau zu lokalisieren und es kann meist nur mit statistischen Mitteln eine qualitative Aussage getroffen werden. Krümmungsinformationen werden herkömmlicherweise auch nicht direkt in Form von Linienzügen visualisiert sondern indirekt mittels Reflexionslinien, welche relativ einfach mit den im letzten Abschnitt vorgestellten Mitteln oder [YK04] erzeugt werden können.

Außenkanten können sehr einfach detektiert werden. Innerhalb eines Bauteils gehört eine Kante in der Regel zu zwei aneinander angrenzenden Elementen. Am Rand hingegen ist die Kante nur einem Element zugeordnet, da hier das Nachbarelement fehlt. Für eine gegebene Liste an Kanten müssen also lediglich die Elemente gesucht werden, welche die beiden Knoten, die die jeweilige Kante definieren, beinhalten. Findet man nur ein Element, so handelt es sich somit um eine Außenkante und diese kann entsprechend markiert werden. Diese einfache Methode liefert auch im Fall von Löchern das korrekte Ergebnis, wie Abb. 3.7(a) zeigt. Außerdem lassen sich damit Vernetzungsfehler leicht identifizieren, bei denen zwar kein sichtbarer Spalt entsteht, aber unbeabsichtigt ein doppelter Knoten erzeugt wurde, der nicht gemeinsam von benachbarten Elementen benutzt wird, so dass eine Art Haarriß entsteht.

Im Fall eines T-Stoßes (siehe Abb. 3.7(b)) entsteht eine lokale Nicht-2-Mannigfaltigkeit, d.h. hier treffen drei Elemente an derselben Kante aufeinander. An einer Kreuzung teilen sich entsprechend vier Elemente ein und dieselbe Kante. Obwohl es sich hierbei um im Bauteilinneren liegende Elemente handelt, werden diese ebenfalls als Randkante betrachtet. Dies ist zum einen konform mit der Realität, da ein Blech nicht ohne Verschweißung oder Ähnlichem in eine nicht-2-mannigfaltige Form gebracht werden kann, d.h. hier muss sich eine Schnitt- bzw. Außenkante befinden. Zum anderen ist an dieser Stelle eine eindeutige Zuordnung der Oberflächenorientierung nicht möglich. Das heißt, es kann keine klare Aussage getroffen werden, welche Blechseite

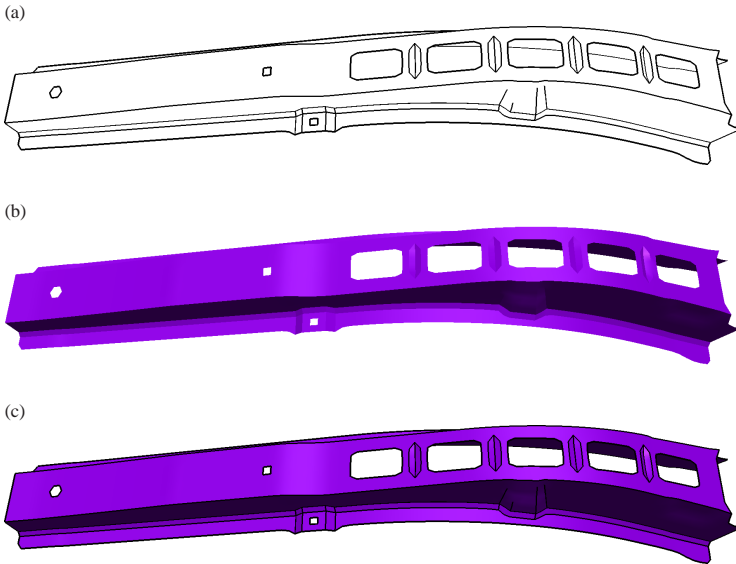


Abbildung 3.6: Die Hervorhebung der Oberflächenmerkmale trägt wesentlich zum Verständnis bei: (a) Die Skizzierung der Außenkanten und scharfen Innenkanten enthält bereits genügend Informationen um das Bauteil zu erkennen; (b) Die schattierte Darstellung allein lässt den hinteren, großteils verdeckten Falz aufgrund der gleichartigen Illumination lediglich erahnen; (c) Darstellung aus Abb. 3.6(a) und 3.6(b) kombiniert.

als Ober- und welche als Unterseite zu betrachten ist. Zudem tritt an solchen Nahtstellen immer eine scharfe Kante auf, da hier die Elementnormalen um mindestens 60° abweichen müssen.

Die Detektion von scharfen Kanten, d.h. Kanten, bei denen die Normalen der beiden angrenzenden Elemente um mehr als einen bestimmten Winkel abweichen, ist hingegen etwas aufwändiger, da nicht immer garantiert ist, dass die Elementnormalen zur gleichen Oberflächenseite gerichtet sind. Dies bedeutet, dass die Normalen ggf. in die entgegengesetzte Richtung zeigen – also einen Winkel von 180° einschließen – obwohl die beiden betreffenden Elemente in der gleichen Ebene liegen und somit keine scharfe Kante bilden.

Um dieses Problem zu umgehen, wird die gemeinsame Kante als lokaler Bezugsvektor \vec{b} gewählt. Ausgehend von einem beliebigen Punkt auf dieser Kante werden zwei Tangentenvektoren \vec{t}_1, \vec{t}_2 zu den Mittelpunkten⁴ der beiden Elemente definiert, wie in Abb. 3.8 skizziert. Damit lassen sich

⁴gewichtetes Mittel der Koordinaten \vec{p}_i der k Elementknoten: $\vec{c} = \frac{1}{k} \sum_{i=1}^k \vec{p}_i$

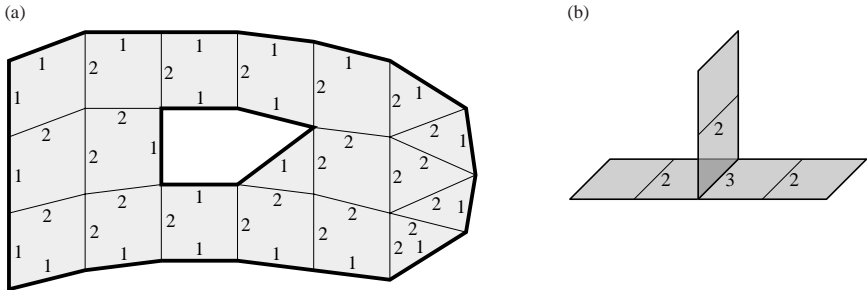


Abbildung 3.7: Ermittlung der Außenkanten einer Oberfläche: (a) Durch Abzählen der Elemente die zur gleichen Kante gehören lässt sich leicht zwischen Innen- (zwei Elemente pro Kante) und Außenkanten (nur ein Element pro Kante) unterscheiden; (b) Bei nicht-2-mannigfaltigen Oberflächen, wie an einem T-Stoß, teilen sich mehr als zwei (hier: drei) Elemente die gleiche Kante.

die – im lokalen Bezugssystem gleichartig orientierten – Normalen wie folgt berechnen:

$$\vec{n}_1 = \frac{\vec{b} \times \vec{t}_1}{|\vec{b} \times \vec{t}_1|} \qquad \vec{n}_2 = \frac{\vec{t}_2 \times \vec{b}}{|\vec{t}_2 \times \vec{b}|} \qquad (3.3)$$

Da die Tangentenvektoren zum jeweiligen Elementmittelpunkt anstatt zu einem der gegenüberliegenden Elementknoten aufgespannt wurden, findet – für den Fall eines nicht planaren Viereckselements – eine implizite Mittelung der betreffenden Elementnormalen statt.

Das Skalarprodukt der beiden Elementnormalen erlaubt nun eine sichere Aussage, wie spitz die Elemente an der gemeinsamen Kante zulaufen. Es hat sich gezeigt, dass Winkel unter 15°

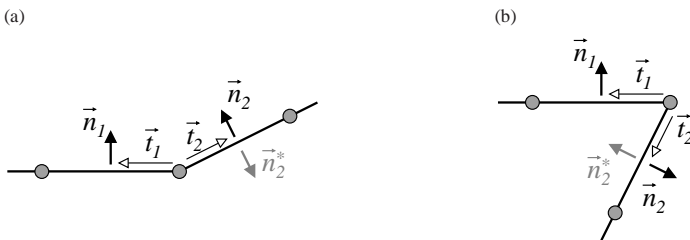







Abbildung 3.8: Bestimmung der lokalen Orientierung der Elementnormalen: (a) annähernd flaches Bauteil, (b) stark scharfkantiges Bauteil, jeweils von der Seite betrachtet.

nicht als scharfe Innenkante betrachtet werden sollten, da ansonsten – aufgrund der relativ grob aufgelösten FE-Netze – auch Rundungen als scharfkantig markiert werden. Ein Winkel von 20° hat sich als guter Kompromiss herauskristallisiert. Damit werden scharfe Kanten sicher erkannt und auch Rundungen mit kleineren Radien nicht irrtümlicherweise als Kante detektiert. Dieser Wert hat sich auch in Zusammenhang mit der in Kapitel 2.2.2 vorgestellten Beleuchtung von Kantenverläufen bewährt.

Die detektierten Außenkanten und abgewinkelten Innenkanten liegen nun in Form voneinander losgelöster, einzelner Knotenpaare vor. Für eine performante Visualisierung müssen diese Einzelkanten zu einem oder mehreren zusammenhängenden Linienzügen zusammengefasst werden. Ein Knotenpaar kann zu diesem Zweck als Dominostein interpretiert werden, dessen beiden Felder mit der jeweils einzigartigen Kennnummer bzw. ID der Knoten dieser Kante beschriftet sind. In Abb. 3.9(b) ist solch ein generischer Dominostein als Repräsentant einer Kante skizziert.

Zur Verdeutlichung des Algorithmus wird das aus den drei Schalenelementen A, B und C bestehende Beispiel aus Abb. 3.9(a) hergenommen. Alle in diesem Beispiel vorkommenden Kanten bzw. die sie repräsentierenden Dominosteine sind in Abb. 3.9(c) aufgelistet. Da in diesem Fall lediglich die Bauteilaußenkanten von Interesse sind, werden alle innenliegenden Kanten – also die Kanten, welche zwei Elementen zugehörig sind – aus der Liste gestrichen. Die verbleibenden Bauteilkanten bzw. Dominosteine werden so angeordnet, dass der Knoten mit der jeweils kleineren Kennnummer an erster Stelle – bzw. in Abb. 3.9(d) an oberer Stelle – steht. Die Dominosteine können dann primär anhand dieser kleineren ID aufsteigend sortiert werden. Der Sinn hinter dieser Sortierung ist der schnellere Zugriff auf die einzelnen Dominosteine.

Mit einem beliebigen Stein aus dieser sortierten Liste kann die Liste zusammenhängender Kanten begonnen werden. Der verwendete Stein (in Abb. 3.9(e) z.B. ) wird aus der sortierten Liste entfernt. Am Ende dieser noch aus einem Stein bestehenden Dominokette können nun passende Steine – d.h. Kanten mit einem Knoten derselben ID – aus der sortierten Liste angehängt werden. Eine binäre Suche liefert jedoch nur dann den passenden Stein zurück, wenn der betreffende Anschlussknoten (hier: ) die kleinere ID auf diesem potenziellen Anlege-Stein hat. Daher wird eine andere Suchstrategie gewählt: Für den bekannten Anschlussknoten werden alle von diesem ausgehenden Kanten bestimmt. Diese Information ist über Nachbarschaftsbeziehungen bereits im Vorfeld gegeben. Anhand der wiederum kleineren Knoten-ID wird gesucht, ob diese Kandidaten (also die Kanten , ) in der sortierten Liste aus Abb. 3.9(d) vorkommen. In diesem Fall taucht lediglich  darin auf, da diese Kante als einzige eine Außenkante ist⁵. Dieses Anfügen neuer Steine an den hinteren Anschlussknoten wird solange fortgesetzt, bis sich entweder keine Steine bzw. Kanten mehr in der sortierten Liste befinden, oder man den Ausgangsstein wieder erreicht und somit einen geschlossenen Ring wie in Abb. 3.9(e) gebildet hat, oder aber keine ans Ende passenden Steine mehr gefunden werden. Im Falle von Außen- bzw. Lochkanten wird man immer einen ringförmigen Kantenzug erhalten. Bei Innenkanten hingegen können offene Enden vorhanden sein, und in solchen Fällen muss zusätzlich versucht werden, am Anfang der Dominokette weitere Steine anzufügen.

⁵bei 2-mannigfaltigen Flächen tritt an einem Randknoten immer genau eine eingehende und eine ausgehende Außenkante auf

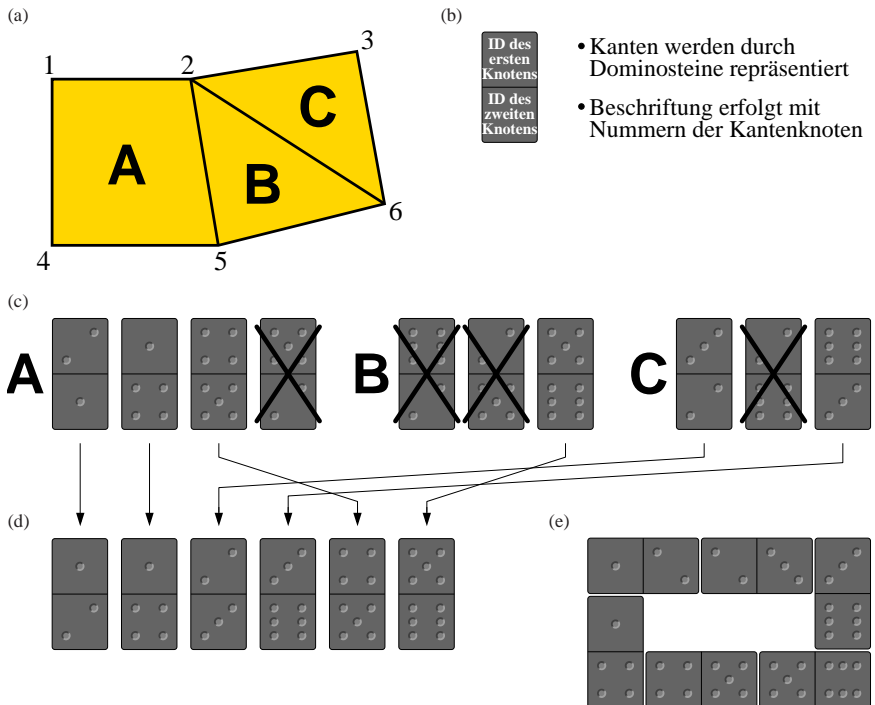


Abbildung 3.9: Das Zusammensetzen einzelner Kanten zu einem einzigen Polygonzug gestaltet sich vergleichbar einem Dominospiel: (a) Bauteil mit drei Elementen A, B und C; (b) Kanten werden durch die aufspannenden Knoten definiert; (c) In diesem Fall sollen nur die Randkanten weiter betrachtet werden; (d) Sortierung der Einzelkanten anhand der kleineren Knoten-ID; (e) Zusammensetzen der Dominosteine zu einer geschlossenen Linie.

Falls nach Abschluss einer Kantenkette noch weitere Steine in der sortierten Liste vorhanden sind, so wird der Algorithmus zum Bau einer weiteren Kette wiederholt, bis alle Kanten „verbaut“ sind. Die erhaltenen Kantenzüge können nun in Form von `GL_LINE_STRIP`s performant dargestellt werden.

Alternativ zur oben gewählten Strategie, alle vom Knoten am Kettenende ausgehenden Kanten auf ihr Vorkommen in der Liste zu überprüfen, kann man auch zwei sortierte Listen verwalten, wobei die zweite entsprechend nach der größeren – anstatt der kleineren – Knoten-ID der jeweili-

gen Kante sortiert ist. Somit findet man die Anschlusssteine, indem man in beiden Listen anhand des Anschlussknotens eine binäre Suche durchführt. Auf den ersten Blick mag diese Variante weniger aufwändig erscheinen, jedoch belegen die folgenden Überlegungen das Gegenteil.

Im ersten Ansatz hat man einen einmaligen Sortieraufwand zum Generieren der Kantenliste. In dieser Liste wird für alle n Kanten bei einem üblicherweise hauptsächlich aus Vierecken bestehenden FE-Netz im schlimmsten Fall dreimal eine binäre Suche durchgeführt, da in der Regel von einem Knoten vier Kanten ausgehen abzüglich der eingehenden, bereits der Kette hinzugefügten Kante. Im Schnitt erhält man jedoch bereits bei der zweiten Suche einen geeigneten Kandidaten, so dass der Aufwand für diesen Algorithmus durchschnittlich insgesamt von der Ordnung

$$\mathcal{O}(n \log_2(n) + \sum_{e=2}^n 2 \log_2(e)) = \mathcal{O}(n \log_2(n) + 2 \log_2(n!)) \quad (3.4)$$

ist. Bei der zweiten Variante hat man den doppelten Sortieraufwand und muss in der Hälfte der Fälle nicht nur in der ersten, sondern auch in der zweiten Liste nach einer geeigneten Kante suchen. Somit ergibt sich hierfür ein durchschnittlicher Gesamtaufwand von

$$\mathcal{O}(2n \log_2(n) + 1,5 \log_2(n!)) . \quad (3.5)$$

Da $\log_2(n!) < n \log_2(n)$ gilt, ist der auf zwei Listen basierende Ansatz für vierecksdominante Netze langsamer und zudem speicherplatzintensiver.

Wie bereits erwähnt und in Abb. 3.6(a) – sowie vergrößert in Abb. 3.10(a) – zu sehen, kann es passieren, dass scharfe Kanten im Bauteilinneren „auslaufen“, d.h. die Normalenabweichung fällt unter den Wert, der als Kriterium für einen sichtbaren Knick angenommen wird. Diese offenen Enden werden entlang der lokal stärksten, ausgehenden Knickkante fortgesetzt, bis man an einem Elementknoten landet, von dem mindestens eine weitere Merkmalslinie abgeht (Abb. 3.10(b)). Diese Verlängerung ist jedoch nur bis zu einem gewissen Grad sinnvoll. Lläuft eine Knickkante in zu flache Regionen, so wird die weitere Suche abgebrochen, um dem Benutzer nicht fälschlicherweise eine scharfe Kante zu suggerieren. Sinnvolle Werte für dieses Abbruchkriterium liegen bei einer Winkelabweichung der Elementnormalen von circa 5° bis 7° .

Um z-Fighting, also das unkontrollierte Flackern der Merkmalslinien beim Zeichnen über die schattierte Bauteiloberfläche zu vermeiden, muss man sich eines Tricks bedienen. Da es sich um Linienzüge und nicht um ein polygonales Netz handelt, kann man die Linien nicht unter Verwendung des `glPolygonOffset` anheben. Zwar könnte man den `glPolygonOffset` verwenden, um umgekehrt die schattierte Repräsentation des Bauteils ein wenig vom Betrachter wegzuschieben, so dass die Merkmalslinien in den Vordergrund rücken, jedoch ist zum einen eine Manipulation der Originaloberfläche unerwünscht oder zumindest fraglich und zum anderen kann dieses Vorgehen zu Darstellungsproblemen in Verbindung mit knapp dahinterliegenden Bauteilen führen. Daher werden bei keinem der beiden Zeichenvorgänge die Koordinaten verändert, sondern stattdessen vor dem Zeichnen der Merkmalslinien der adressierte Wertebereich des Tiefenpuffers mittels `glDepthRange` leicht verschoben, so dass aus Sicht des Tiefenpuffers die Linien stets ein wenig vor den schattierten Flächen zu liegen kommen.

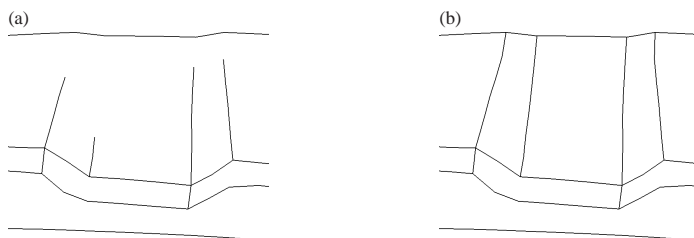


Abbildung 3.10: Robuste Detektion weicher Kanten: (a) Bei flach auslaufenden Knicken brechen die Merkmalslinien ab, sobald die Normalenabweichung unter den Detektionsschwellwert fällt; (b) Verfolgung der angrenzenden stärksten Knickkanten vervollständigt die Kantenlinien.

3.2 Optimierung der Performanz

Bereits in den vorangegangenen Unterkapiteln wurde stets auf eine performante Darstellung der implementierten Erweiterungen geachtet. In diesem Abschnitt soll untersucht werden, wie bereits bestehende und im Einsatz befindliche Vorgehensweisen zum Rendern der Bauteilgeometrie weiter verbessert und beschleunigt werden können.

3.2.1 Generelle Ansätze zur Entlastung der OpenGL-Pipeline

Die große Anzahl an Vertices in einem FE-Datensatz bildet beim Zeichnen die Hauptlast für die Graphikkarte, da sie jeden dieser Punkte mitsamt seiner Normalen transformieren und die Beleuchtungsberechnung durchführen muss. OpenGL selbst stellt verschiedene Mittel bereit, um diese Last zu minimieren, welchen die Idee gemein ist, einen mehrfach verwendeten Vertex nur einmal zu übermitteln. Für ein vierecksdominantes Netz ergibt sich dadurch idealerweise eine Ersparnis um den Faktor vier.

Dies wird erreicht, indem man die Geometrieinformationen, wie Koordinaten, Normalen und ggf. Texturkoordinaten, in einem `VertexArray` speichert. Auf die entsprechenden Einträge in diesem Feld kann dann mittels Indizierung zugegriffen werden. Ein solcher Index muss von der Graphikkarte zum einen nicht transformiert werden und zum anderen ist sein Bandbreitenverbrauch bei der Übermittlung vom Hauptspeicher in die Graphikkarte deutlich kleiner als die Übertragung von jeweils acht Fließkommazahlen⁶. Eine weitere Beschleunigung kann man erzielen, indem man das FE-Gitter in zusammenhängende, jeweils ein Element breite Streifen zerlegt und die entlang eines solchen Streifens abfolgende Reihe von Knotenindizes in Form

⁶3D Koordinate, 3D Normale und 2D Texturkoordinaten summieren sich zu 256 Bit Bandbreitenverbrauch pro Vertex.

eines `QuadStrips` zeichnet [NDW93]. Die Probleme hierbei sind das Auffinden möglichst langer Streifen in der FE-Gitterstruktur und die Behandlung von Dreiecken, welche in der Kompromisslösung als entartetes Viereck mit einem doppelten Knoten aufgefasst werden können [Had98, Som03]. Bei der Wahl geeigneter Streifen lässt sich dadurch die Zahl der benötigten Indizes nahezu halbieren. Der Einsatz von `DisplayList`en, welche unveränderliche Geometrieteile auf der Graphikkarte ablegen, so dass eine andauernde Übertragung der Daten über den AGP- bzw. PCI(e)-Bus entfällt, ist in Verbindung mit `VertexArrays` nur bedingt ratsam. Bei aktuellen Übertragungsraten vom Hauptspeicher in den Graphikkartenspeicher fällt die Dauer des `VertexArray`-Transfers nicht so sehr ins Gewicht wie der Performanzverlust aufgrund der relativ speicherintensiven Repräsentation des Feldes in Form einer `DisplayList`e. Eine Alternative zu `DisplayList`en bilden `VertexBufferObjects`, durch deren Einsatz sich in der Regel höhere Bildwiederholraten erzielen lassen. Allerdings kann es auch hier – in Abhängigkeit von eingesetzter Graphikhardware, installierter Treiberversion und gewählter Genauigkeit – zu teilweise hohen Geschwindigkeitseinbußen kommen.

Möchte der Anwender zusätzlich die Drahtgitterrepräsentation über der schattierten Modell-darstellung einzeichnen lassen, so müssen bei der konventionellen Vorgehensweise zumindest die Vertexkoordinaten ein zweites Mal traversiert werden. Die Normalen und Texturkoordinaten werden zum Zeichnen dieser in der Regel schwarzen Umrandungslinien nicht benötigt. In Kapitel 2.2.2 wurde bereits ein Ansatz vorgestellt, der den zweiten Zeichendurchgang überflüssig werden lässt. Einzige Nachteile hierbei sind, dass für die Drahtgittertextur zusätzliche Texturkoordinaten übertragen werden müssen und bei Verwendung von Hardware ohne Multitexturen-Unterstützung eine gleichzeitige Anwendung anderer Texturverfahren wie z.B. Parametertexturen zur Visualisierung skalarer Datenwerte nicht möglich ist.

Interagiert der Benutzer nur in einem bestimmten Bereich mit dem Datensatz, indem er z.B. ein einzelnes Bauteil dreht, so ist es nicht notwendig, den ganzen Datensatz für jedes Bild erneut zu zeichnen, sondern lediglich die Geometrie des bewegten Teils. Dazu wird zu Beginn der Interaktion das gesamte Modell ohne die ausgewählten, dynamischen Komponenten gezeichnet und dieser Zustand abgespeichert, d.h. es werden Farb- und Tiefenpuffer des statischen Szenenanteils in einer Textur abgelegt. In der darauffolgenden Benutzerinteraktion wird für jedes Bild der Farb- und Tiefenpuffer mit diesem statischen Anteil aus der Textur initialisiert und anschließend die dynamische Unterszene wie üblich gezeichnet. Da bei jedem Durchgang auch die Information des Tiefenpuffers restauriert wird, ist auch für diesen bewegten Anteil eine korrekte Verdeckungsbehandlung möglich. Um eine weitere Beschleunigung des Verfahrens zu erreichen, kann man das Initialisieren des Bildpuffers auf den Ausschnitt des Bildes beschränken, der im vorangegangenen Schritt verändert wurde. Das Ermitteln des betroffenen Bereichs erfolgt üblicherweise anhand der Bounding Box der bewegten Komponente.

Die beiden zuletzt vorgestellten Methoden beinhalten bereits erste Ansätze, die Geometrielast der Graphikkarte auf Kosten ihrer Füllrate zu reduzieren. Weiterführende Umsetzungen, die in aktueller Hardware vorhandene, teilweise enorme und zudem parallelisierte Texturierungs- und Füllleistung auszureizen, folgen in Abschnitt 3.2.3.

3.2.2 Adaptive Detaillierungsstufen

In vielen Fällen liegt nicht das gesamte Fahrzeugmodell im Interesse des Ingenieurs. Um eine bessere Interaktivität zu erreichen, wird daher meist nicht mit einem Gesamtfahrzeugmodell gearbeitet, sondern lediglich mit einem spezifischen Ausschnitt daraus. Es werden also Untermenüen des Originaldatensatzes erstellt, die anschließend wieder zu einem Gesamtfahrzeug zusammengefügt werden. Problematisch bei dieser Vorgehensweise ist jedoch die Tatsache, dass man bei einer Änderung in solch einem Unterdatensatz nicht erkennen kann, wie sich diese Operation auf ggf. benachbarte – in diesem Submodell aber nicht vorhandene – Bauteile auswirkt.

Eine Lösung des Problems bieten adaptive Methoden, welche es ermöglichen, Komponenten die von großem Interesse sind, in ihrem höchsten Detailgrad zu präsentieren, andere, weit davon entfernte Bauteile, die unter Umständen unwichtig sind, hingegen nur in einer sehr groben Auflösung zeigen. Somit hat man zwar ggf. nicht mehr alle Nachbarregionen eines Interessenzentrums sehr genau dargestellt, aber ist zumindest über die gröberen Umrisse informiert und kann die Umgebung besser einschätzen, als wenn sie komplett weggelassen wird. Da die im Gesamtmodell entfernter liegenden Bauteile bei solch einem adaptiven Verfahren nur noch mittels einer sehr groben Netzstruktur skizziert werden, ist der Zeichenaufwand für die Graphikkarte entsprechend geringer als bei einem überall bis auf den feinsten Detailgrad aufgelösten Originalmodell.

Progressive Meshes

Eine hinreichend erforschte Methode, dies zu erreichen, sind die sogenannten „Progressive Meshes“ [Hop96]. Bei diesem Verfahren wird in einem Vorverarbeitungsschritt ein bestehendes Netz so lange vereinfacht, bis es eine vorgegebene gröbste Auflösung erreicht hat. In der Regel werden bei dieser Vereinfachung die Vertices des Originalnetzes wieder verwendet, so dass hierfür kein zusätzlicher Speicheraufwand anfällt. Während der Vereinfachung des Netzes werden sukzessive einzelne Vertices aus dem Netz entfernt, wobei anhand einer Prioritätenliste entschieden wird, welcher Vertex als nächstes herausgenommen wird. Ein Maß für diese Priorität kann z.B. die Krümmung der Oberfläche sein oder die Entfernung des betreffenden Knotens zur entstehenden simplifizierten Fläche. Für FE-Bauteile ist das zuletzt genannte Kriterium sinnvoller, da sich Krümmungen – wie im letzten Abschnitt erwähnt – u.U. nur recht umständlich genauer ermitteln lassen aufgrund der diskreten Auflösung der gegebenen Oberflächen. Nach dem Entfernen eines Vertex wird die verbleibende Umgebung im Gegensatz zu einigen anderen Simplifizierungsverfahren nicht neu trianguliert, sondern ein sogenannter „Edge Collapse“ [HDD⁺93, RR96] bzw. „Half Edge Collapse“ [Cam98] durchgeführt, bei welchem die Kante, die diesen Knoten mit einem seiner Nachbarn verbindet, auf die Länge Null kollabiert und die beiden Vertexpositionen somit zusammenfallen, so dass der verschobene Vertex entfernt werden kann (siehe auch Abb. 3.11). Näher soll an dieser Stelle nicht auf das Verfahren eingegangen werden, da zu diesem Thema ausreichend weiterführende Literatur gegeben ist [Hop98a, SSGH01].

Die konkrete Anwendung dieses Verfahrens in der Fahrzeugvisualisierung wandelt zunächst alle vierecksbehafteten Bauteilnetze in Dreiecksnetze um. Für jedes der entstehenden Dreiecksnetze

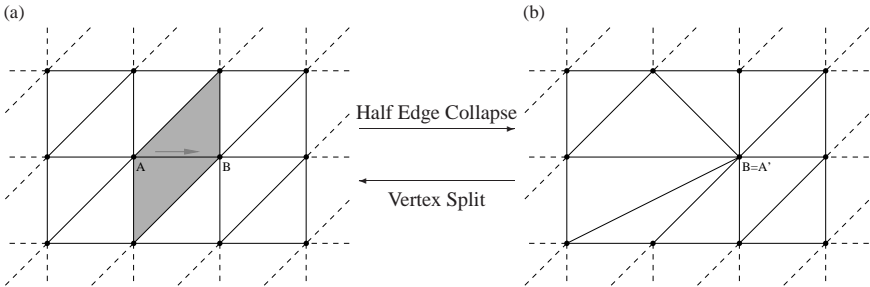


Abbildung 3.11: Half Edge Collapse Operation: (a) Netz vor dem Kollabieren der Kante A-B; (b) Nach dem Kollabieren fällt A' auf B und kann entfernt werden, ebenso fallen die in Abb. 3.11(a) grau markierten Dreiecke weg. Mit einer Vertex Split Operation kann ein Edge Collapse rückgängig gemacht werden und das nächsthöher aufgelöste Netz gebildet werden.

wird die Abfolge der vereinfachenden Kollabierungsschritte ermittelt und abgespeichert. Aufgrund eines bestimmten Qualitätskriteriums kann dann jedes Bauteil bis zu einem bestimmten Detaillierungsgrad anhand der abgespeicherten Kollabierungsliste mittels Vertex Splits (siehe Abb. 3.11) wiederhergestellt werden. Ein sinnvolles Qualitätsmaß in diesem Zusammenhang wäre z.B. die Entfernung des Bauteilschwerpunkts zum Bildmittelpunkt und zur Betrachterposition. Somit werden zentrale, nahe Komponenten in der Originalauflösung dargestellt und am Rand oder im Hintergrund gelegene Objekte lediglich grob rekonstruiert. Bei diesem Ansatz ergeben sich jedoch Probleme mit relativ langgezogenen Bauteilen, wie Seitenteilen oder dem Bodenblech, da bei diesen ggf. unterschiedliche Auflösungsstufen für ein und dasselbe Bauteil gewählt werden sollten oder evtl. auf einen zu feinen Detailgrad zurückgegriffen werden muss. Allerdings gibt es auch hierfür bereits Lösungswege ([Hop97] sowie [Hop98b]), welche die Rekonstruktion eines „Progressive Meshes“ mit lokal unterschiedlichem Detailgrad erlauben.

Unterteilungsflächen

Als Alternative zu den Progressive Meshes wurde im Rahmen einer Diplomarbeit [Kad00] zusätzlich ein Ansatz mit Unterteilungsflächen untersucht. Unterteilungsflächen gehen zunächst von einem groben Kontrollnetz aus und konvergieren bei zunehmender Verfeinerung gegen eine glatte Zielgeometrie. Bei der Verfeinerung wird jede Gitterzelle des gröbereren Netzes in mehrere kleine Zellen unter Anwendung bestimmter Regeln unterteilt, daher auch der Name des Verfahrens. Diese Unterteilung kann approximierend oder interpolierend erfolgen. Bei ersterem enthält die nächstfeinere Auflösungsstufe nicht mehr die Ausgangsvertices, und die Zielgeometrie entfernt sich somit zunehmend vom Ausgangsnetz. Interpolierende Verfahren hingegen beinhalten stets die Gitterknoten der gröbereren Stufe und daher immer die Vertices des Kontrollnetzes. Im

Hinblick auf die Steuerbarkeit der verfeinerten Grenzfläche ist für die Visualisierung von Fahrzeuggeometrien das interpolierende Unterteilungsschema dem approximierenden vorzuziehen.

Da Crashnetze zwar vierecksdominant sind, aber nicht ausschließlich aus Vierecken bestehen – und um flexibler bei der Generierung von Zwischenstufen zu bleiben – wurden die Untersuchungen auf dreiecksbasierte Netze beschränkt, obwohl es mittlerweile Unterteilungsverfahren gibt, die vierecksbasierte Netze unterstützen [VZ01]. Jüngste Forschungsergebnisse haben gezeigt, dass auch eine Mischung zwischen dreiecks- und vierecksbasierten Unterteilungsflächen möglich ist [PS04], welche jedoch bereits im groben Kontrollnetz verankert werden muss, so dass dieser Ansatz sich nur für Netze eignet, bei denen zusammenhängende Cluster von Dreiecken bzw. Vierecken auftreten, welches bei typischen FE-Modellen allerdings nicht der Fall ist.

Um auf einer FE-Oberfläche eine Unterteilungshierarchie aufbauen zu können, muss zunächst aus dieser Fläche ein geeignetes Basisnetz extrahiert werden. Eine wenig geeignete Basis bildet hierbei das grobe Netz, welches mittels des „Progressive Meshes“-Ansatzes gewonnen wird. Aufwändigere Verfahren berücksichtigen die speziellen Konnektivitätsanforderungen der Unterteilungsschemata [Hor02, HD04] und liefern daher bessere Ergebnisse.

Ausgehend von dem gewonnenen Kontrollnetz kann mittels eines interpolierenden Unterteilungsverfahrens das Originalnetz sukzessive angenähert werden. Eines dieser interpolierenden Schemata ist die Butterfly-Unterteilung [DLG90], welche den Namen aufgrund der Anordnung der Nachbarknoten trägt, die zur Positionsbestimmung eines neuen Unterteilungsknotens herangezogen werden. Abbildung 3.12 zeigt, dass zur Berechnung der Position des im Verfeinerungsschritt hinzugekommenen, in Bildmitte weiß markierten Vertex die Koordinaten der umliegenden – anhand einer schmetterlingsförmigen Maske ausgewählten – Gitterpunkte verwendet werden. Im ursprünglichen Ansatz fließen diese mit folgenden Gewichtungen in die neue Knotenkoordinaten ein:

$$a : \frac{1}{2} \quad b : \frac{1}{8} + 2w \quad c : -\frac{1}{16} - w \quad (3.6)$$

Mittels des zusätzlichen – ggf. lokal unterschiedlichen – Parameters w kann dabei die Entfernung des feinen Netzes von der Kontrollfläche und damit die resultierende Form geregelt werden. Bei der erweiterten Version der Butterfly-Flächen werden zusätzlich die beiden Punkte d berücksichtigt und jeweils mit w gewichtet, die Gewichtung der Punkte a wird dafür auf $\frac{1}{2} - w$ gesenkt.

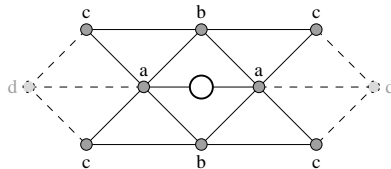


Abbildung 3.12: Butterfly Unterteilungsschema: Die ursprüngliche Unterteilungsmaske ist wie ein Schmetterling geformt.

Die hierarchische Rekonstruktion eines Bauteils mittels Unterteilungsflächen kann entweder statisch oder adaptiv erfolgen. Eine statische Unterteilung erfolgt für das gesamte Netz und zerlegt im Fall des Butterfly-Ansatzes jedes Dreieck der vorhergehenden Verfeinerungsstufe in vier neue Dreiecke, indem die Dreieckskanten, wie in Abb. 3.13 gezeigt, jeweils halbiert werden. Die Vor-

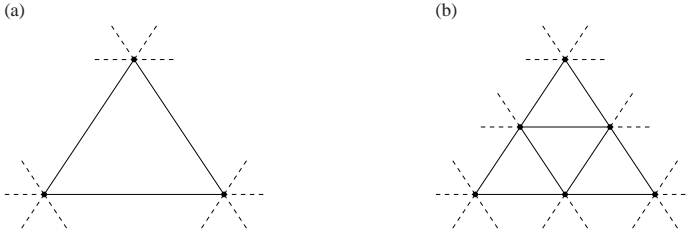


Abbildung 3.13: Gleichmäßige, dyadische Unterteilung eines Dreiecks: (a) grobe Unterteilungsstufe; (b) nächstfeinere Auflösung mit halbiertem Kantenlänge und vierfacher Anzahl an Dreiecken.

teile der statischen Unterteilung sind ihr einfacher sowie schneller Aufbau, und dass sämtliche Strukturen in einem Vorverarbeitungsschritt berechnet werden können. Ähnlich den Progressive Meshes kann dann anhand eines Qualitätskriteriums die entsprechende Auflösungsstufe gewählt und dargestellt werden. Da mit jeder Netzverfeinerung die Anzahl der Dreiecke um den Faktor vier zunimmt, stößt man jedoch sehr schnell an die Grenzen der Interaktivität.

Eine adaptive Unterteilungsstrategie hingegen initiiert blickpunktabhängig – nur in Bereichen, wo nötig – eine Verfeinerung der Dreiecke. Die Entscheidung, ob in einem Bereich eine höhere Unterteilungstiefe sinnvoll ist, kann ebenfalls mittels eines Qualitätskriteriums gefällt werden, welches die Verbesserung der Bildqualität, die durch einen weiteren Unterteilungsschritt eintritt, abschätzt. Hier bietet sich an, die maximale lokale Distanz zwischen gröberem und nächstfeinerem Netz auszurechnen, da diese Information aus der lokal gegebenen Unterteilungskonnektivität, wie in Abb. 3.14 angedeutet, leicht zu ermitteln ist. Projiziert man den Distanzvektor in den Bildraum, so lässt sich damit eine Aussage über den maximalen Pixelfehler treffen. Allerdings

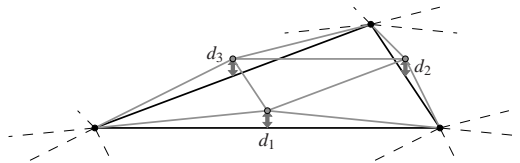


Abbildung 3.14: Maximale lokale Distanz $d = \max(d_1, d_2, d_3)$ zwischen Netzen zweier Unterteilungstiefen.

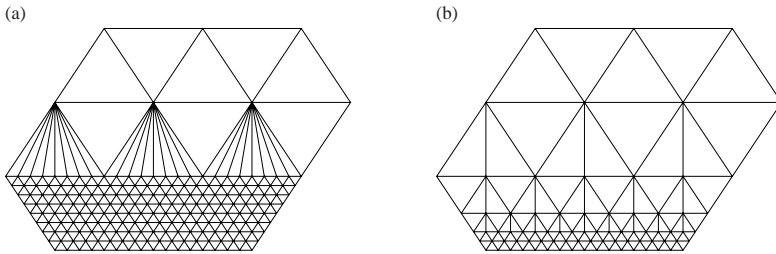


Abbildung 3.15: Saubere Übergänge zwischen unterschiedlichen Unterteilungsstufen: (a) Bei un stetigen Sprüngen in der Unterteilungstiefe müssen deformierte, langgezogene Dreiecke eingefügt werden, um die Entstehung von Spalten bzw. T-Vertizes zu verhindern; (b) Verläuft der Wechsel zwischen unterschiedlichen Unterteilungsstufen stetig, so entsteht ein deutlich ausgeglicheneres und besser strukturiertes Netz.

sind adaptive Unterteilungsnetze aufwändiger zu handhaben, da aufgrund mehrerer Nebenbedingungen im Vergleich zur statischen Unterteilung komplexere Datenstrukturen aufgebaut werden müssen. So sollte die Unterteilungstiefe benachbarter Gebiete um nicht mehr als eine Hierarchiestufe abweichen, um bei zunehmender Verfeinerung ein gut strukturiertes und balanciertes Netz beizubehalten (siehe Abb. 3.15). Dies lässt sich erreichen, indem man während der Unterteilung zwischen der regulären „red“-Aufteilung eines Dreiecks in 4 kleinere Dreiecke (Abb. 3.13) und der „green“-Aufteilung eines Dreiecks in zwei (Abb. 3.16(a)) oder drei (Abb. 3.16(b)) Dreiecke – um die Bildung von Spalten an der Stoßkante zwischen zwei unterschiedlichen Verfeinerungstiefen zu vermeiden – unterscheidet. Verbieta man das mehrmalige Anwenden eines „green“-Schritts auf dasselbe Dreieck und fordert stattdessen zuerst die Auflösung in einer „red“-Unterteilung, so lassen sich unformige Dreiecke, wie in Abb. 3.15(a) zu sehen, vermeiden [BSW83].

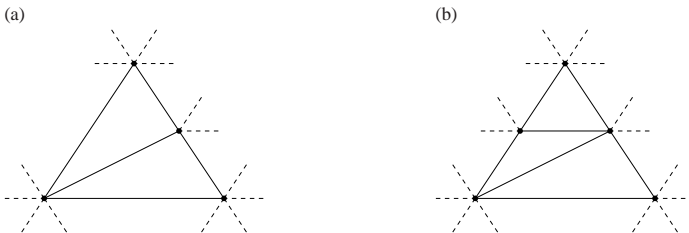


Abbildung 3.16: Ungleichmäßige „green“-Unterteilung eines Dreiecks [BSW83].

In der betreuten Diplomarbeit [Kad00] wurden Algorithmen entwickelt, welche es erlauben, in Echtzeit eine adaptive Unterteilung von beliebigen 2-mannigfaltigen Dreiecksnetzen adaptiv durchzuführen. Dabei wurde unter Anwendung der „red-green“-Strategie auf stetige Übergänge zwischen den unterschiedlichen Unterteilungsaufösungen und eine besonders speicherfreundliche Programmierung geachtet. Das Einsparen von Speicherplatz ist bei der Visualisierung größerer Datensätze sinnvoll. Es bedeutet aber, dass sämtliche Unterteilungsschritte erst zu dem Zeitpunkt durchgeführt werden können, an dem sie aufgrund einer Verletzung des oben genannten Qualitätskriteriums erforderlich werden. Notwendige Unterteilungsschritte werden also on-the-fly nur auf Basis der in der Hierarchie jeweils direkt darüberliegenden Auflösungsstufe durchgeführt. Um eine möglichst hohe Effizienz zu erreichen, werden Zwischenergebnisse, die sich in der nächsttieferen Unterteilungsebene wieder verwerten lassen, zusätzlich abgespeichert. Der Kosten/Nutzen-Faktor – zusätzlicher Speicherverbrauch im Verhältnis zu Performanzgewinn – rechtfertigt den relativ zur Datensatzgröße geringen zusätzlichen Speicherbedarf. So können auch die Koordinaten der Knoten in der nächsttieferen Unterteilungsebene günstiger vorberechnet werden, da nach Durchführung der Unterteilungsschritte meist mehrere Dreiecke in anderen Auflösungsstufen entstehen, welche die Zuordnung der mittels der Butterfly-Maske gesuchten Knoten (Abb. 3.12) für eine bestimmte Auflösungstiefe erschweren.

Bauteilränder erfordern eine gesonderte Behandlung, da hier ggf. Knoten für das Füllen der Butterfly-Maske fehlen. Bei konventionellen Ansätzen wird dieses Problem durch eine abgewandelte Zuordnungsmaske gelöst, welche jedoch unter Umständen der entstehenden Grenzfläche eine andere Form gibt, wie aufgrund der Standard-Maske erwartet. Diese Inkonsistenz in Randnähe erschwert die geeignete Wahl der Gewichtungsfaktoren w zur Steuerung der Grenzgeometrie, so dass es nur umständlich gelingt diese der originalen Bauteiloberfläche anzunähern. Daher wurde in [Kad00] bzw. [RKE01] vorgeschlagen, virtuelle Knoten außerhalb des Bauteils einzufügen, welche nicht visualisiert werden, es jedoch erlauben, den Bauteilrand wie ein inneres Gebiet zu behandeln. Diese virtuellen Knoten werden abhängig von der Unterteilungstiefe bei Bedarf ebenfalls zur Laufzeit generiert.

Bei der Repräsentation scharfer Kanten gilt Ähnliches. Auch hier kommen in der Regel modifizierte Unterteilungsmasken zur Anwendung. Jedoch ist bis zu einem gewissen Grad auch eine Steuerbarkeit mittels entsprechend gewählter Gewichtungsfaktoren möglich, eine absolut scharfkantige Grenzfläche lässt sich hiermit allerdings nicht realisieren. Aufgrund der Eigenschaft, glatte Flächen zu generieren, eignet sich eine Unterteilungsrepräsentation einer Oberfläche besonders für Komponenten wie Karosserieverkleidungen. Dies kommt vor allem der Tatsache entgegen, dass seitens des CAD beim Design dieser Komponenten vermehrt Unterteilungsflächen zum Einsatz kommen. Für die Darstellung von Geometrien im Fahrzeuginneren eignen sich diese Flächentypen allerdings in der Regel wenig, da es sich hierbei meist um scharfkantige, strukturttragende Konstruktionen handelt, bei denen das Hauptaugenmerk nicht auf der Glattheit der Oberfläche liegt.

In Hinblick auf die Visualisierung von Crashmodellen ergibt sich aus diesen Randbedingungen somit die Tatsache, dass Unterteilungsflächen nur bedingt für die Repräsentation von strukturellen FE-Bauteilen geeignet ist. Auch blieb die ursprünglich erhoffte Performanz des Echtzeit-Unterteilungsansatzes trotz vieler Optimierungen etwas hinter den Erwartungen zurück. Ein ge-

eignetes Qualitätskriterium für die Abschätzung des gewünschten Auflösungsvermögens ist – ähnlich wie bei den Progressive Meshes – unter Umständen nicht trivial und es ist zudem fraglich, ob sich bei sehr kompakten Objekten, um die es sich bei Automobilen handelt, überhaupt eine starke Abstufung des Detaillierungsgrades erzielen lässt, so dass sich daraus ein größerer Performanzgewinn ergeben kann.

Dennoch ist der Ansatz der adaptiven Unterteilungsflächen für die Zukunft attraktiv, da mit wachsender Rechenleistung auch die ermöglichte Modellkomplexität steigt und zunehmend auch nichttragende Verkleidungskomponenten simuliert werden können. Da diese – wie erwähnt – zunehmend als Unterteilungsflächen definiert werden, ist eine entsprechend native Visualisierungsmöglichkeit für solche Bauteile weiterhin reizvoll. Auch in weiterführenden Untersuchungen – wie virtuellen Fahrsimulationen – können solche Methoden zum Einsatz kommen um z.B. eine realistische und ansprechende Umgebung wiederzugeben [RKE01].

Ein weiteres Manko aller blickpunktabhängigen Detaillierungsverfahren und auch anderer geometriebasierter Vereinfachungsverfahren ist allerdings zum einen die Tatsache, dass sie meist nur auf dreiecksbasierten Netzen akzeptable Ergebnisse liefern und zum anderen die Umrisse der einzelnen finiten Elemente nicht mehr sichtbar sind. Daher wurden neue, speziell für FE-Modelle zugeschnittene Algorithmen entwickelt, welche die in diesem Abschnitt formulierten Nachteile beheben und im Folgenden vorgestellt werden sollen.

3.2.3 Optimierte Simplifizierungsverfahren

Viele Simplifizierungsalgorithmen sind so konzipiert, dass sie möglichst universell eingesetzt werden können, und sie basieren in der Regel auf der Annahme, dass relativ kleine Details in einem Modell vernachlässigbar sind. Diese Hypothese ist für die meisten Modelle – insbesondere natürlicher Art – wie z.B. die Hautoberfläche menschlicher Charaktere gültig. Bei technischen Simulationsbauteilen hingegen können auch kleinste Details einen gravierenden Einfluss auf das Ergebnis haben. Daher sollte auch die visuelle Wiedergabe einer FE-Oberfläche zu jeder Zeit alle ihre Merkmale beibehalten. Zu diesem Zweck wurde ein Simplifizierungsverfahren entwickelt, welches speziell auf die Anforderung von FE-Geometrien zugeschnitten ist und im Gegensatz zu gängigen Verfahren auch sehr kleine Details beibehält.

Üblicherweise verwenden Simplifizierungsverfahren geeignete Fehlermetriken, um zu beurteilen, welche Bereiche einer Oberfläche durch eine gröber aufgelöste Struktur ersetzt werden können, ohne dass eine stark wahrnehmbare Veränderung eintritt. Die Bewertung eines Oberflächennetzes anhand solch einer Fehlermetrik kann dabei entweder – wie teilweise im vorherigen Abschnitt angewendet – blickpunktabhängig zur Laufzeit erfolgen oder statisch in einem Vorverarbeitungsschritt. Um möglichst maximale Performanz bei gleichzeitig geringem Speicherverbrauch und bester Qualität zu erreichen, erscheint eine statische Simplifizierung, welche die Knotenkoordinaten des Ursprungsnetzes wieder verwendet, am geeignetsten.

Auf dieser Basis operieren die drei in [GH97] vorgestellten und sehr häufig verwendeten Ansätze zur Simplifikation bzw. Reduktion von Oberflächennetzen. Der erste, „Vertex Clustering“ [RB93], verbindet Vertices, welche die gleiche Zelle eines uniformen, volumetrischen Git-

ters teilen zu einem gemeinsamen Vertex in diesem Cluster. Dieser Algorithmus ist sehr schnell, hat jedoch den Nachteil, dass Netze von sehr geringer Qualität entstehen und zudem die Topologie eines Netzes – z.B. durch Schließen von Löchern – drastisch verändert werden kann. Dieser Ansatz wurde unter Berücksichtigung der geforderten Ansprüche an die Darstellungsqualität von FE-Netzen nicht weiter verfolgt.

Der zweite Ansatz ist noch bekannt aus dem letzten Abschnitt und wird meist „Edge Collapsing“ [HDD⁺93, RR96], seltener auch „Edge Contraction“ betitelt. Er benutzt eine lokale Fehlerabschätzung, z.B. den ein- [Cam98, Had98] oder zweiseitigen [KLS96] Hausdorff-Abstand, um zu entscheiden, welche Kante als nächste in dem iterativen Prozess zu entfernen ist. Abhängig vom Fehlerkriterium liefert dieser Ansatz gute Ergebnisse mit vertretbarem Zeitaufwand und wird daher auch sehr oft eingesetzt. Allerdings wurde diese Methode – wie bereits diskutiert – allgemeingültig für beliebige Netze entworfen und ist daher nicht auf die Besonderheiten von FE-Netzen spezialisiert. Ohne ein geeignet gewähltes Dezimierungskriterium kann diese Methode somit unter Umständen sogar deutlich sichtbare und für die Struktursimulation ggf. sehr bedeutende Kanten entfernen.

Das dritte Verfahren namens „Vertex Decimation“ [SZL92] benutzt ebenfalls eine lokale Fehlerabschätzung, um die für eine verlustarme Dezimierung am besten geeigneten Vertices zu bestimmen. Diese Methode operiert im Gegensatz zum „Edge Collapsing“ nicht auf Kanten, sondern den Knoten selbst. Die durch die Entfernung des jeweiligen Knotens betroffenen Dreiecke in der direkten Umgebung werden gelöscht und das entstehende Loch wird komplett neu vernetzt. Der im Folgenden präsentierte Ansatz baut auf diesem Gedanken auf und erweitert ihn derart, dass nicht nur sequentiell einzelne Knoten entfernt werden, sondern zusammenhängende Gebiete, welche durch relevante Merkmalslinien begrenzt werden, geschlossen analysiert und ggf. mehrere Knoten in einem Schritt entfernt werden. Die anschließende Neuvernetzung kann somit entsprechend großflächiger und optimiert erfolgen.

Merkmalerhaltende Simplifizierung

Verschiedene Autoren haben sich bereits mit speziellen Simplifizierungsverfahren befasst, welche besondere Eigenschaften und Merkmale einer Oberfläche möglichst gut erhalten. Für Fahrzeugteile kann es unter Umständen wichtig sein, Krümmungen gut auf ein vereinfachtes Netz zu übertragen. In [Tur92] werden daher in stark gekrümmten Bauteilbereichen zusätzliche Punkte im Polygonnetz erzeugt und deren Umgebung entsprechend neu trianguliert („Re-tiling“). Dies bewirkt bei der anschließenden Simplifizierung durch Entfernen von geeignet erscheinenden Knoten eine bessere Auflösung des Netzes in diesen Krümmungsgebieten. Allerdings ist es zum einen fraglich, wie auf einem FE-Netz zusätzliche Knoten korrekt erzeugt werden, ohne dass das Ergebnis falsche Gegebenheiten widerspiegelt. Zum anderen werden durch dieses Vorgehen unvermeidlich neue Vertices eingeführt, was zu einem erhöhten Speicherverbrauch führt.

Andere Ansätze versuchen z.B. Farbverläufe oder die korrekte Texturierung auf einer Geometrie zu erhalten [GH98]. Zu diesem Zweck werden die betreffenden Parameter in der Regel in das Fehlermaß mit aufgenommen. Dadurch kann das Berechnen und Bewerten des Fehlerkriteriums

unter Umständen jedoch recht aufwändig und zeitraubend werden. Mit solch einem Ansatz ließen sich zwar scharfe Kanten als wichtiges Attribut markieren, jedoch werden unter Umständen auch hier schwächer ausgeprägte Merkmale vernachlässigt.

Den meisten Methoden liegt entweder eine stark lokal begrenzte oder eine netzweit operierende Simplifizierungsstrategie zu Grunde. Beide Strategien haben Nachteile: Sie sind entweder ungenau und liefern inakzeptable Ergebnisse oder der Aufwand steigt mit zunehmender Modellgröße überproportional an. Um gute Ergebnisse mit vertretbarem Aufwand zu erzielen, bietet es sich an, das Modell an den Stellen in kleinere Bereiche aufzuteilen, bei denen keine oder nur wenig Möglichkeiten zur Simplifizierung gegeben sind. Im Fall von FE-Netzen sind dies die Merkmalslinien eines Bauteils, d.h. scharfe Kanten, da diese zusammen mit den Außenkanten das Aussehen der Bauteilsilhouette prägen und deshalb mit hoher Genauigkeit wiedergegeben werden sollten. Algorithmen zur Detektion solcher für die FE-Simulation wesentlichen Merkmale wurden bereits in Kapitel 3.1.2 mit einem besonderen Augenmerk auf die Robustheit der Verfahren vorgestellt. Mit diesen Methoden lässt sich ein Bauteil in klar abgegrenzte Bereiche einteilen, wie in Abb. 3.17(b) zu sehen. Die stabile Fortsetzung von schwach ausgeprägten Kanten garantiert hierbei eine fehlerarme Unterteilung der Bauteile in zusammenhängende, merkmalsdefinierte Bereiche auch in nahezu ebenen Regionen (siehe kurze weiße Linie in Abb. 3.17(b) bzw. 3.17(c) an der linken Seite der Ausbuchtung).

Eine ähnliche Vorgehensweise, basierend auf der Idee, Gebiete zur genaueren Analyse aufgrund von Oberflächen-Eigenschaften zusammenzugruppieren, wurde auch bereits von anderen Autoren vorgeschlagen [IYY⁺01, She02], jedoch basieren deren Ansätze auf CAD-Daten, welche – im Vergleich zu FE-Modellen – zahlreichere und hochauflösendere Informationen enthalten, so dass eine genaue Analyse der Bauteile deutlich einfacher ausfällt als im Folgenden diskutiert. Im Allgemeinen kann jedoch eine CAD-Geometrie – sofern diese in frühen Designphasen vorhanden sein sollte – nicht direkt auf ein FE-Gitter übertragen werden, so dass eine Verwertung dieser zusätzlichen Datenbasis nicht praktikabel ist.

Durch die vorgeschlagene Aufteilung entlang der Merkmalslinien entstehen zusammenhängende Gebiete, welche einfacher und schneller auf ihre Komplexität hin analysiert werden können. Da es sich bei Fahrzeugkomponenten in der Regel um „technisch“ geformte Bauteile handelt, treten innerhalb der einzelnen Gebiete leichte Krümmungen in meist nur einer Dimension auf. Teile, welche sich an der Fahrzeugverkleidung finden, sind meist komplexer geformt und ggf. in zwei Richtungen gekrümmt und verwunden. Letztere Komponenten nehmen in Crashesimulationen jedoch nicht den Stellenwert wie in Designbeurteilungen ein und müssen daher bei einer simplifizierten Darstellung auch nicht optimal angenähert werden. Erstere Bauteilart hingegen trägt zur strukturellen Stabilität eines Fahrzeugs bei, und eine möglichst exakte Annäherung in der reduzierten Darstellung ist daher notwendig. Diese Bauteile lassen sich jedoch aufgrund ihrer meist eindimensional und nur schwach ausgeprägten Krümmung sehr gut durch ebene Flächenstücke approximieren.

Geeignete Regressionsebenen durch die Knoten eines Gebietes lassen sich mittels der Methode der kleinsten Quadrate [BSMM97] finden. Führt man als Nebenbedingung einen maximal erlaubten Abstand d_{\max} der Vertizes von solch einer Regressionsebene ein, so bilden sich im Allgemeinen mehrere Knoten-Cluster, die jeweils approximativ eine Ebene definieren. Somit defi-

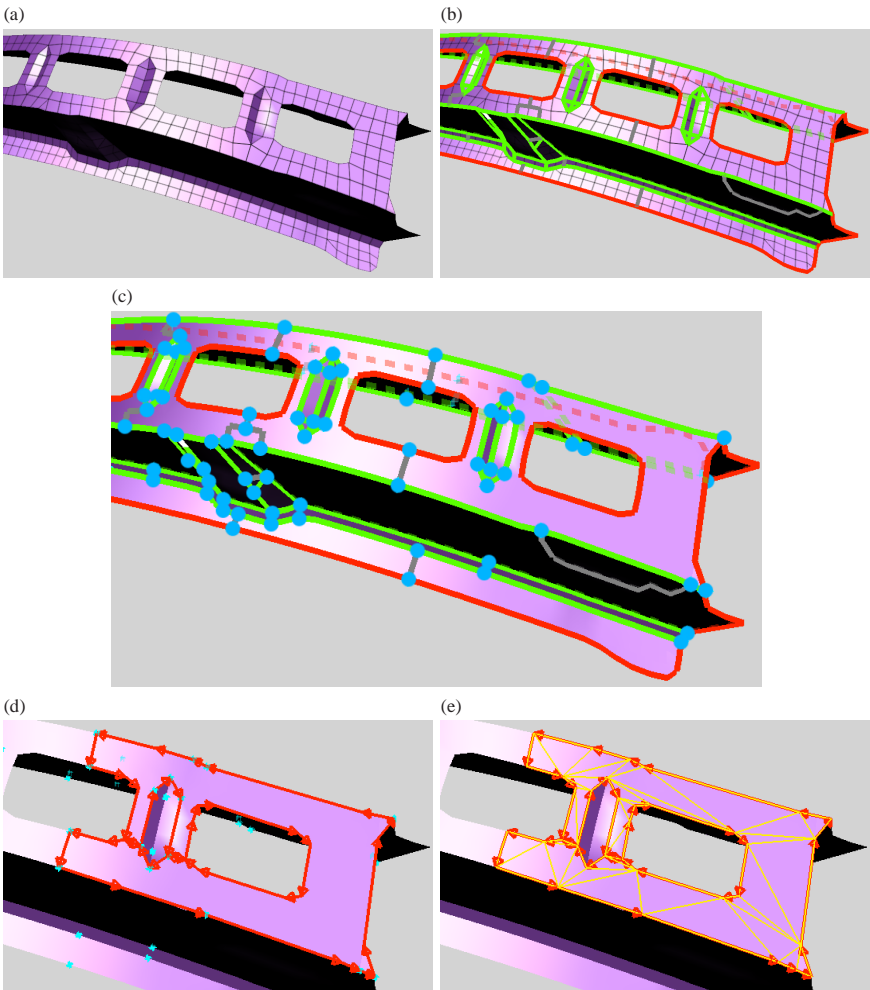


Abbildung 3.17: Simplifizierungsprozess: (a) Hochauflöstes Originalnetz; (b) Gebietsgrenzen: Außenkante (rot), scharfe Innenkante (grün), vervollständigte auslaufende Kante (weiß), Untergebiete- / Ebenengrenze (grau); (c) arretierte Kopplungspunkte (blau); (d) simplifizierte und orientierte Begrenzung eines Analysegebietes mit Löchern; (e) endgültige Triangulierung des vereinfachten Netzes.

nieren die einzelnen Ebenen Untergebiete, und die Nahtstellen zwischen angrenzenden Ebenen können als weitere Merkmalsgrenze interpretiert werden, welche in Abb. 3.17(b) bzw. 3.17(c) grau markiert sind. Da alle entsprechend gruppierten Knoten nahezu in derselben Ebene liegen, kann innerhalb eines solchen Untergebietetes eine sehr einfache Parametrisierung mittels einer Projektion in die jeweilige Ebene erfolgen. Diese einfache und direkte Parametrisierung ist besonders im nachfolgenden Unterkapitel von Bedeutung.

Ein weiterer Vorteil der Aufteilung in planare Untergebiete ist die Garantie für die Einhaltung einer maximalen Abweichung bzw. Entfernung der einzelnen finiten Elemente von der Ebene, der sie jeweils zugeordnet sind. Mittels d_{\max} lässt sich also direkter Einfluss auf die Qualität der Approximation nehmen.

Nach der Unterteilung eines Bauteils kann die eigentliche Simplifizierung bzw. Polygonreduktion für jedes Untergebiet getrennt erfolgen. Innerhalb eines planaren Untergebets können sämtliche Knoten entfernt werden, da für diese bereits die Einhaltung der Fehlerschranke d_{\max} explizit erfüllt wird. Übrig bleiben somit die umrandenden Merkmalslinien. Diese sollten mit einer geringeren Fehlertoleranz simplifiziert werden, da sie – wie angesprochen – je nach Blickwinkel zur Bauteilsilhouette gehören können. Außerdem wird damit garantiert, dass in sämtlichen angrenzenden Ebenen die Abweichung nicht über d_{\max} hinauswächst. Punkte, an denen sich zwei oder mehr Merkmalslinien treffen, dürfen zudem nicht verschoben oder entfernt werden, da dies das Aussehen der Oberfläche gravierend verändern würde. Durch einfaches Abzählen der ausgehenden Merkmalskanten lassen sich diese Punkte, wie in Abb. 3.17(c) hellblau markiert, als nicht veränderbar deklarieren.

Abhängig von der Art der Merkmalslinie kann man unterschiedliche Toleranzkriterien anwenden. Veränderungen an Außenkanten sind am ehesten erkennbar und daher sollten für eine Reduktion dieser Linienzüge besonders strenge Qualitätsmaße gelten. Ebenso sollen Knoten auf scharfen Kanten nur dann entfernbar sein, wenn sie nahezu auf einer Linie mit ihren Nachbarknoten auf der Kantenlinie liegt. Bei nur schwach ausgeprägten Kanten hingegen liefern auch größere Annäherungen akzeptable Ergebnisse. Anhand des Kantenwinkels α kann man mittels des Wichtungsterms $\cos^2 \alpha$ aus dem an scharfen Kanten bzw. dem an schwachen Kanten angestrebten Maximalfehler den lokal entsprechenden Toleranzwert nahtlos ableiten. Für ein planares Untergebiet ist in Abb. 3.17(d) die aufgrund dieser Regeln resultierende, simplifizierte Berandung eingezeichnet.

Diese reduzierte Berandung beschreibt die simplifizierte Version des umschlossenen Untergebets und muss zu Darstellungszwecken trianguliert werden. Das in Abb. 3.17(d) (und abstrahiert in Abb. 3.18(a)) gezeigte umrandete Gebiet ist zugleich auch ein Beispiel für eine nicht-triviale Berandung, da das Gebiet zusätzlich zwei Löcher enthält. Ein Loch wird dabei durch die entsprechende Bauteilberandung vorgegeben, das andere Loch ergibt sich aufgrund der merkmalsbedingt getrennt behandelten Betrachtung der kleinen Sicke links daneben. Diese komplexe Polygonberandung lässt sich jedoch leicht in einen simplen Polygonzug überführen, indem man möglichst kurze Stege von Punkten auf der äußeren Umrandung zu den Löchern bildet und entlang dieser Stege das Polygon wie in Abb. 3.18(b) skizziert aufschneidet. Das resultierende simple Polygon ist frei von Löchern und hat den zusätzlichen Vorteil, dass seine Berandung eindeutig orientierbar ist.

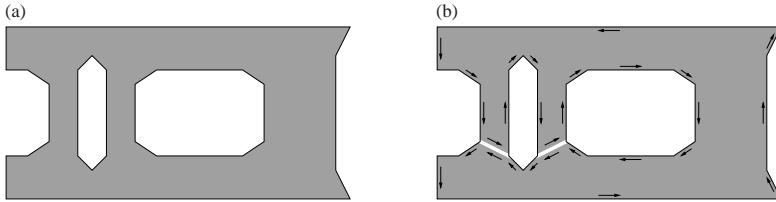


Abbildung 3.18: Aufspaltung eines komplexen Polygons: (a) komplexes Ausgangspolygon mit zwei Löchern; (b) Auftrennung entlang von Stegen zwischen Außenrand und Lochumrandungen liefert ein simples Polygon und erlaubt eine eindeutige Orientierung der Begrenzungslinie.

Mittels einer für konkave Vielecke geeigneten Triangulierungsroutine kann das Polygon abschließend trianguliert werden. In diesem Fall kommt einer der ältesten aber zugleich auch häufigsten im Einsatz befindlichen Algorithmen zum Einsatz, die Ear-Clipping-Methode [Mei75]. Hierbei werden aus dem Polygon hervorstehende, dreieckige „Ohren“ sequentiell abgeschnitten, bis nur noch ein einziges Dreieck verbleibt. Letztendlich erhält man dadurch eine Menge an abgeschnittenen Dreiecken, welche eine korrekte Tessierung des Polygons darstellen. Der Algorithmus wurde derart erweitert, dass möglichst sowohl nahe der ursprünglichen Oberfläche liegende⁷ als auch gleichseitig geformte „Ohren“ bevorzugt werden, so dass in den meisten Fällen eine ausgewogene Triangulierung entsteht. Das exemplarische Ergebnis ist in Abb. 3.17(e) zu sehen. Die Polygonzahl gebräuchlicher Crashsimulations-Modelle lässt sich mit dem präsentierten Simplifizierungsansatz auf etwa 40 bis 50 Prozent reduzieren, ohne dass in der schattierten Darstellung ein merklicher Qualitätsverlust erkennbar wird.

Texturbasierte Detail-Erhaltung

Wird eine zunehmend stärkere Polygonreduktion angestrebt, so nimmt mit der Polygonanzahl auch die Darstellungsqualität entsprechend ab. Bei konventionellen Simplifizierungsansätzen lässt sich dies auf den steigenden Verlust von Details – vor allem in der Silhouette – und zum anderen auf eine zunehmend grobmaschigere Beleuchtungsberechnung zurückführen. Das erste Problem wird durch das im vorhergehenden Abschnitt erläuterte, auf FE-Geometrien spezialisierte, Simplifizierungsverfahren hinreichend gelöst.

Um dem Verlust einer guten Beleuchtungsqualität bei grober Simplifizierung vorzubeugen, schlagen verschiedene Autoren vor, die Normalen des hochaufgelösten Originals in einer Textur zu speichern. Diese Textur wird beim Zeichnen des reduzierten Modells herangezogen, um die Beleuchtungsberechnung durchzuführen [CMSR98, COM98, RE00] bzw. die Illuminationswerte in einer – für alle im Modell auftretenden Normalenrichtungen – vorberechneten Tabelle

⁷ermittelt über den einseitigen Hausdorff-Abstand [Cam98]

nachzuschlagen [TCRS00]. Mittels solch eines Ansatzes ist es zudem möglich, die einfache Beleuchtungsberechnung von OpenGL zu übertreffen und z.B. zusätzliche Effekte wie Spiegelungen oder „Environment Mapping“ zu verwirklichen [TCRS00].

Im Wesentlichen können die gleichen Algorithmen zum Einsatz kommen, wie sie in Kapitel 3.1.1 zur pixelgenauen Phong-Beleuchtung verwendet werden. Daher wurde das dort präsentierte Verfahren adaptiert, um die an den Knoten des unsimplifizierten Originalnetzes gesetzten Normalenvektoren in einer zweikomponentigen Textur zu speichern. Im Unterschied zur pixelexakten Beleuchtung auf einem nicht-reduzierten Gitter erfolgt die Zuordnung der Normalen nicht elementweise. Jedem Dreieck des simplifizierten Netzes werden die Elemente aus dem Originalnetz zugeordnet, aus denen das simplifizierte Dreieck hervorgegangen ist. Die Knotennormalen dieser Elemente werden entsprechend ihrer Position in dem planaren Unterbereich ihres Bauteilabschnitts auf das simplifizierte Primitiv abgebildet. Die Abbildungsparameter sind dabei innerhalb eines ebenen Untergebiets konstant, und die Bildung eines Textur-Atlanten ist aufgrund der einfachen Projektion in eine Ebene trivial. Dementsprechend werden alle Elemente eines Untergebiets einem gemeinsamen Texturabschnitt zugeordnet. Die Abtastdichte der Normalen richtet sich dabei nach den kürzesten, auf die beiden Texturkoordinatenachsen projizierten Kantenlänge aller zugehörigen Elemente. Dies bedeutet, alle Elementkanten werden, je nach ihrer Hauptausrichtung, entweder auf die u -Achse oder auf die v -Achse der Textur projiziert, und für jede der beiden Achsen wird die kürzeste projizierte Länge als Auflösungsmaßstab herangezogen, wie in Abb. 3.19(a) skizziert. Im Allgemeinen kann sich somit die anhand des Nyquist-Shannon Theorems ([Sha49]) daraus ergebende Abtastrate in den beiden Dimensionen unterscheiden. Soll eine ebenso hohe Qualität erreicht werden, wie bei der pixelgenauen Beleuchtung in Abb. 3.1(c), so sollte die Abtastauflösung verdoppelt werden (Abb. 3.19(b)) um zusätzlich zu den Knotennormalen an den Eckpunkten eines Elements die Elementnormalen in der Mitte der Elementkanten berücksichtigen zu können.

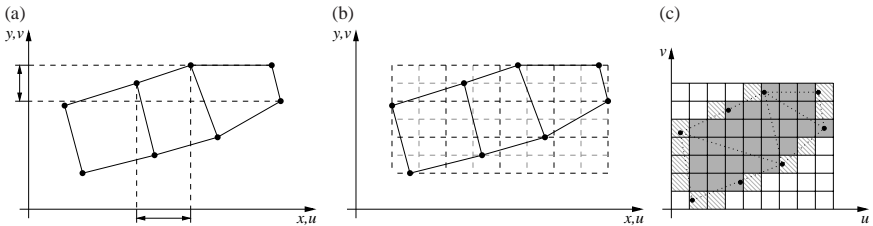


Abbildung 3.19: Geeignete Auflösung der Normalentextur: (a) Ermitteln der kürzesten projizierten Kanten; (b) Raster zur Abtastung der Knotennormalen mit in beiden Dimensionen doppelt hoher Auflösung um auch Normalen auf der Kantenmitte zu erfassen; (c) Resultierende Normalentextur (die zu initialisierenden Texel sind grau hinterlegt, der simplifizierte Dreiecksstreifen ist gepunktet auf die Textur ebene abgebildet).

Die zweikomponentige Normalentextur (Abb. 3.19(c)) lässt sich hardware-beschleunigt aufbauen, indem die finiten Elemente (Abb. 3.20(a)) in das planare Untergebiet bzw. die Texturebene projiziert werden (Abb. 3.20(b)) und mit entsprechenden Grün- bzw. Blaukanalwerten für die Normalenkomponenten an den Eckpunkten gezeichnet werden. Eine Normalisierung der Normalen innerhalb der Elementflächen kann aus den in Abschnitt 3.1.1 aufgeführten Gründen entfallen.

An der Berandung eines Normalentextur-Fleckens muss zusätzlich eine weitere Texelreihe initialisiert werden – in Abb. 3.19(c) schraffiert dargestellt – um eine korrekte bilineare Interpolation beim Extrahieren der Normalen aus der Textur zu garantieren. Dabei muss anhand des Typs des Begrenzungsmerkmals unterschiedlich vorgegangen werden. Handelt es sich um eine scharfe Bauteilinnenkante (siehe Abb. 3.20(b) links) oder um eine Bauteilaußenkante, so wird die am Rand gegebene Normaleninformation übernommen. Dies entspricht einer Dilatation-Operation auf das texturierte Gebiet [HE00]. Somit wird bei einer bilinearen Textur-Interpolation verhindert, dass falsche Informationen in den betroffenen Grenzbereich diffundieren, und dass dadurch die Normalen entlang scharfer und äußerer Kanten korrekt wiedergegeben werden. Im Falle einer nicht scharfen Innenkante (in Abb. 3.20(b) rechts) werden zusätzlich angrenzende Elemente aus den benachbarten Gebieten auf die momentane Texturebene projiziert und die Textur wird um deren Normaleninformationen erweitert. Die Projektion in die lokale Ebene ist zulässig und nur mit sehr kleinen Auflösungsfehlern behaftet, da es sich aufgrund der Tatsache, dass sich an diesen Stellen keine scharfen Kanten befinden, nur um sehr schwache Abweichungen in der Ausrichtung der beiden benachbarten Ebenen handeln kann.

Die einzelnen Normalentextur-Flecken eines Bauteils werden aus Performanz- und Speicherplatzgründen in einer gemeinsam genutzten großen Textur zusammengefasst. Beim Aufbau eines solchen bauteileigenen Textur-Atlanten sollte darauf geachtet werden, dass möglichst wenig Fläche der Textur ungenutzt bzw. frei bleibt. Mittels der Bounding Boxes der Textur-Flecken lassen sich ähnlich hohe Texturen gruppieren und hintereinander aufreihen [CMSR98, IC01, SSGH01, CH02], so dass zwischen untereinander stehenden Reihen möglichst wenig Platz verbleibt, wie das Beispiel in Abb. 3.21(a) zeigt. Diese Gruppierung ist sehr schnell und effizient, allerdings bleiben bei diesem Vorgehen die freien Texel am Rand der einzelnen Textur-Flecken ungenutzt. Ebenso werden etwaige Löcher nicht aufgefüllt. Durch eine Analyse der freibleibenden Texel lassen sich – ähnlich [RE00, LPRM02] – die Dimensionen von rechteckigen Freiflächen bestimmen, in denen kleinere Textur-Flecken untergebracht werden können. Durch diese zusätzliche Analyse verzögert sich zwar der Aufbau des Textur-Atlanten, im Endeffekt ergibt sich jedoch eine deutlich höhere Packdichte, wie in Abb. 3.21(b) zu sehen ist.

Texturbasierte Restauration der Berandungen finiter Elemente

Neben der Erhaltung der Beleuchtungsverhältnisse auf einer simplifizierten Oberfläche ist vor allem das zusätzliche Drahtgitter, d.h. die Berandungen der finiten Elemente, für die Anwender wichtig. Ein Zeichnen dieser Konturen in einem zweiten Durchgang (Abb. 2.13(a)) ist in Verbindung mit einem simplifizierten Modell nicht sinnvoll, da zum einen dadurch die, durch die Geometriereduktion gewonnene Darstellungsgeschwindigkeit zunichte gemacht wird, und zum

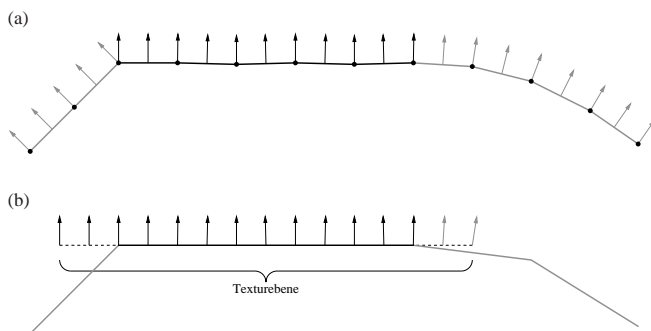


Abbildung 3.20: Initialisierung einer Normalentextur: (a) finite Elemente und zugehörige Normalen an Knoten und Elementmitten; (b) exemplarische Texturebene auf der reduzierten Oberfläche mit der übernommenen Normaleninformation; außerhalb der Ebene (gestrichelt dargestellt) werden je nach Übergang zur Nachbarschaft die Normalen von Nachbarelementen adaptiert (an nicht scharfen Kanten, siehe rechts) oder die entsprechende Randnormale kopiert (an scharfen Kanten oder Außenkanten, siehe links).

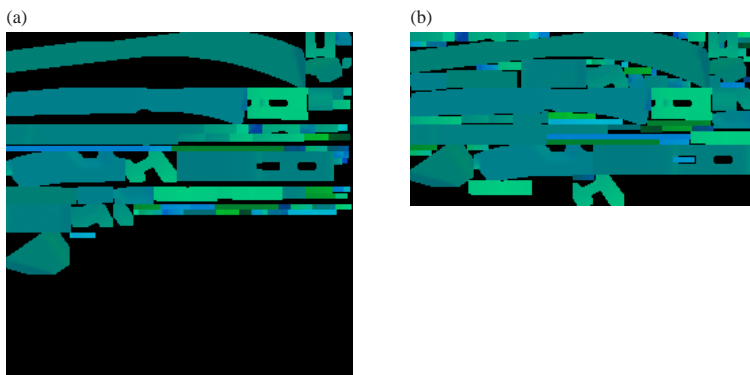


Abbildung 3.21: Normalentextur-Atlas des in Abb. 3.6(b) bzw. 3.17 gezeigten Bauteils: Im grünen und blauen Farbkanal sind die Normalenkomponenten n_x bzw. n_y gemäß Abschnitt 3.1.1 gespeichert. (a) Gruppierung anhand der vertikalen Ausdehnung der einzelnen Textur-Flecken, Gesamtgröße der Textur: 256×256 ; (b) zusätzliche Ausnutzung der Lücken und freibleibender Randflächen, Texturdimensionen: 256×128 .

anderen aufgrund des leicht veränderten Oberflächenverlaufs die Drahtgitterlinien u.U. die schattierten Bauteiloberflächen durchdringen und daher nicht immer auf dem Modell sichtbar sind.

Die in Abb. 2.13(b) bzw. [KHSE98] gezeigte Lösung kann in diesem Zusammenhang ebenfalls nicht zum Einsatz kommen, da bei einem simplifizierten Modell für jedes Dreieck im reduzierten Netz eine individuelle, hochauflösende ein-kanalige Luminanz-Textur mit den Konturlinien des entsprechenden Ausschnitts aus dem Originalnetz verwendet werden muss. Eine direkte Bereitstellung der Berandungslinien über Texturen ist jedoch für übliche Modellgrößen zu speicherintensiv. Möchte man z.B. eine ähnlich gute Qualität wie in Abb. 2.13(b) erreichen, so steht in einem 128MB großen Speicher einer Graphikkarte lediglich Platz für etwa 7500 Elemente bereit⁸.

Die Grundidee des hier bzw. in [RE03] vorgestellten Ansatzes beruht auf einer indirekten und dabei sehr kompakten Speicherung der geradlinigen Elementberandungen in einem Distanzfeld. Dieses Distanzfeld beschreibt den senkrechten Abstand der Abtastpunkte bzw. Texel zu den Kanten der finiten Elemente. Abbildung 3.22(a) erläutert, wie der Verlauf einer geradlinigen Elementkante in von vorgegebenen Abtastpunkten gemessenen Distanzen ausgedrückt werden kann. Theoretisch ist es auf diese Weise möglich, aus lediglich an drei Texeln gegebenen Distanzwerten die Position und Richtung der Elementkante zu ermitteln. Es erweist sich jedoch als praktikabler, mindestens vier Texel einzusetzen und zusätzlich die Distanzwerte auf einer Seite der Geraden mit einem negativen Vorzeichen zu behaften. Entsprechend dieser gerichteten Distanzen werden die Luminanzwerte der einzelnen Texel definiert (Abb. 3.22(b)). Auf diese Weise kann die Graphikhardware dazu verwendet werden, die Distanzwerte von jeweils vier benachbarten Texeln bilinear zu interpolieren und mittels einer davon abhängigen Nachschlageoperation oder unter Anwendung eines Fragmentprogramms die Pixel zu finden, an denen der Abstand zur Geraden – respektive der Luminanzwert des Fragments – Null beträgt. Diese Pixel geben dann

⁸hier wurde bereits der Platzbedarf eines doppelt gepufferten 1280×1024 großen Bildschirmspeichers berücksichtigt

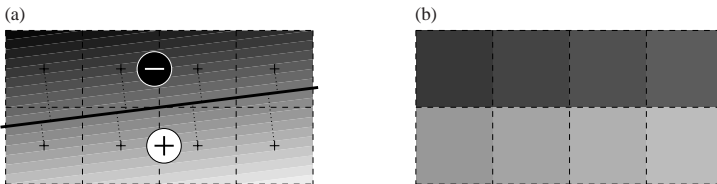


Abbildung 3.22: Eine Gerade kann indirekt in einer gerichteten Distanztextur gespeichert werden: (a) Berechnung der senkrechten Abstände von der jeweiligen Texelmittle zur Geraden; befinden sich die Texel in derselben, durch die Gerade definierte Halbebene wie das finite Element, so ist das Vorzeichen positiv, ansonsten negativ; (b) Die Distanztextur kodiert die in Texelmittle vorliegenden, gerichteten Abstände zur Geraden als Luminanzwert.

exakt den Verlauf der Geraden wieder. Je nach Betrachterentfernung ist die so gewonnene Linie jedoch sehr dünn oder lückenhaft. Daher empfiehlt es sich nicht auf Gleichheit mit Null zu testen, sondern alle Pixel mit der Umrandungsfarbe (in der Regel Schwarz) einzufärben, deren Absolutwert der Distanz unter einer positiven Schranke $\varepsilon \ll 1$ liegt. Für größere ε ergeben sich somit dickere Linien.

Auf die gleiche Art können die verbleibenden drei Kanten eines Vierecks (bzw. die verbleibenden zwei Kanten eines Dreiecks) in separaten Distanzfeldern kodiert werden. Bestimmt man für alle Distanzfelder eines Elements die Geradenstücke, so erhält man ein Doppelkreuz wie in Abb. 3.23(a) exemplarisch wiedergegeben. Zusätzlich gilt beim Erstellen der Distanztextur die Konvention, den dem Element zugewandten Bereich des Distanzfeldes (Abb. 3.22(a)) mit einem positiven Vorzeichen zu versehen, jenseits der entsprechenden Geraden liegende Bereiche hingegen mit einem negativen Vorzeichen zu behaften. Aufgrund dieser Tatsache lässt sich leicht zwischen dem Inneren und Äußeren eines Elements unterscheiden, denn nur im Inneren sind die Vorzeichen aller beteiligten Distanzfelder positiv (Abb. 3.23(a)). Mittels einer geeigneten Nachschlagetextur [RE03] oder entsprechender Fragmentprogrammanweisungen können somit alle Geradenteile außerhalb des Elements gekappt werden, indem man Bereiche, in denen mindestens eine der Distanzen unter $-\varepsilon$ liegt, abschneidet. Wie Abb. 3.23(b) demonstriert, wird mittels dieser Operationen aus den bilinear interpolierten Distanztexturen letztendlich genau die Umrandung eines finiten Elements wiederhergestellt.

Um auf diese Weise die Berandung eines Elements kodieren zu können benötigt man vier unabhängige Kanäle für das Speichern der Distanzfelder, eine RGBA-Textur ist somit für diesen Zweck ausreichend. Solange mindestens 2×2 Texel innerhalb eines Viereckselements zu liegen

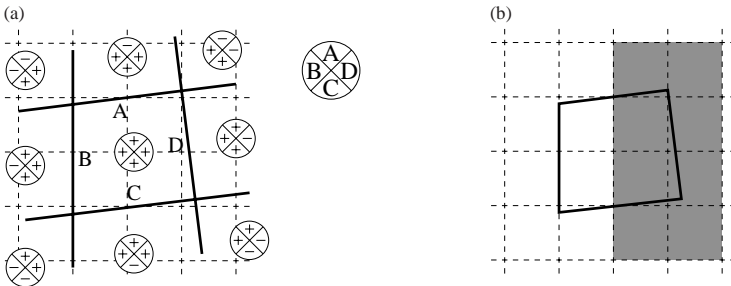


Abbildung 3.23: Vollständige Berandung eines finiten Elements: (a) Nur wenn die Vorzeichen aller vier beteiligten Distanzfelder positiv sind, befindet man sich im Inneren des finiten Elements (die Vorzeichen für die Distanzfelder der vier Geraden A, B, C und D sind für die verschiedenen Bereiche in dem Kompass-Symbol aufgeführt); (b) Mittels der Vorzeichenbedingung können die Geraden auf die Elementgrenzen zurechtgestutzt werden.

kommen, kann der Geltungsbereich zweier gegenüberliegender Kanten jedoch so eingeschränkt werden, dass die jeweiligen Distanzfelder der gegenüberliegenden Geraden im gleichen Farbkanal gespeichert werden können, ohne dass diese miteinander interferieren. Die Anzahl der für die Distanzkodierung eines Elements benötigten Farbkänäle sinkt damit auf zwei. Um dabei eine korrekte bilineare Interpolation in der Umgebung einer Kante zu ermöglichen und diese auch entsprechend zurechtztrimmen werden mindestens zwei Texel bzw. Distanzwerte innerhalb des Elements und insgesamt mindestens sechs Texel außerhalb des Elements benötigt, wie die acht in Abb. 3.23(b) grau hinterlegten Texel für die rechte Elementkante verdeutlichen.

Ebenso muss darauf geachtet werden, dass die Distanzfelder benachbarter Elemente sich nicht überschneiden. Mittels zweier Farbkänäle lassen sich daher nur ein Viertel der Elemente umranden, wie in Abb. 3.24 zu sehen. Es genügt jedoch, ein weiteres Viertel der Elemente zu zeichnen. Wenn die beiden Elementmengen entsprechend Abb. 3.25 schachbrettartig angeordnet werden,

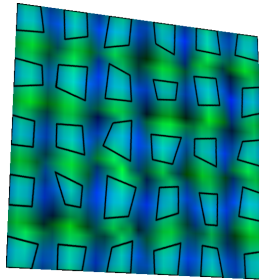


Abbildung 3.24: Eine zweikanalige Distanzfeldtextur (hier 32×32 Texel groß) ermöglicht die Restauration hochaufgelöster Berandungslinien von einem Viertel der Elemente.

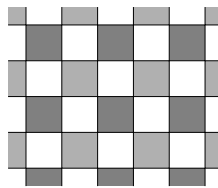


Abbildung 3.25: Die schachbrettartige Anordnung der grauen Elemente ermöglicht eine implizite Darstellung der Berandungslinien der weiß dargestellten Elemente, es muss somit nur die Hälfte der Elemente mit einer Distanztextur versehen werden, solange ein Viertel der Elemente (dunkelgrau) entsprechend dem gezeigten Schema in Kombination mit einem weiteren Viertel (hellgrau) angeordnet werden kann.

so erhält man implizit die Konturen der zweiten Hälfte der Elemente. Die Konturinformation eines Netzes, das nur aus Viereckselementen besteht, lässt sich daher in einer einzigen RGBA-Textur unterbringen, wobei jeweils einem Viertel der Elemente zwei Farbkanäle zur Ablage der Distanzwerte zugeordnet sind.

Dreieckige Elemente zerstören die schachbrettartige Anordnung jedoch und müssen daher entsprechend beachtet werden. Außerdem überlagern sich bei einem Dreieck die Distanzfelder aller drei Kanten, so dass eine Reduktion der verwendeten Texturfarbkanäle auf zwei – wie es bei Vierecken möglich ist – nicht angewendet werden kann. Als Lösung wäre es denkbar, Dreiecke separat zu behandeln und deren Umfangslinien z.B. in baryzentrischen Koordinaten zu kodieren, welche eine ähnlich hohe Informationsdichte in Hinblick auf die benötigte Anzahl an Texeln erlauben. Allerdings ist es aufgrund der durch Dreiecke hervorgerufenen Verschiebungen in der geordneten Schachbrettstruktur ohnehin unumgänglich, zwei weitere Distanzfelder einzusetzen, um Überschneidungen in solch gestörten Bereichen zu vermeiden. In einem gemischten Dreiecks-/Vierecksnetz benötigt man daher insgesamt sechs Farbkanäle, um sämtliche Berandungslinien wiedergeben zu können. In Abb. 3.26 ist jeweils ein Distanzfeldpaar, eingebettet in einen Texturatlas – analog zu Abb. 3.21(a) erstellt – für ein exemplarisches Bauteil gezeigt.

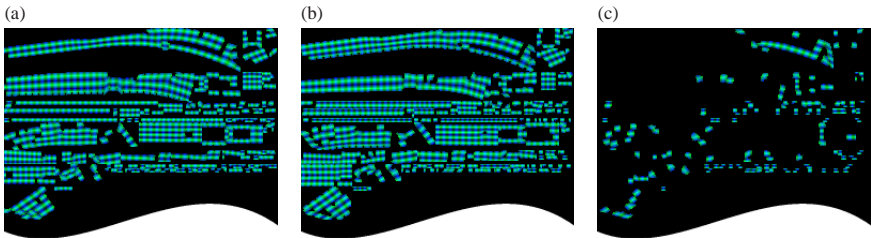


Abbildung 3.26: Distanzfeldtextur-Atlanten des in Abb. 3.6(b) bzw. 3.17 gezeigten Bauteils, jeweils 256×256 Pixel groß. Im Sinne des besseren Verständnisses ist hier die weniger dichte Anordnung der einzelnen Texturflecken analog zu Abb. 3.21(a) gewählt und der ungenutzte Texturbereich abgeschnitten.

In sehr seltenen Fällen, bei denen die Valenz eines FE-Knotens größer als sechs beträgt, ist auch diese Anzahl an Distanzfeldern nicht mehr ausreichend. Aufgrund der geringen Häufigkeit werden die überzähligen Kanten jedoch nicht mittels eines weiteren Distanzfeldes kodiert, da hierfür der Kosten/Nutzen-Faktor zu hoch ist. Stattdessen werden die Konturlinien, die nicht durch ein Distanzfeld abgedeckt sind mittels einfacher `GL_LINES`-Primitive wiedergegeben.

Verbindet man diese kompakte Repräsentation der Elementkonturlinien mit der Konservierung der Normalen in Texturen, so lässt sich diese Information in zwei RGBA-Texturen speichern. Berücksichtigt man die gegebenen Anforderungen an die Texturauflösung und einen gewissen Verlustfaktor aufgrund unterschiedlich großer und verschieden ausgerichteter Elemente sowie eine unvollständige Ausnutzung der zur Verfügung stehenden Texturfläche, so belegt jedes Element insgesamt etwa 10-12 RGBA Texel. Unter der bereits getroffenen Annahme eines

1280×1024 großen doppelt-gepufferten Bildschirmspeichers inklusive Tiefenpuffer können somit auf einer Graphikkarte mit 128MB Graphikspeicher circa 2,5 bis 3 Millionen finite Elemente auf den simplifizierten Oberflächen dargestellt werden⁹. Normalentexturen und Drahtgitterkonservierung erlauben dabei – im Vergleich zu einem reinen Simplifizierungsansatz ohne Verwendung von Texturen – eine bessere Polygonreduktionsrate auf etwa ein Fünftel der Ausgangsgeometrie ohne merkliche Qualitätsverluste wie Abb. 3.27 belegt. Die Umverteilung der Geometrie-last auf die – bei herkömmlichen Ansätzen kaum beanspruchte – Füllleistung der Graphikkarte ermöglicht hierbei eine Beschleunigung in der Darstellung um das bis zu Vierfache.

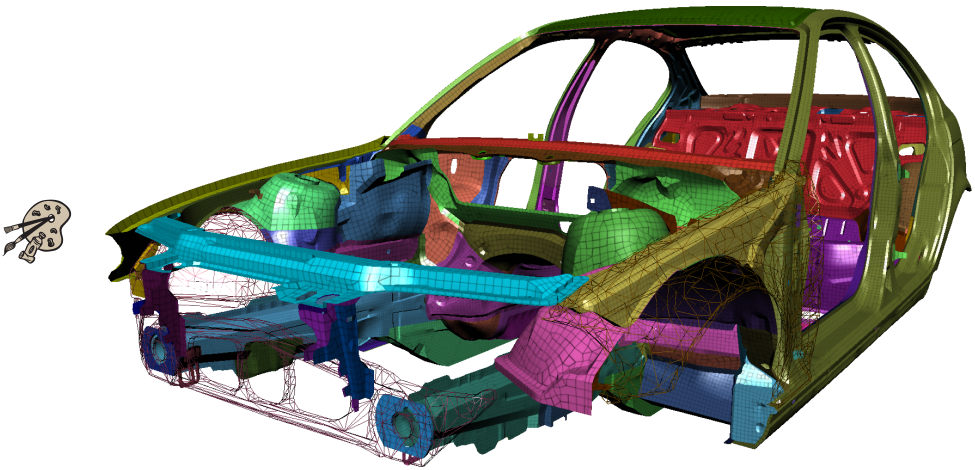


Abbildung 3.27: Simplifiziertes Gesamtfahrzeug mit restaurierten Elementberandungen und Phong-Beleuchtung/-Schattierung. An den in Drahtgitteransicht dargestellten Fahrzeugkomponenten im Vordergrund ist die zu Grunde liegende Simplifizierung zu erkennen.

Simplifizierung zur Laufzeit und Parallelisierung

Trotz der auf FE-Modelle spezialisierten und optimierten Simplifizierungsalgorithmen kann der Aufbau eines geometriereduzierten Gesamtfahrzeugmodells und das Generieren der benötigten Texturen je nach Komplexität auf einem durchschnittlichen Rechnersystem (AMD Athlon mit 1,8GHz) u.U. mehrere Minuten in Anspruch nehmen. Da die vorgestellten Algorithmen jedoch

⁹bei mehr Elementen bzw. höherem Speicherverbrauch findet eine Auslagerung über den AGP-Bus in den Hauptspeicher des Computers statt, so dass ein deutlicher Einbruch in der Bildwiederholrate zu verzeichnen ist

bauteilbasiert vorgehen, ist eine Simplifizierung zur Laufzeit, d.h. parallel zur Benutzerinteraktion möglich. Nach dem Laden eines Modells wird dabei zunächst das Originalmodell gezeichnet. Mit fortschreitender Zeit und Programm Benutzung, bei der sämtliche Interaktionsmöglichkeiten zur Verfügung stehen, werden nach und nach die einzelnen Fahrzeugkomponenten simplifiziert und gegen die Originalnetze ausgetauscht, so dass nach wenigen Minuten der Anwender schließlich mit maximaler Performanz interagieren kann. Bei eventuellen Änderungen an der Netzgeometrie wird ebenso zunächst auf der Originaloberfläche gearbeitet und nach Abschluss der Modifikationen im Hintergrund simplifiziert.

Auf Mehrprozessorsystemen kann zudem die Simplifizierung auf die zusätzlichen Prozessoren ausgelagert werden, so dass reguläre Aufgaben mit der vollen Leistung des ersten Prozessors durchgeführt werden können. Ebenso ist es – analog zu oben aufgeführtem Vorgehen – denkbar, dass mehrere Bauteile parallel und unabhängig voneinander simplifiziert werden können. Als dritte Variante lässt sich ein Bauteil auch parallel auf mehreren Prozessoren simplifizieren, wenn es zuvor anhand der Oberflächenmerkmale in mehrere, getrennt behandelbare Abschnitte aufgeteilt wird. Jedoch gestaltet sich bei letzterem Ansatz das Erstellen eines optimalen Texturatlanten als relativ aufwändig.

(If you are reading this on planet Earth then:

- a) Good luck to you. There is an awful lot of stuff you don't know anything about, but you are not alone in this. It's just that in your case the consequences of not knowing any of this stuff are particularly terrible, but then, hey, that's just the way the cookie gets completely stomped on and obliterated.
- b) Don't imagine you know what a computer terminal is. A computer terminal is not some clunky old television with a typewriter in front of it. It is an interface where the mind and body can connect with the universe and move bits of it about.)

Kapitel 4

Douglas Adams: Mostly Harmless

Intuitive und interaktive Modifikationsmechanismen

Eine performante und qualitativ hochwertige Darstellung allein genügt nicht, um eine Applikation bedienbar zu gestalten, sondern sie liefert vielmehr die Grundvoraussetzung für eine angenehme Arbeitsweise. Im Folgenden liegt daher das Hauptaugenmerk nicht auf Renderingsaspekten, sondern auf interaktiven Modifikationstechniken verbunden mit einer intuitiven und optimalen Bedienbarkeit.

Im Umgang mit dreidimensionalen Objekten ergibt sich dabei meist das Problem, dass bei Veränderungen an bestehender oder der Erzeugung neuer Geometrie die Eingabe absoluter oder relativer dreidimensionaler Koordinaten erforderlich ist. Bei vielen Anwendungen geschieht dies mittels entsprechender Eingabefelder, in welche solche Koordinaten-Tupel mittels der Tastatur eingegeben werden können. Obwohl dies eine sehr exakte Positionierung zulässt, ist diese Vorgehensweise sehr mühsam und aufwändig, insbesondere, wenn der Anwender keine Rückkopplung darüber erhält, an welcher Position er sich zur Zeit befindet und er somit keine unterstützenden Anhaltspunkte nutzen kann.

Die folgenden Unterkapitel beschäftigen sich hauptsächlich mit Interaktionsansätzen, die per Maus gesteuert werden können. Die Algorithmen sind so ausgelegt, dass eine intuitive Abbildung der dadurch gegebenen zweidimensionalen Eingabeparameter auf die dreidimensionalen Körper und Flächen erfolgt. Zunächst wird hier auf Methoden eingegangen, die sich an bereits existierende Algorithmen zur interaktiven Definition von Bauteilverbindungen orientieren und welche auf in der Arbeitsgruppe in Kooperation mit BMW entwickelten sowie bereits veröffentlichten Ergebnissen [Som03, Fri04] aufbauen.

4.1 Linienförmige und flächige Verbindungselemente

In Kapitel 2.1.3 und 2.2.3 wurde bereits kurz auf die Definition von Schweißpunkten und Schweißpunkt Nähten eingegangen. Schweißpunkte sind zur Zeit das mit Abstand am häufigsten eingesetzte Mittel, um benachbarte Bauteile oder ganze Bauteilgruppen miteinander zu verbinden. Sie spielen somit eine wichtige Rolle für die Tragfähigkeit bzw. Belastbarkeit einer Karosseriestruktur. Entsprechend wichtig ist die Beachtung physikalisch und technisch vorgegebener Randbedingungen, so dass die Simulation nicht durch fehlerhaft platzierte Schweißpunkte negativ beeinflusst wird. An dieser Stelle soll daher in Anlehnung an [FRSE02] im Kurzen die Vorgehensweise bei der manuellen Neudefinition von Schweißpunkten unter automatischer Berücksichtigung der entsprechenden Parameter näher erläutert werden.

Ein einzelner Schweißpunkt kann mittels zweier Mausklicks definiert werden. Hierbei bestimmt der erste Klick eines der zu verbindenden Bauteile, welches daraufhin in Drahtgitteransicht dargestellt wird und nicht mehr selektiert werden kann. Der nächste Klick legt sowohl den zugehörigen, zweiten Verbindungspartner als auch die Position des Schweißpunktes fest. Dazu wird zunächst der Durchstoßpunkt des Sichtstrahls¹ auf der zweiten Bauteiloberfläche bestimmt. Dieser Punkt wird anschließend auf die Oberfläche des ersten Bauteils projiziert. Die gemittelten Koordinaten dieser beiden Punkte legen die Position des Schweißpunktes fest, dessen zugehörigen Verbindungseigenschaften durch die Simulationssoftware interpretiert und entsprechend auf die beiden beteiligten Fahrzeugkomponenten angewandt werden. Der generierte Schweißpunkt wird mittels eines kleinen, roten Quaders visualisiert – in Abb. 4.1 hell dargestellt. Sein Zentrum ist durch die Schweißpunktcoordinate festgelegt und er wird entlang der Verbindungslinie ausgerichtet, welche durch die oben genannten Punkte auf den beiden verschweißten Materialien definiert ist.

Zur Erzeugung eines Schweißpunktes ist somit keine Eingabe der Koordinaten über die Tastatur nötig. Mittels der direkten Mauseingabe ist zwar keine genaue Platzierung möglich, die exakte Position ist jedoch von geringerer Bedeutung als die Einhaltung geforderter Kriterien. So sollte der Abstand der beteiligten Verbindungspartner unter einem vorgegebenen Höchstwert liegen, ebenso sollten die Bauteiloberflächen an der Schweißstelle eine maximale Winkelabweichung nicht überschreiten. Außerdem dürfen Schweißpunkte nicht zu nah beieinander platziert werden.

Unter Hinzunahme einer weiteren Mausklickoperation ist es möglich, nicht nur einen einzigen Schweißpunkt sondern lange Schweißpunktlinien zu erzeugen, welche auch entlang gekrümmter Flansche gezogen werden können. Der erste Klick bestimmt ebenfalls einen der Verbindungspartner, der zweite den Startpunkt und der letzte den Endpunkt der Schweißpunktnaht auf dem zweiten Bauteil. Die einzelnen Schweißpunkte werden entlang der Mittellinie des Flansches unter Einhaltung sämtlicher Randbedingungen erzeugt. Ein Flanschgebiet wird dabei durch diejenigen gegenüberliegenden Elemente der beiden beteiligten Komponenten definiert, welche einen entsprechend kleinen Abstand bei gleichzeitig geringer Verkipfung dieser Elemente gegeneinander aufweisen. So dehnt sich die Flanschregion des dunklen Bauteils in Abb. 4.1 großteils über zwei Elemente aus, im Bereich der Ausbuchtung hingegen lediglich über ein Element, so dass die

¹der Sichtstrahl verläuft durch das Kamerazentrum und die Mausposition auf der Bildebene

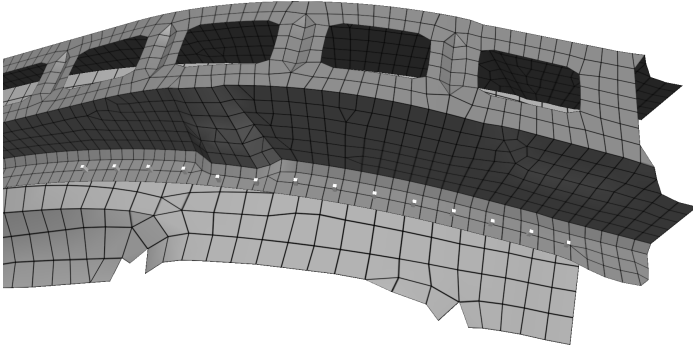


Abbildung 4.1: Erzeugen einer gültigen Schweißpunktlinie entlang eines Flansches. Die einzelnen Schweißpunkte werden durch rote (hier: helle) Quader repräsentiert.

Mittellinie – entlang derer die einzelnen Schweißpunkte unter Einhaltung des Mindestabstands zueinander erzeugt werden – an dieser Stelle entsprechend nach außen gedrängt ist. Gültige Schweißpunkte können somit nur innerhalb solcher Flanschregionen definiert werden. Außerhalb liegende Punkte werden nicht erzeugt, so dass z.B. bei gewellten Blechen ggf. auftretende Unterbrechungen in der Flanschregion korrekt gehandhabt werden.

Weiterführende Informationen zur Erzeugung von Schweißpunktnähten entlang von Flanschen finden sich in [Fri04]. Im Folgenden werden Verbindungstypen betrachtet, welche die punktförmigen und daher nulldimensionalen Verbindungen um jeweils eine Dimension erweitern. Die bei diesen Elementen ebenfalls zu beachtenden Qualitätskriterien sind in der Regel aufgrund der höheren Dimensionalität etwas aufwändiger zu handhaben. In den folgenden Abschnitten wird zunächst auf die Erzeugung und Validierung von eindimensionalen Schweißnähten und daran anschließend auf die entsprechenden Operationen bei zweidimensionalen Klebeverbindungen eingegangen.

4.1.1 Schweißnähte

Schweißnähte bestehen im Unterschied zu Schweißpunktnähten nicht aus einzelnen Punkten sondern durchgängigen Linienzügen. Zudem werden Schweißnähte nur in Ausnahmefällen entlang von Flanschmittellinien gezogen. Stattdessen werden solche linienförmigen Verbindungen in der Regel entlang der Außenkante von einem der zu verschweißenden Bauteile platziert. Der Grund für die Bevorzugung dieser sogenannten Edge Links ist die technisch einfachere Realisierbarkeit. Im Gegensatz dazu ist das Ziehen einer Schweißnaht auf Oberflächen – also nicht entlang einer Kante – relativ aufwändig und erfordert die Anwendung eines speziellen Schweißverfahrens (Laserschweißen). Diese Nahtart wird bei einer Simulation zwar sehr ähnlich gehandhabt, ist jedoch im Allgemeinen dennoch durch einen eigenen Verbindungstyp (Line Link)

repräsentiert. Um die Unterschiede zwischen den beiden Schweißnahttypen zu verdeutlichen, sind verschiedene Varianten beispielhaft in Abb. 4.2 skizziert.

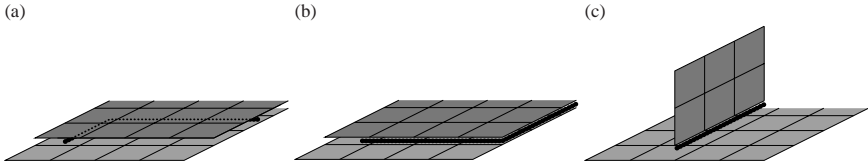


Abbildung 4.2: Unterscheidung zwischen inneren (Line Link) und Kantenschweißnähten (Edge Link): (a) Innere Schweißnähte verlaufen innerhalb der Flächen der verbundenen Bauteile; (b) Eine Kantenschweißnaht verläuft entlang der Bauteilaußenkante einer oder beider beteiligter Blechkomponenten; (c) Bei einem T-Stoß verläuft die Naht ebenfalls entlang einer Außenkante und wird folglich als Edge Link behandelt.

Interaktive Definition von „Line Links“

In Anlehnung an das Vorgehen bei Schweißpunktnähten wird bei der Erzeugung einer inneren Schweißnaht ebenfalls zunächst einer der Verbindungspartner mittels Mausclick ausgewählt und der Start- und Endpunkt der Naht anhand der folgenden Klicks auf dem zweiten Bauteil festgelegt. Da sich eine solche Liniennaht im Allgemeinen nicht zwingend an Merkmalen der geometrischen Bauteilstruktur orientiert, entfällt die Bestimmung von potenziell geeigneten Flanschmittellinien. Vielmehr wird eine möglichst geradlinige Naht zwischen den beiden gewählten Punkten erzeugt. Dazu wird zunächst von der direkten Verbindungslinie zwischen Start- und Endpunkt ausgegangen und auf dieser äquidistant² eine Menge an Punkten erzeugt. Diese Punkte werden zunächst auf die Oberfläche des ersten Bauteils abgebildet, indem jeweils der nächstliegende Punkt auf dieser Fläche gesucht wird. Die erhaltene Punktmenge wird anschließend nach dem gleichen Verfahren auf die zweite Verbindungskomponente projiziert. Die Punktkoordinaten aus beiden Listen können nun paarweise gemittelt werden und man erhält somit den Verlauf der Schweißnaht zwischen den beiden Fahrzeugkomponenten. Die einzelnen Schweißnahtelemente werden dabei durch ein Liniensegment definiert, welches jeweils von einem gemittelten Punkt zum in der Liste darauffolgenden aufgespannt wird.

Diese Vorgehensweise ist ebenso wie Schweißpunktnähte robust gegenüber veränderlichen Bauteilabständen oder Löchern in den FE-Oberflächen, da anhand der beiden Punktlisten schnell der lokale Abstand zwischen den Blechen bestimmt werden kann und ggf. ungültige Kandidaten bereits vor der endgültigen Erzeugung der Verbindungselemente aussortiert werden können. Sollten durch die Projektion die erhaltenen Linienabschnitte zu kurz ausfallen, so können benachbarte Linienteile zu größeren Segmenten verschmolzen werden. Ebenso können zu lange

²der Abstand zwischen diesen Abtastpunkten sollte in der Größenordnung der Elementkantenlängen liegen

Linienabschnitte unterteilt werden, indem man weitere Abtastpunkte auf der ursprünglichen Verbindungsgeraden an entsprechender Stelle einstreut.

Interaktive Definition von „Edge Links“

Bei der Erzeugung von „Edge Links“ muss genauer unterschieden werden, ob die Naht nur an der Außenkante eines Bauteils entlang verläuft, d.h. eine Verbindung von Kante zu Fläche eingegangen wird (siehe Abb. 4.2(b) Mitte bzw. Abb. 4.2(c)), oder ob sich die Schweißlinie entlang des Kantenverlaufs beider Bauteile bewegt (Verbindung von Kante zu Kante, Abb. 4.2(b) rechts).

Bei ersterer Verbindungsart genügt es – nach der analog zum bisherigen Vorgehen verlaufenden Benutzereingabe mittels drei Mausklicks – die Kantenknoten des einen Bauteils auf die nächstliegenden Punkte auf der Oberfläche des anderen Bauteils abzubilden. Die gemittelten Koordinaten der Kantenknoten und der zugehörigen Projektionen beschreiben wie gehabt den Verlauf der Schweißnaht. Eine genauere Analyse der Schweißelementlänge ist jedoch nicht erforderlich, da – bedingt durch den geforderten kurzen Abstand zwischen den betreffenden Bauteilen – keine großen Verzerrungen auftreten können und die Schweißnahtelemente somit ungefähr die gleiche Länge wie die Elementkanten des Bauteils besitzen.

Etwas aufwändiger gestaltet sich die Bestimmung der Mittellinie bei Kanten-Kanten-Verbindungen. Um einen möglichst optimalen Verlauf der Schweißnaht entlang der beiden Außenkanten zu gewährleisten, werden die Knoten der beiden Kanten jeweils auf die gegenüberliegende Kante projiziert wie in Abb. 4.3 exemplarisch gezeigt. Bei nicht genau parallel verlaufenden Kanten ergibt sich somit eine visuell ansprechendere Repräsentation der Mittellinie³. Auch hier gilt jedoch, dass zu kurze Linienabschnitte aussortiert werden sollten, indem sehr nahe beieinander liegende Knotenpunkte zu einem gemeinsamen Knoten – ähnlich dem Vorgehen bei einem „Edge Collapse“ (siehe Kapitel 3.2.2) – verschmolzen werden.

³für eine korrekte Simulation ist die genaue Lage eines Schweißnahtelements unerheblich, da im Simulationsalgorithmus die Verbindung über die Knoten in der Umgebung der Naht hergestellt wird und nicht über das Verbindungselement selbst

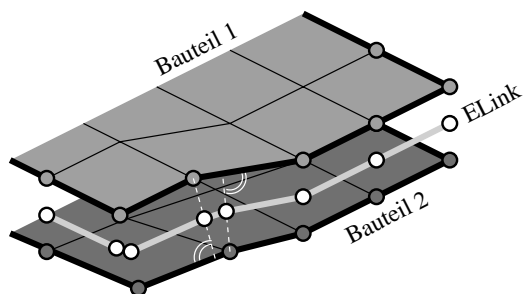


Abbildung 4.3: Bestimmung der Mittellinie von Schweißnähten entlang von Außenkanten.

Die Entscheidung, ob ein „Edge Link“ oder ein „Line Link“ gesetzt werden soll kann vom Anwender über eine entsprechende Platzierung von Start- und Endpunkt gesteuert werden. Er muss sich somit nicht im Voraus auf einen bestimmten Verbindungstyp festlegen. Liegen die beiden entsprechenden Mausklickpositionen in der Nähe einer Kante von einem oder beiden Bauteilen, so wird automatisch eine Außennaht entlang dieser Kante gezogen. Befindet sich hingegen einer oder beide ausgewählten Punkte nicht in der Nähe einer Außenkante, so wird ein „Line Link“ erzeugt.

Graphische Darstellung

Wie auch bei Schweißpunkten tritt zumindest bei inneren Schweißnähten das Problem auf, dass die einzelnen Elemente zwischen den verbundenen Bauteilen liegen und daher von außen nicht sichtbar sind. Bei einzelnen Schweißpunkten wurde dieses Sichtbarkeitsproblem mit länglichen Quadrern als Glyph für die einzelnen Punkte gelöst. Analog kann bei Schweißnähten vorgegangen werden, indem die Schweißnahtlinie als Skelettlinie eines Prismenzuges dient.

Die Form der Grundfläche der Prismenelemente kann dabei einen Hinweis auf den Nahttyp liefern. Ein dreieckiger Querschnitt symbolisiert dabei eine Kantenschweißnaht, ein quadratischer Querschnitt hingegen wird für „Line Links“ eingesetzt und ist zusätzlich um 45° gedreht, wie in Abb. 4.4 zu erkennen ist. Diese rotierte Darstellung hat den Vorteil, dass der Abstand zwischen den Bauteilen anhand der überstehenden Prismensegmente auch von außen betrachtet stets gut abschätzbar ist. Die geometrisch weniger anspruchsvolle Visualisierung der „Edge

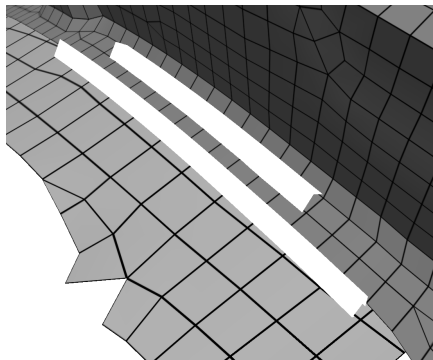


Abbildung 4.4: Repräsentation einer inneren (Line Link) und einer Kantenschweißnaht (Edge Link).

Links“ mittels dreieckiger Prismen wurde gewählt, da diese Verbindungsart aus den bereits diskutierten Gründen ungleich häufiger auftritt als „Line Links“. Im Hinblick auf eine performante und außerdem übersichtliche Darstellung sollten daher für den häufiger auftretenden Schweißnahttyp möglichst wenige graphische Primitive eingesetzt werden. Gegebenfalls kann zusätzlich

der Durchmesser des Prismenzuges dazu herangezogen werden, die maximal erlaubte Distanz zwischen zwei Bauteilen zu visualisieren, bis zu der eine Schweißnaht gültig ist. Da jedoch fehlerhafte Elemente – wie im folgenden gezeigt – ohnehin markiert werden, ist diese Information nicht unbedingt notwendig, und bei häufiger Variation der erlaubten Materiallücke sind unterschiedlich dicke Schweißnahtlinien u.U. eher verwirrend.

Detektion und Visualisierung fehlerhafter Schweißnahtelemente

Bestehende Schweißnähte werden ähnlich wie Schweißpunkte auf Fehler überprüft, so dass die Gültigkeit von extern – d.h. mittels anderer Applikationen – definierten oder von importierten Schweißnähten stets gewährleistet ist. Analog zu den punktförmigen Verbindungen werden verschiedene Sachverhalte überprüft. Im einzelnen sind dies fehlende Verbindungsbauteile, zu weit von der Naht entfernte Schweißkomponenten, zu große Winkelabweichungen zwischen verschweißten Elementen, zu nah beieinander platzierte Schweißverbindungen oder Überschneidungen mit an der Verbindung nicht beteiligten Fahrzeugkomponenten.

Etwas aufwändiger als bei Schweißpunkten gestalten sich hierbei die Distanzberechnungen. So reicht zur Distanzbestimmung zwischen naheliegenden Nähten die Berechnung des räumlichen Abstands zwischen den einzelnen Schweißelementknoten nicht aus, sondern es muss ebenso die Distanz der Knoten zum jeweils benachbarten Liniensegment und der Abstand zwischen den einzelnen, im Allgemeinen windschiefen Elementlinien berücksichtigt werden, wie in Abb. 4.5(a) skizziert ist. Ebenso muss bei der Entfernungsbetrachtung zu den verschweißten Bauteiloberflächen zusätzlich zu den Abständen Punkt zu Punkt, Punkt zu Kante und Punkt zu Fläche auch die Distanz des Schweißliniensegments zu den Bauteilelementkanten bzw. zu den Elementknoten beachtet werden (Abb. 4.5(b)).

Um aus der Vielzahl an finiten Elementen das jeweils nächstliegende performant bestimmen zu können, kommt eine Bounding-Volume-Hierarchie [GLM96] zum Einsatz, welche es ermöglicht, die Suche nach den entsprechenden Kandidaten auf einen logarithmischen Aufwand zu reduzieren. Als weitere Beschleunigungsmaßnahme kann für die Überprüfung der Einhaltung von Maximaldistanzen eine Überschätzung verwendet werden, welche lediglich die von den Bauteilknoten gemessenen Distanzen in Betracht zieht. Diese Vorgehensweise ist zwar ungenau, aber garantiert die Einhaltung der oberen Schranke und ist konform mit der i.A. üblichen Vorgehensweise der Simulationsalgorithmen.

Zusätzlich zu den aufgezählten Fällen können bei Schweißnähten auch Überschneidungen untereinander auftreten. Dieser Fall wird jedoch implizit durch die Forderung nach einem gewissen Mindestabstand der Nähte zueinander und die damit verbundene Abstandsberechnung abgedeckt.

Die Visualisierung der ungültigen Nahtelemente erfolgt durch eine dem Fehlerfall entsprechende Farbgebung analog zu den bei Schweißpunktverbindungen auftretenden Fällen. Auf eine geometrische Veränderung der einzelnen Elemente wurde jedoch an dieser Stelle bewusst verzichtet, um eine visuelle Überfrachtung zu vermeiden. Außerdem sind farbliche Veränderungen innerhalb einer kontinuierlich verlaufenden Schweißnaht deutlich besser zu erkennen als bei einzelnen

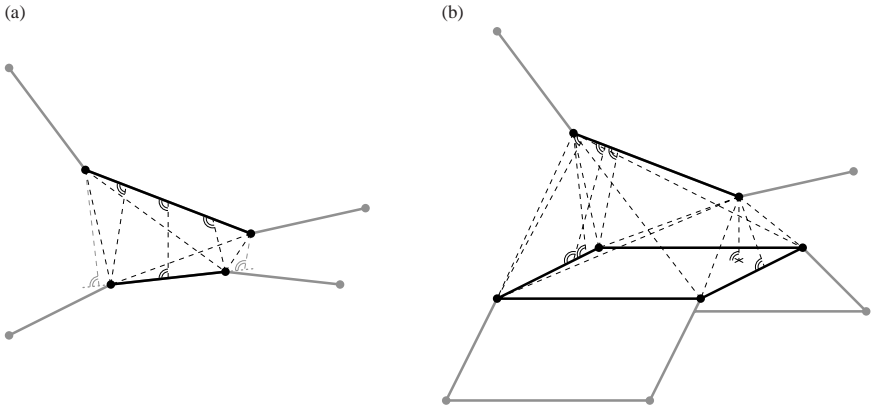


Abbildung 4.5: Distanzanalyse von Schweißnähten im dreidimensionalen Raum: (a) Abstände zwischen zwei Schweißnahtlinien, graue Elemente und Distanzen werden hier nicht betrachtet; (b) Distanzen eines Schweißliniensegments zu einem finiten Viereckselement, graue Elemente sind hier nicht analysiert, im Sinne der Übersichtlichkeit sind außerhalb der Parametergrenzen liegende Distanzstrecken nicht eingezeichnet.

Schweißpunkten (Abb. 2.16). Sind mehrere zusammenhängende Schweißnahtelemente ungültig, so werden diese zu einer gemeinsamen Gruppe zusammengefasst, welche dem Anwender durch eine entsprechende Kameraeinstellung als Ganzes präsentiert werden kann, so dass dieser nicht durch die einzelnen fehlerhaften Verbindungselemente navigieren muss.

Erzeugung von Schweißnähten aus CAD-Kurven

Um die Zusammenarbeit mit der CAD-Abteilung zu erleichtern, lassen sich Schweißnahtlinien mittels sogenannter VDAFS-Dateien von dort importieren. Das VDAFS-Format erlaubt die Speicherung verschiedener geometrischer Elemente, u.a. Punkte, Kreise, Flächen oder Kurven. Für den Austausch von Schweißnahtverläufen sind vor allem Kurvenzüge im dreidimensionalen Raum von Interesse. Abbildung 4.6 zeigt die Kurvendefinition einer Kantenschweißnaht. In diesem Beispiel besteht der Kurvenzug aus drei zusammenhängenden Polynomen von unterschiedlich hohem Grad, nämlich zwei Linien (Polynomgrad 1 bzw. Polynomordnung 2) und ein Polynom fünften Grades (Ordnung 6). Die jeweiligen Dateiabschnitte – in der Abbildung mit den Buchstaben A, B und C markiert – enthalten die dem Grad entsprechende Anzahl an Polynomkoeffizienten a_i in allen drei Raumachsen, um den Verlauf der Koordinate x entsprechend

```

VDAFS = HEADER / 0
$$vip:property:1,?,?,11411,13001
$$vip:1d-esseam,873,1,CV000001
CV000001 = CURVE /      3, 0., 1., 2., 3.,
— 2, 550.55757418176, 31.334442891281, -460.92409264443,
A 1.7409706452436, -31.42144612588, -21.940616014869,
— 6, 581.89201707304, 240.70749993377, 51.678972001547,
-14.836148173078, -1.5972695621185, .1960349244207, -459.18312117273,
B 13.373817753357, -6.7325770500265, -.85282167351033, .08333221655721,
— .11794054807456, -53.362071560816, -168.54393219646, 73.280878877993,
10.713363389719, -2.3751704159957, -.13164843610261,
C 2, 858.04110619758, 21.331693035444, -453.19342937828,
— -.12523126576139, -140.41858034166, .62295953284774E-7
$$$ Bauteildefinitionen
$$$ :Sachnr. |Ax|Al|Benennung |Material |Dicke|
$$part:11411 | G | A | LI BODENBLECH VORN |11411 | 0.65|
$$part:13001 | S | A | ZB LI MOTORTRAEGER HINTEN |13001 | 1.60|
VDAFS = END

```

Abbildung 4.6: Beispiel für eine VDAFS-Eingabedatei, in der eine gekrümmte Außenschweißnaht definiert wird.

dem Parameter t entlang eines Kurvenstücks rekonstruieren zu können:

$$\mathbf{x}(t) = \sum_{i=0}^{\text{Grad}} \mathbf{a}_i t^i \quad (4.1)$$

Über spezielle Kennwörter kann hier zudem in die Entscheidung eingegriffen werden, welcher Nahttyp erzeugt werden soll. Um aus einer Kurve dementsprechende Verbindungselemente generieren zu können, muss diese in lineare und möglichst äquidistante Segmente unterteilt werden. Zwar ist die Forderung nach gleich langen Elementen bei Schweißnähten nur von untergeordneter Rolle, aber da zur Verteilung einzelner Schweißpunkte auf einer gekrümmten Schweißpunktlinie ohnehin ein entsprechender Algorithmus implementiert werden musste, kommt dieser hierfür ebenfalls zum Einsatz. Durch Berechnen der Bogenlänge eines einzelnen Kurvensegments

$$s = \int_{t_0}^{t_1} |\mathbf{x}'(t)| dt \quad (4.2)$$

lässt sich durch Aufsummieren die Gesamtlänge der Kurve ermitteln, so dass in diese Länge ein ganzzahliges Vielfaches der nach unten beschränkten Elementlänge eingepasst werden kann. Numerisch können damit unter Anwendung von (4.2) äquidistant verteilte Stützpunkte für die einzelnen Schweißnahtelemente ermittelt werden. Abbildung 4.7 zeigt die Kantennaht, welche mit diesem Verfahren aufgrund der in Abb. 4.6 gegebenen Daten generiert wurde.

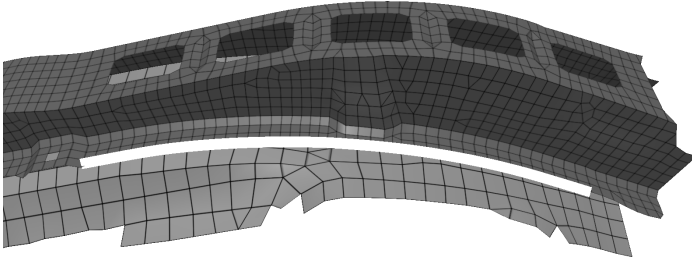


Abbildung 4.7: Aus VDAFS-Daten generierter „Edge Link“.

4.1.2 Klebeverbindungen

Im Automobilbau kommen neben Schweißnähten und Verschraubungen zunehmend auch modernere Verbindungstechniken zum Einsatz. Zu diesen zählen Klebeverbindungen, deren Verbindungseigenschaften auf Simulationsmodelle entsprechend übertragen werden können. Klebeschichten fügen ein Bauteilpaar mittels einer flächigen Verbindung zusammen und werden entsprechend durch zweidimensionale Elemente zwischen den beiden verbundenen Fahrzeugkomponenten repräsentiert. Die Interaktionsmethoden, Algorithmen und Qualitätskriterien sind jedoch in vielen Gesichtspunkten denen bei eindimensionalen und punktförmigen Verbindungselementen ähnlich. Im Folgenden werden die entsprechenden Erweiterungen daher nur knapp erläutert.

Interaktive Definition von Klebeverbindungen

Analog zu dem in den letzten Abschnitten vorgestellten Vorgehen, erfolgt die interaktive Definition von Klebeverbindungen ebenfalls mittels dreier Mausklicks. Wie auch bei Schweißpunkt­nähten werden daraufhin die umgebenden Bauteilelemente auf die Erfüllung der Flansch­kriterien hin analysiert und bis zu einem benutzerdefinierten Abstand von der Flanschmittellinie als zukünftige Kontakt­oberfläche für den Kleber markiert. Die Suche nach neuen Kontaktelementen wird dabei solange fortgesetzt, bis in der direkten Nachbarschaft der bereits hinzugefügten Elemente keine weiteren geeigneten Kandidaten gefunden werden. Um Unterbrechungen in der Klebeschicht – z.B. aufgrund von Ausbeulungen oder Schlitzen – zu überwinden, werden zusätzliche Saatelemente entlang der Flanschmittellinie generiert, sofern diese die bereits erwähnten Flanschbedingungen erfüllen.

Allerdings ist bei der Ausbreitung der Klebefläche durch Einfügen benachbarter Flanschelemente darauf zu achten, dass sich der Klebestreifen nicht über den vom Anwender vorgegebenen Start- bzw. Endpunkt hinausbewegt. Zu diesem Zweck wird an diesen Punkten jeweils eine Trennebene eingefügt, welche senkrecht auf der Bauteiloberfläche steht. Als erster, grober Anhaltswert für die Normale dieser Trennebene wird die Richtung der vom Start- bzw. Endpunkt ausgehenden Flanschmittellinie herangezogen, welche in Abb. 4.8 durch \vec{p} angegeben ist. Durch Projektion

der Knoten des Start-/Zielelements auf diesen Richtungsvektor lassen sich mittels Bewertung des Skalarprodukts die beiden am weitesten außen liegenden Punkte ermitteln. Im hier skizzierten Fall sind dies die Knoten A und B. Der Vektor \vec{n} vom Mittelpunkt der dadurch aufgespannten Elementkante AB zum Elementmittelpunkt definiert schließlich die Normale der Begrenzungsebene. Liegt der Mittelpunkt eines im Zuge der Kontaktflächenausbreitung neu gewonnenen Elements jenseits dieser Ebene, so wird es nicht berücksichtigt. Dieses etwas umständlich anmutende Vorgehen hat sich in der Praxis bewährt, da es zum einen einen eindeutig festgelegten sowie geradlinigen Abschluss eines Klebestreifens ermöglicht und zum anderen die lokal gegebene Ausrichtung der Elemente berücksichtigt.

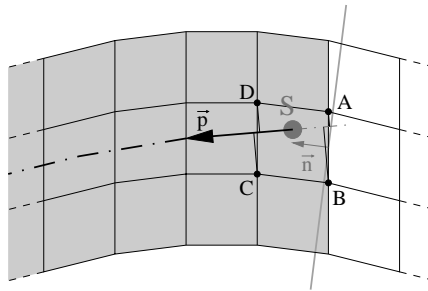


Abbildung 4.8: Anfang bzw. Ende eines Klebestreifens werden durch eine anhand des ersten bzw. letzten Elements ausgerichtete Ebene begrenzt.

Die Knoten der so gewonnenen Kontaktelemente des ersten Bauteils können daraufhin auf die gegenüberliegende Oberfläche des zweiten Bauteils projiziert werden. Den Verlauf der Klebeschichtfläche erhält man daraus durch Mittelung der originären und projizierten Knotenkoordinaten. Besitzen die beiden Fahrzeugkomponenten unterschiedliche Blechstärken, so kann die exakte Lage der Klebeschicht durch eine entsprechend gewichtete Mittelwertbildung gewonnen werden. Die somit resultierende asymmetrische Lage der Klebefläche dient jedoch lediglich als visuelle Rückkopplung für eine unterschiedliche Dickenverteilung, da die Position des Klebestreifens – wie auch bei Schweißnähten – keinen direkten Einfluss auf die Interpretation durch die Simulationsalgorithmen hat.

Graphische Darstellung

Im Allgemeinen verläuft ein Klebestreifen zwischen den beiden Blechstücken, die er verbindet, und ist daher von außen betrachtet nicht zu sehen. Greift man die Idee der bei Schweißverbindungen verfolgten Strategie auf, die Verbindungselemente durch eine entsprechend ausgedehnte Geometrieform darzustellen, so ergibt sich jedoch das umgekehrte Problem, dass die Klebeschichtrepräsentation die Bauteile verdeckt. Eine Möglichkeit wäre eine transparente Darstellung der Klebeschicht, was jedoch eine u.U. komplexe, räumliche Sortierung der Klebe-

schichtelemente notwendig macht um eine korrekte Verdeckung der Elementfacetten untereinander zu gewährleisten. Die geeignetere und zur Darstellung der Klebeschichtelemente gewählte Methode lehnt sich an das Erzeugen von Transparenz mittels dem „Screen-Door“-Verfahren [FDF⁺94, MGW98] an. Das heißt, die Oberfläche der verdickten Klebeschichtelemente wird mit einer Luminanz-Alpha-Textur überzogen, welche ein schachbrettartiges Muster darstellt, das die darunterliegenden Bauteilflächen nur teilweise verdeckt, wie in Abb. 4.9 zu erkennen ist.

Um die gezeigte, geschlossene Form der Klebeschicht zu erhalten, werden die Vertizes der Klebelemente jeweils einen bestimmten Betrag entlang ihrer Normalen verschoben, so dass eine untere und obere Deckfläche entsteht. Dabei ist darauf zu achten, dass die Knotennormalen so ausgerichtet werden, dass sie auf dieselbe Flächenseite zeigen. Zusätzlich werden am Rand der Klebeschicht dazu orthogonal stehende Elemente eingefügt, wobei der Verlauf der Randlinie nach der bereits in Kapitel 3.1.2 bzw. Abb. 3.7(a) präsentierten Methode ermittelt wird.

Diese Art der Darstellung bildet einen guten Kompromiss und lässt zudem die Gestalt der einzelnen Klebelemente gut erkennen. Mittels der Dicke der Klebeschichtrepräsentation kann dem Anwender eine hilfreiche Rückkopplung gegeben werden, wie groß die Klebelücke zwischen zwei Bauteilen maximal werden darf. Wird der Abstand zu groß, so verschwindet an dieser Stelle die texturierte Oberfläche unter der Bauteilfläche.

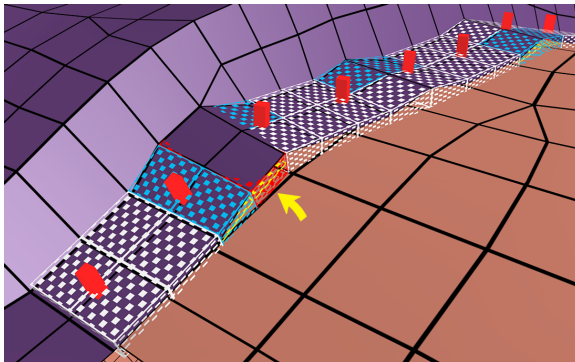


Abbildung 4.9: Visualisierung einer Klebeschicht. Ungültige Verbindungselemente sind farblich hervorgehoben, und um dem Anwender eine genauere Analyse zu ermöglichen, ist zusätzlich die exakte Lage des Klebelements gelb eingezeichnet (Pfeil).

Detektion und Visualisierung fehlerhafter Klebereiche

Bei Klebeflächen gibt es ähnliche Fehlerkriterien wie bei Schweißverbindungen. Auch hier gilt es, Flanschabstände nicht zu überschreiten, Verkippungen gegenüberliegender Bauteilelemente zu vermeiden und Überschneidungen bzw. Durchdringungen unterschiedlicher Klebepaarungen

zu verhindern. Im Gegensatz zu Schweißverbindungen brauchen zwischen aneinander grenzenden Klebenähten keine Minimalabstände eingehalten werden, so dass diese Überprüfung entfällt.

Die für die Validierung von Klebeflächen notwendigen Distanzbetrachtungen zwischen benachbarten Flächenelementen lassen die bereits bei Schweißnähten im Vergleich zu Schweißpunkten erhöhte Komplexität weiter ansteigen. Daher ist es bereits im Vorfeld einer Fehlerüberprüfung nötig, den Datensatz auf ein geeignetes Maß an Elementen in der näheren Umgebung der Klebeschicht zu reduzieren. Dazu wird die minimale Entfernung aller Fahrzeugkomponenten zu einer reduzierten Menge an Klebeschichtvertizes ermittelt und alle Bauteile, die garantiert außerhalb des Nachbarschaftsbereichs der Klebeschicht liegen, aus der Liste der genauer zu analysierenden Komponenten aussortiert. Dadurch wird eine entsprechende Ausdünnung in der Bounding-Volume-Hierarchie erzielt und der Aufwand für die Suche nach umgebenden Elementen minimiert. Die Überprüfung der tolerierten Bauteilabstände wird somit deutlich beschleunigt.

Liegen in der Nachbarschaft einer Klebeverbindung mehrere Bauteile eng beieinander, so ist eine Aussage über gegenseitige Durchdringungen mittels reiner Abstandsbetrachtungen u.U. schwierig, da potenziell alle im Einflussbereich⁴ einer Klebeschicht liegenden Bauteilelemente mit der gewünschten Klebeverbindung zweier Materialien interferieren können und diese ggf. trennen. Abbildung 4.10 verdeutlicht dies. Hier sollen Bauteil 1 und 2 durch den dunkelgrauen Klebe-

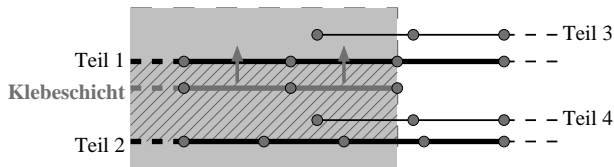


Abbildung 4.10: Die Klebeschicht wird von Bauteil 4 durchdrungen. Der Einflussbereich der Klebeverbindung ist grau hinterlegt, die wahre Ausdehnung der Klebeschicht ist schraffiert.

streifen verbunden werden. Obwohl Bauteil 3 räumlich näher an der Klebeschicht liegt als Bauteil 2 kommt es nicht zu einer Durchdringung der Klebeschicht. Bauteil 4 hingegen trennt die Klebeverbindung trotz eines größeren Abstands zur Klebefläche als Bauteil 1. Relativiert man jedoch die Abstände in Bezug auf die Normalen der Klebeelemente, so kann man eindeutige Aussagen treffen, welche Bauteile die Klebeschicht durchdringen [FRSE02]. Unter Zuhilfenahme dieser Normalen lässt sich auch leicht beurteilen, ob die Klebeschicht tatsächlich zwischen den Bauteilen 1 und 2, wie eingezeichnet, verläuft. Überschätzt man zunächst den Abstand der Verbindungspartner zur Klebeschicht, indem man lediglich den Abstand der Bauteilknoten zur Kleboberfläche bestimmt, so lassen sich in der Regel fast alle finiten Elemente, die nicht mit der Klebeschicht in Kontakt treten, relativ schnell aussortieren, so dass lediglich für wenige Elemente eine entsprechend genaue Abstandsberechnung durchgeführt werden muss.

⁴der Einflussbereich einer Klebefläche erstreckt sich senkrecht dazu in beide Richtungen, so dass alle FE-Knoten der beteiligten Bauteile innerhalb des für die Klebeverbindung maximal erlaubten Flanschabstands erfasst werden

Ungültige Elemente werden entsprechend dem festgestellten Fehlerfall farblich markiert. Zusätzlich wird die exakte Lage des Klebeschichtelements⁵ wie in Abb. 4.9 gelb eingezeichnet, um bei Bedarf dem Anwender eine genauere Begutachtung des aufgetretenen Fehlerfalls zu ermöglichen. Um die Bewertung fehlerbehafteter Verbindungen für den Benutzer weiter zu beschleunigen, werden benachbarte Verbindungselemente, welche die gleichen Gültigkeitskriterien verletzen, zu einem gemeinsamen Gebiet zusammengefasst und eine Kameraperspektive berechnet, welche dem Anwender auf Anfrage ein schnelles Navigieren zu diesem Cluster ermöglicht.

Erzeugung von Klebeverbindungen aus CAD-Kurven

Die Basis für die Generierung von Klebeverbindungen aus CAD-Daten bildet die auch bei Schweißverbindungen eingesetzte Beschreibung von Kurven im VDAFS-Format. Die Kurve wird hierbei nicht direkt zur Erzeugung von Klebeelementen herangezogen, sondern dient als Skelettlinie, entlang derer Saatpunkte für die Suche nach geeigneten Flanschelementen generiert werden. Das Vorgehen ist dabei dasselbe wie bei der regulären, interaktiven Erzeugung, welche zu Beginn dieses Unterkapitels vorgestellt wurde. Die Breite des anhand der Flanschelemente letztendlich generierten Klebestreifens kann durch den Anwender vorgegeben werden. Seine Länge wird – ebenfalls analog zum erwähnten Verfahren – durch zwei Ebenen begrenzt, welche jeweils senkrecht auf den beiden Enden der CAD-Kurve stehen.

4.2 Sensorpunkte

Während eines Crashversuchs treten neben den sichtbaren Verformungen auch Kräfte und Beschleunigungen auf, die an definierten Stellen mittels Sensoren gemessen werden. Um die Rechenergebnisse mit der Realität vergleichen zu können, bieten die entsprechenden Simulationsprogramme mittlerweile einfache Möglichkeiten an, solche Sensoren nachzubilden.

Je nach Anwendungszweck kommen unterschiedliche Arten von Sensoren zum Einsatz. Eine exemplarische Auswahl zeigt Abb. 4.11(a). Im Allgemeinen werden solche Sensoren an einem Bauteil festgeklebt oder -geschraubt. Das Zentrum der eigentlichen Mess-Sensorik liegt dabei konstruktionsbedingt nicht auf der Bauteiloberfläche, sondern – wie in Abb. 4.11(b) skizziert – um einen sensortypischen Abstand D verschoben.

Visualisierung von Sensoren und Interaktionsmöglichkeiten

Die Visualisierung des Sensorkörpers erfolgt durch einen Oktaeder, dessen eine Spitze auf dem Fußpunkt sitzt, an dem der Sensor auf dem Bauteil befestigt ist. An der gegenüberliegenden Spitze liegt das Zentrum der Messaufnahme innerhalb des Sensorkörpers. Die Höhe des Oktaeders gibt somit den sensortypischen Oberflächenabstand D wieder. An der oberen Oktaederspitze wird

⁵bei gültigen Klebeelementen wird auf die explizite Darstellung der Mittelebene aus Performanzgründen verzichtet da zudem diese Elemente in der Regel nicht sichtbar bzw. von Interesse sind

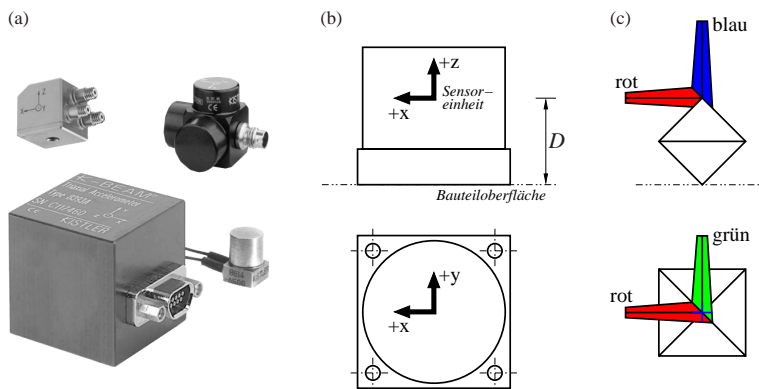


Abbildung 4.11: Vergleich zwischen realen und modellierten Sensoren: (a) Beispiele für reale Sensor-Bauformen; (b) prinzipieller Aufbau eines realen, triaxialen Sensors; (c) graphische Modellrepräsentation.

zusätzlich das orthonormale Achsensystem dieser Messaufnehmer angezeigt. Die Bezeichnung der Achsen ist farblich kodiert: Der positiven x -, y - und z -Achse werden entsprechend die Farben rot, grün und blau zugeordnet, wie der Beschriftung in Abb. 4.11(b) und 4.11(c) zu entnehmen ist. Auf eine aufwändige Darstellung des Sensorkoordinatensystems, zum Beispiel durch dreidimensionale Pfeile, wurde bewusst verzichtet, da dies sowohl die Darstellungsperformanz, als auch die Übersichtlichkeit verschlechtert und zudem durch eine solche Art der Visualisierung keine weitere Information vermittelt wird.

Der Anwender hat die Möglichkeit, mittels eines Mausklicks auf die Bauteiloberfläche den Fußpunkt eines Sensors an der angegebenen Stelle zu platzieren. Der Sensorkörper wird automatisch anhand der Oberflächennormalen und der angewählten Oberflächenseite ausgerichtet. Durch Klicken auf den Sensorkörper können fehlerhafte oder nicht mehr benötigte Sensoren entfernt oder modifiziert werden. Dies schließt auch Änderungen der Eigenschaften – wie zum Beispiel die Masse oder Achsenausrichtung – ein.

Die Ausrichtung der Mess-Sensorik wird in der Regel in globalen Koordinaten angegeben, die relative Orientierung einer Achse orthogonal zur Befestigungsfläche ist jedoch ebenfalls ohne Aufwand möglich. Das Koordinatensystem kann um jede Achse in 45° -Schritten rotiert werden. Eine prinzipiell beliebige Ausrichtung ist zwar mittels einer einfachen Quaternionen-Rotation [Sho85] realisierbar, in der Realität werden Sensoren im Allgemeinen jedoch nicht in beliebigen Ausrichtungen hergestellt, sondern stehen lediglich im gleichen eingeschränkten Winkelraster zur Verfügung. Daher ist die schnelle Einstellmöglichkeit mittels 45° -Drehungen der Angabe beliebiger Winkel mittels eines Eingabefelds vorzuziehen.

Validierung und Korrektur platzierter Sensoren

Die bei Verbindungselementen bereits bewährte Technik, fehlerhafte Elemente mittels einer auffälligen Farbgebung hervorzuheben, kann unverändert bei Sensoren angewandt werden. Abbildung 4.12 zeigt, wie – abhängig vom Fehlerfall – der Sensorkörper entsprechend gefärbt ist.

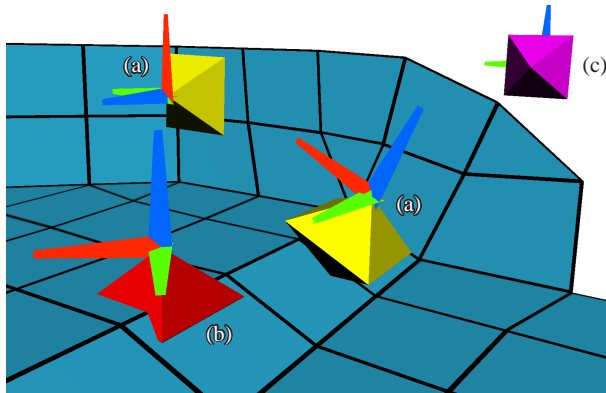


Abbildung 4.12: Visualisierung von Sensorpunkten: (a) korrekt platzierte Sensoren; (b) Sensorkörper (rot) durchdringt die Montagefläche; (c) Sensorpunkt (violett) mit ungültiger Bauteilreferenz.

Beim Einlesen und Validieren bestehender Sensordaten ergibt sich hier zunächst das Problem, dass die Ausrichtung des Sensorkörpers nicht explizit gespeichert wird, sondern lediglich die Ausrichtung und die Ursprungscoordinate der Sensorik-Einheit. Unter Zuhilfenahme der Information, auf welchem Bauteil der Sensor montiert ist und des ebenfalls angegebenen Abstands des Messaufnehmerzentrums zur Oberfläche dieses Bauteils, lässt sich der entsprechende Fußpunkt des Sensors auf dem Bauteil bestimmen. Durch eine hierarchische Suche nach dem nächstgelegenen finiten Element lässt sich diese Bestimmung beschleunigen. Die Verbindungslinie des Sensorfußpunkts zum Ursprung des Sensorik-Koordinatensystems gibt nun die Ausrichtung beziehungsweise die Normale des Sensorkörpers an. Dadurch lassen sich leicht Aussagen über die Einhaltung bestimmter Kriterien treffen – wie zum Beispiel korrekter Abstand zum Bauteil oder planare, orthogonale Montagefläche. Ein Distanzfehler, wie in Abb. 4.12(b), lässt sich daraufhin automatisch durch eine entsprechende Verschiebung entlang der Sensorkörpormalen beheben.

4.3 Modifizieren der Bauteilgeometrie

Eine unzureichend hohe Auflösung oder Ungenauigkeiten bei der Vernetzung können zu Durchdringungen oder Überschneidungen zwischen benachbarten Bauteilen – wie in Abb. 4.13 gezeigt – führen. Ebenso kann es durch den Austausch einer nicht genau eingepassten Variante für ein älteres Bauteil zu solchen Penetrations- und Perforationsproblematiken kommen.

In Abb. 2.15 wurde bereits demonstriert, dass solche Vernetzungsfehler anhand von Abstandsbeachtungen in Verbindung mit gleichseitig orientierten Oberflächennormalen automatisch detektiert werden und mittels eindimensionaler Texturen, welche eine entsprechende Farbskala wiedergeben, hervorgehoben werden können [Som03]. Einfachere Fehlerfälle lassen sich im Allgemeinen automatisch oder halb-automatisch bereinigen, indem der Anwender vorgibt, welcher Bauteilabschnitt die Durchdringung verursacht [Fri04]. Jedoch ist dies nicht immer möglich, da sich bei komplexeren Durchdringungssituationen mit ggf. mehrfach überlappenden Falzen nur schwer eine eindeutige Aussage treffen lässt, welche Komponente lokal perforierend wirkt.

Ein entsprechender Änderungswunsch an einem Bauteil bedeutet in solchen Fällen bei konventioneller Vorgehensweise daher einen langwierigen Umweg über die CAD-Abteilung (siehe auch Abb. 2.1). Kleinere Änderungen können ggf. auch mit Hilfe von Vernetzungswerkzeugen durchgeführt werden, indem die betroffenen Elemente gelöscht werden und neue Flächen erzeugt werden. Bei beiden Ansätzen ist eine erneute Vernetzung der Fahrzeugkomponente involviert, welche – wegen der bei Finite-Element-Modellen lediglich grob gegebenen Abtastung der Oberflächenstruktur – zu den in Abb. 4.13 verdeutlichten Penetrationen und Perforationen führen kann.

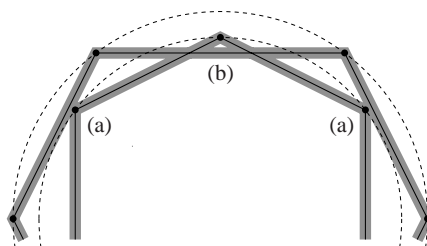


Abbildung 4.13: Bauteilpenetration und -perforation: (a) Die innenliegende Fahrzeugkomponente penetriert das äußere Blechbauteil; (b) Das innere Blech durchstößt die Mittelebene des umgebenden Bauteils. Die Blechstärke ist durch den grauen Bereich symbolisiert, die zu Grunde liegende CAD-Geometrie ist gestrichelt skizziert.

Bereits in früheren Arbeiten wurden daher manuelle Deformationsmöglichkeiten [FE02] untersucht, welche eine benutzerkontrollierte Verformung der Bauteilgeometrie auf Basis des FE-Netztes ermöglichen. Eine direkte Modifikation der FE-Geometrie bietet dabei mehr Potenzial

als die Bereinigung fehlerhafter Bereiche und kann ebenso zur schnellen und unkomplizierten Generierung neuer Bauteilvarianten – z.B. zur Erzeugung von Sicken zur lokalen Erhöhung der Biegesteifigkeit – dienen.

Die erwähnten Modifikationsoperationen sind in der Regel auf einen bestimmten Bereich des Bauteils begrenzt und es bietet sich daher an, keinen globalen Ansatz – ggf. verknüpft mit einer lokalen Steuerung über abstrakte Kontrollvolumen – wie die meisten vorausgegangenen Arbeiten zu verfolgen [FE02, NC99, SP86], sondern dem Anwender entsprechende Methoden zur Verfügung zu stellen, um den Wirkungsbereich seiner Geometriemanipulation einschränken zu können. Dabei hat es sich als nicht zweckmäßig erwiesen, die Auswahl des Modifikationsbereichs an den eigentlichen Manipulationsschritt zu binden, wie dies u.a. in [GP95] vorgeschlagen wird, indem z.B. für eine durchgeführte Manipulationsoperation ein gewisser Aktionsradius definiert werden kann. Bei komplex geformten Bauteilen oder ggf. vielschichtigeren Modifikationsoperationen sind die Grenzen solch einer starren Kopplung schnell erreicht, und das Verhalten ist zudem nicht sehr intuitiv. Stattdessen ist es sinnvoll, die Selektion der Manipulationsregion von der eigentlichen Verformung des gewählten Bereichs loszulösen, indem der Anwender zunächst eine Auswahl an Knoten trifft, welche er dann im anschließenden Schritt modifizieren kann.

4.3.1 Selektieren im dreidimensionalen Raum

Eine Selektion aus räumlich verteilten Knoten kann auf sehr unterschiedliche Arten erfolgen. Die aus der Sicht des Implementierungsaufwands einfachste Methode ist die Bereitstellung eines numerischen Eingabefelds, in welchem die eindeutigen Identifikationsnummern der gewünschten Knoten angegeben werden können. Dies ist zwar – trotz verschiedener Erweiterungen, wie eine Auswahl durch Bereichsangaben oder logische Verknüpfungsoperatoren – eine sehr umständliche Möglichkeit eine Gruppe von Knoten zu selektieren, sie wird jedoch von einigen Ingenieuren unter gewissen Umständen bevorzugt, da sie sich teilweise anhand dieser Knotennummern im Datensatz orientieren. Jedoch mit zunehmender Modellkomplexität und den über die verschiedenen Konstruktionsstadien hinweg stetig wechselnden Knotennummerierungen bestätigt sich die geringe Praxistauglichkeit dieser kryptischen Methode.

Eine weitaus intuitivere Vorgehensweise ist das direkte Anwählen eines Punktes auf der Oberfläche eines Bauteils mit der Maus. Diese Wahlmöglichkeit ist daher auch in vielen kommerziell erhältlichen FE-Modellierungsprogrammen realisiert. Möchte man allerdings eine größere Menge an Knoten selektieren, so erweist sich auch diese Methode als äußerst mühsam. Daher wird von einigen Anwendungen zusätzlich eine BereichsSelektion angeboten, welche dem Benutzer erlaubt, durch Aufziehen einer rechteckigen Region mit der Maus die innerhalb des Rechtecks befindlichen Knoten auszuwählen. Somit können mehrere Knoten auf einmal selektiert werden, jedoch ist die Auswahlsschablone im Allgemeinen immer auf Rechtecksformen begrenzt.

Freihandselektionen, wie sie aus zweidimensionalen Zeichen- bzw. Malprogrammen bekannt sind, bieten hierfür eine leistungsfähige Alternative. Dabei kann der Benutzer mit der Maus bei gedrückter Maustaste eine zweidimensionale Linie beschreiben. Diese Freihandlinie beschreibt hierbei – aufgrund der perspektivischen Sicht – die Oberfläche eines pyramidenartigen Gebildes

in der dreidimensionalen Szene. Um eine geschlossene Auswahlkurve zu garantieren wird dabei stets der Startpunkt mit dem momentanen Endpunkt der Linie verbunden, wie in Abb. 4.14(a) zu sehen. Um ein ständiges Neuzeichnen der Szene während der Auswahlinteraktion zu vermeiden, wird die Umrandungslinie mittels einer bitweisen Exklusiv-Oder-Verknüpfung in den Bildschirmspeicher geschrieben. Durch ein erneutes Zeichnen der Linie kann diese somit aufgrund der angewandten Bitoperation wieder entfernt werden und die Darstellung der Umrandung entsprechend dem aktuellen Zustand angepasst werden.

Trifft der Anwender mit der Freihandlinie ein Bauteil, so wird die getroffene Auswahl auf die Knoten dieses Bauteils begrenzt. Somit ist ohne zusätzlichen Interaktionsaufwand eine automatische Begrenzung der Selektion auf das zu modifizierende Bauteil möglich.

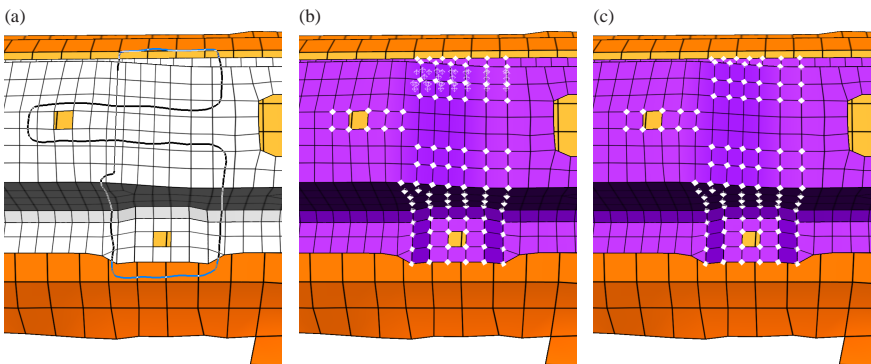


Abbildung 4.14: Selektion mittels einer Freihandlinie: (a) Die Freihandlinie ist stets durch Verbindung von Start- und Endpunkt (dünne Linie) geschlossen; (b) Auswahl aller in der Umrandung liegenden Knoten, verdeckte Knoten sind als Drahtgitter-Oктаeder dargestellt; (c) Selektion mit zusätzlicher Filterung der verdeckten Knoten.

Lässt der Anwender die Maustaste los, so werden sämtliche Knoten des gewünschten Bauteils, welche innerhalb des beschriebenen Pyramidenstumpfs liegen, selektiert und entsprechend Abb. 4.14(b) bzw. 4.14(c) durch einen weißen Oktaeder markiert. Verdeckt liegende Knoten werden dabei je nach Vorgabe transparent dargestellt oder in Drahtgitteransicht gezeichnet, so dass der Anwender aus jedem Blickwinkel den Überblick über die gesamte Selektion behält.

Die Entscheidung, ob ein Knoten eines Bauteils innerhalb oder außerhalb des Selektionsbereichs liegt, muss nicht im dreidimensionalen Raum getroffen werden. Stattdessen genügt es, die Koordinate eines Knotens auf die Bildfläche zu projizieren und zu testen, ob der projizierte Punkt innerhalb der Freihandlinie liegt. Die Beurteilung, ob ein Punkt innerhalb oder außerhalb der Berandung liegt kann sehr schnell erfolgen, indem eine beliebig orientierte Halbgerade von diesem

Punkt aus definiert wird und die Anzahl der Schnittpunkte mit der Selektionsberandung gezählt wird [Mor90]. Bei einer geraden Anzahl befindet sich der Knoten außerhalb, bei einer ungeraden Menge an Schnittpunkten befindet sich der Punkt innerhalb des Auswahlgebiets und wird entsprechend markiert (Abb. 4.14(c)).

Dabei muss der Test der Halbgeraden auf Schnittpunkte potenziell mit allen Abschnitten der Freihandlinie durchgeführt werden. Um dies zu beschleunigen, wird zunächst die Halbgerade als horizontal verlaufend definiert. Außerdem werden alle Segmente der Freihandlinie auf jeder Pixelzeile, die sie kreuzen, abgetastet (Abb. 4.15(a)) und jeweils ein zusätzlicher Punkt am Schnittpunkt mit der entsprechenden Zeile erzeugt. Die nun kürzer segmentierten Linienstücke – in Abb. 4.15(b) exemplarisch skizziert – werden primär nach ihrer kleineren y-Koordinate und sekundär nach der zugehörigen x-Koordinate sortiert (Abb. 4.15(c)). Zur Schnittpunktuntersuchung eines projizierten Knotens mit der Selektionslinie kann nun anhand der abgerundeten y-Koordinate des Punktes sehr schnell durch eine binäre Suche auf die ggf. die Halbgerade schneidenden Linienabschnitte zugegriffen werden.

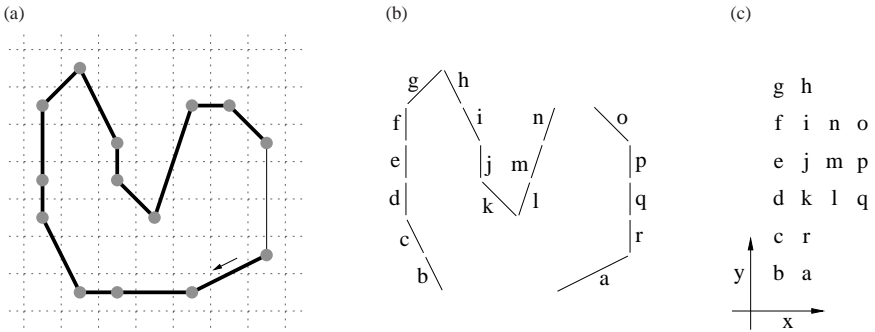


Abbildung 4.15: Sortierung der Berandungskanten einer Freihandselektion: (a) Durch Events übermittelte Mauspositionen (graue Punkte) definieren eine Freihandlinie im Pixelraster; (b) Horizontal verlaufende Abschnitte werden aussortiert, und mehrere Pixelzeilen überspannende Linienabschnitte werden in entsprechend viele kürzere Segmente zerteilt; (c) Sortierung der Liniensegmente anhand der kleineren y- und x-Koordinate ergibt die Liste b,a,c,r,d,k,l,q,e,j,m,p,f,i,n,o,g,h.

Der Anwender kann entscheiden, ob er alle Knoten innerhalb der Freihandlinie auswählen möchte oder nur die in der momentanen Kameraperspektive sichtbaren. Letzteres kann realisiert werden, indem man für jeden potenziellen Selektionskandidaten einen Sichtstrahl in die Szene schießt, welcher prüft, ob der Knoten durch eines oder mehrere Objekte verdeckt ist. Dieses Vorgehen kann mit Hilfe der Graphikhardware beschleunigt werden, indem man das Fahrzeugmodell ohne Beleuchtungsberechnung zeichnet und jedem finiten Element eine eindeutige Farbe

zuordnet. Aus der Farbinformation in den Pixeln in der Umgebung der projizierten Knotenposition lassen sich die damit verknüpften Elemente ableiten. Befindet sich keines dieser Elemente in der näheren Umgebung des Knotens, so ist dieser verdeckt. Andernfalls kann die aufwändigere aber präzisere Verdeckungsanalyse mittels Sichtstrahlen durchgeführt werden.

Eine komfortablere Selektionsmethode bietet ein merkmalsbasierter Ansatz. Dieser ist vor allem dazu geeignet, sehr schnell große, zusammenhängende Flächen auszuwählen. Ausgehend von einer mit der Maus angewählten Position werden dabei alle Knoten markiert, welche innerhalb derselben, durch Merkmalslinien begrenzten, Region eines Bauteils liegen. Diese abgrenzenden Merkmale werden nach denselben Methoden ermittelt wie sie in Kapitel 3.1.2 vorgestellt wurden. In vielen Fällen liefert diese Art der Auswahl bereits genau das vom Anwender gewünschte Ergebnis, da sehr häufig an solch zusammenhängenden Bereichen – wie in Abb. 4.16 exemplarisch verdeutlicht – eine gemeinsame Modifikationsoperation vorgenommen wird.

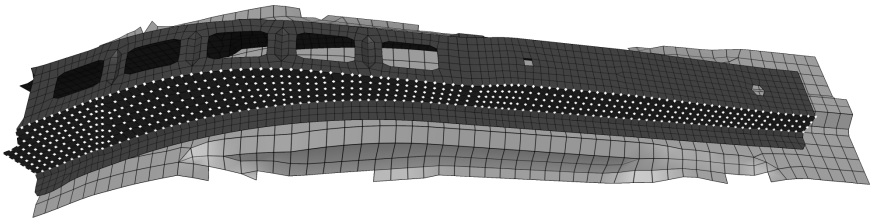


Abbildung 4.16: Merkmalsbasierte Selektion: Auf einer Bauteiloberfläche werden bis zu den umgebenden bauteiltypischen Merkmalslinien alle Knoten ausgewählt.

Die Selektionsmethoden können ohne weiteres kombiniert werden. So kann der Anwender z.B. zunächst eine großflächige und schnelle Auswahl mittels einer merkmalsbasierten Selektion treffen und anschließend mit einer Freihandlinie weitere Knoten hinzufügen oder aus der bestehenden Auswahlmenge entfernen. Ebenso ist auch eine zusätzliche Verknüpfung mit der eingabebasierten Selektion möglich.

4.3.2 Editieren im dreidimensionalen Raum

Im Umgang mit dreidimensionalen Objekten ist die Durchführung selbst einfachster Modifikationsoperationen – wie einer Parallelverschiebung – erschwert, da dem Anwender in der Regel lediglich zweidimensionale Eingabe- und Ausgabegeräte zur Verfügung stehen. Aufgrund der fehlenden Dimension ist die direkte Abbildung einer zweidimensionalen Mausebewegung auf eine dreidimensionale Verschiebung nicht möglich. Soll eine Manipulation im dreidimensionalen Raum⁶ erfolgen, so muss diese Operation in zwei (bei höherer Dimension entsprechend mehrere)

⁶sollen zusätzlich Rotationen bzw. Torsionen adressiert werden, so erhöht sich die Dimension auf bis zu sechs Freiheitsgrade

Modifikationen niedrigerer Dimension, d.h. zwei- und eindimensionale Bewegungen, aufgespalten werden.

Zur Steuerung der einzelnen Bewegungskomponenten hat sich die Verwendung eines dreidimensionalen Widgets bewährt. Der Einsatz solcher Manipulator-Widgets zur Kontrolle verschiedener Manipulationen hat den Vorteil, dass dem Anwender u.U. das Prinzip von anderen Applikationen geläufig ist. Der bekannteste Vertreter solcher Manipulatoren ist die Szenengraphschnittstelle *Inventor* [Wer94], bei welchem verschiedene Widgets zur Steuerung der Kamera oder zur Interaktion mit Szenenobjekten eingesetzt werden. Daneben existieren zahlreiche Anwendungen und Erweiterungen des dreidimensionalen Widget-Konzepts [CSH⁺92, GP95] und Programmgerüste [DH97] zur Realisierung komplexer Interaktionsvarianten. Die Anwendungsgebiete reichen dabei von einfachen Objektinteraktionen, wie z.B. Platzieren und Orientieren eines Körpers, bis zur Modellierung animierter Objekte.

Für die direkte Interaktion mit FE-Flächengeometrien wurde ein Widget entwickelt, welches sowohl zweidimensionale Verschiebungen auf der Bauteiloberfläche als auch eindimensionale Bewegungen entlang der lokalen Oberflächennormalen erlaubt. Letzteres realisiert der Anwender, indem er an dem die Normale symbolisierenden Pfeil des in Abb. 4.17(a) abgebildeten Widgets mit der Maus zieht. Translationen auf der Bauteilfläche werden durch entsprechendes Ziehen an der dargestellten Grundfläche des Widgets erreicht. Durch Kombination der beiden Steuermöglichkeiten lassen sich beliebige Translationen im dreidimensionalen Raum erzielen. Um dem Benutzer Rückkopplung darüber zu geben, welche Möglichkeit der Verschiebung er gewählt hat, wird für die Dauer des Umherbewegens mit der Maus der entsprechende Teil des Widgets grün eingefärbt (in Abb. 4.17(b) dementsprechend hell dargestellt). Parallel dazu wird die modifizierte Bauteilgeometrie in Drahtgitterdarstellung eingezeichnet. Wie man anhand die-

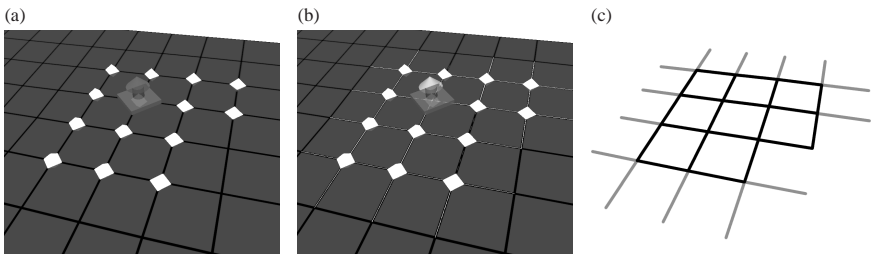


Abbildung 4.17: Dreidimensionales Translations-Widget: (a) Durch die halbtransparente Darstellung der Pfeil- und Flächenkomponente des Widgets wird die darunterliegende Geometrie nicht verdeckt; (b) Beim Anwählen einer Widgetkomponente (hier: Normalenpfeil) wird diese grün (hier: hell) dargestellt und eine Drahtgitteransicht der bewegten Elementkanten eingeblendet; (c) Die Darstellung der Elementkanten geschieht über Vierecksprimitive (schwarz) wo möglich und Linienprimitive (grau) wo nötig.

ser weißen Linien in Abb. 4.18(b) erkennen kann, wird dabei für verdeckt liegende Regionen eine gestrichelte Darstellung gewählt. Um maximale Performanz in der Darstellung dieses Gitters zu gewährleisten, werden – wo möglich, d.h. für vollständig selektierte Elemente – Vierecks- bzw. Dreiecksprimitive im Linienmodus gezeichnet. Somit müssen lediglich die vereinzelt vorhandenen freistehenden Elementkanten als tatsächliche Linienprimitive gezeichnet werden, wie Abb. 4.17(c) verdeutlicht. Erst nach Abschluss der Benutzerinteraktion erfolgt eine Aktualisierung der schattierten Darstellung des Bauteils, welche eine Neuberechnung der Knotennormalen und einen entsprechenden Neuaufbau der Merkmalslinien beinhaltet.

Eine gemeinsame Parallelverschiebung ist die einfachste Editieroperation auf einer Menge an selektierten Knoten. Hierbei wird auf die Position aller ausgewählten Knoten derselbe Translationsvektor addiert, welcher aus der Verschiebung des Widgets gewonnen wird. In diesem speziellen Fall ist die Position des Widgets irrelevant, es kann sich somit prinzipiell an jeder Position der Szene befinden. Im Sinne einer direkten Kopplung von Aktion und Reaktion wird die Positionswahl des Widgets jedoch auf die Knotenkoordinaten der Selektionsmenge beschränkt. Vor einer Modifikationsoperation kann dazu durch einen Mausklick in die Nähe eines Knotens das Widget auf dessen Koordinaten eingerastet werden. Abbildung 4.18 zeigt zwei unterschiedliche Positionierungen der Geometrie bei Anwendung einer Parallelverschiebung.

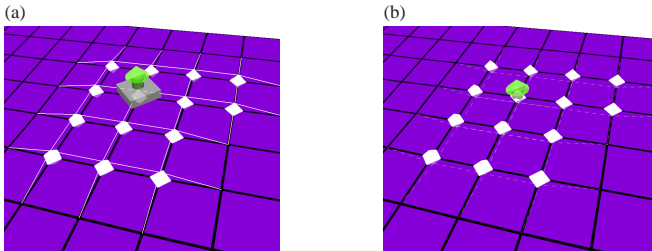


Abbildung 4.18: Gemeinsame Parallelverschiebung einer Menge von Knoten: (a) Verschiebung in lokaler Normalenrichtung nach oben; (b) Translation in lokaler Normalenrichtung nach unten; die durch das Bauteil verdeckte Drahtgitteransicht der modifizierten Geometrie ist gestrichelt dargestellt.

In vielen Fällen bietet solch eine starre Parallelverschiebung nicht genügend Flexibilität. Eine Erweiterung dieses Ansatzes ermöglicht deutlich komplexere Deformationsoperationen und beinhaltet die gemeinsame Parallelverschiebung als Spezialfall. Die intuitive Interaktion durch Ziehen an Teilen eines Widgets wird dabei wie bisher beibehalten, jedoch werden nicht mehr alle Knoten um den selben Betrag entlang des Translationsvektors verschoben. Stattdessen wird – ähnlich wie bei einem unabhängig davon entwickelten Ansatz [BKS03] – für jeden Knoten ein Wichtungsfaktor eingeführt, welcher die Lage dieses Punktes innerhalb des Selektionsgebiets in Bezug zum Angriffspunkt des Widgets beschreibt.

Sei $d_{i,\text{Rand}}$ der kleinste geodätische – d.h. auf der Bauteiloberfläche gemessene – Abstand eines Knotens i zur Berandung des Selektionsgebiets und $d_{i,\text{Zentrum}}$ die geodätische Distanz dieses Punktes i zum Angriffszentrum des Widgets, so lässt sich der Wichtungsfaktor w_i des Knotens wie folgt berechnen:

$$w_i = \frac{d_{i,\text{Rand}}}{d_{i,\text{Zentrum}} + d_{i,\text{Rand}}} \quad (4.3)$$

Aufgrund der Ausgangssituation befindet sich der Wertebereich aller Faktoren garantiert innerhalb des abgeschlossenen Intervalls $[0; 1]$. Werden die Gewichte direkt dazu verwendet, den anhand der Verschiebung des Widgetpfeils gewonnenen Translationsvektor zu skalieren, so erhält man – je nach dem Grundriss des Selektionsgebiets – eine pyramiden- oder kegelähnliche Deformation, wie in Abb. 4.19(a) gezeigt. Bei Verschiebungen innerhalb der Bauteilfläche hingegen findet solch eine spitz zulaufende Verschiebeoperation in der Regel keine Anwendung (Abb. 4.19(b)).

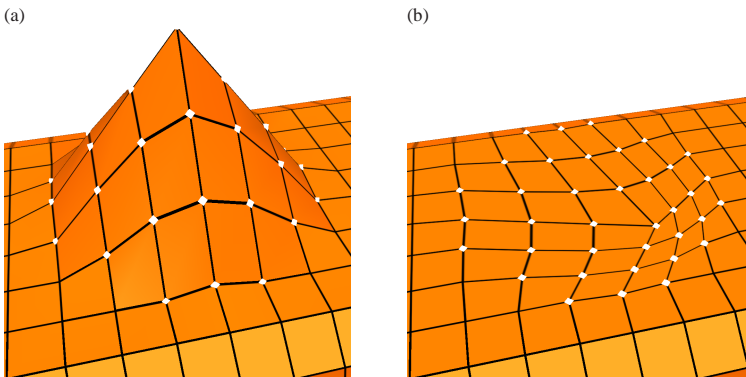


Abbildung 4.19: Kegelförmige Deformation durch direkte Anwendung der Wichtungsfaktoren w_i : (a) Konische Verformung durch Ziehen entlang der Flächennormalen im Zentrum des Selektionsgebiets; (b) Konische Deformation in der Bauteilebene erzeugt i.A. keine sinnvollen FE-Gitterstrukturen.

Die eigentliche Attraktivität dieser Wichtungsmethode liegt jedoch in der Anwendung beliebiger Transferfunktionen, welche es ermöglichen, der Verformungsoperation im Prinzip beliebige Gestalt zu verleihen. So bewirkt z.B. eine entsprechende Quadrierung⁷ der Wichtungsfaktoren eine paraboloidale Deformation und kann – da diese Art der Modifikationsfunktion im Zentrum ($w \rightarrow 1$) nicht spitz zuläuft – auch als sinnvolle Verschiebeoperation innerhalb von FE-Flächen

⁷ $f(w_i) = 1 - (1 - w_i)^2 = 2w_i - w_i^2$

dienen. Um Übergänge im Bereich des Selektionsrands zu den umgebenden Elementen stetig zu gestalten, können Gauss-ähnliche Transferfunktionen eingesetzt werden. Für die Erzielung sinnvoller Verformungen sollten lediglich zwei Bedingungen bei der Wahl geeigneter Transferfunktionen berücksichtigt werden. Zum einen sollte das Funktionsintervall $[0; 1]$ ebenfalls auf einen Wertebereich $[0; 1]$ abgebildet werden, und zum anderen sollte die Transferfunktion monoton steigend verlaufen. Prinzipiell sind zwar andersartige Funktionen denkbar, diese haben jedoch eher einen exotischen Anwendungscharakter.

Hierbei muss die Transferfunktion nicht zwingend rotationssymmetrisch zum Interaktionszentrum, d.h. dem Translationswidet, verlaufen. Stattdessen ist es ebenso möglich, entlang zweier orthogonaler Achsen – welche wiederum senkrecht auf dem Translationsvektor stehen – unterschiedliche Transferfunktionen anzuwenden. Die Ausrichtung der beiden Achsen kann dabei mit der Maus beeinflusst werden, indem man einen Kreisbogen um das Widget beschreibt und dadurch das Achsenkreuz um den Translationsvektor dreht. Für den Anwender ergibt sich dabei der Eindruck, an einem virtuellen Zylinder zu drehen, ähnlich der häufig eingesetzten, intuitiven Rotation von Objekten durch Drehen an einer virtuellen Kugel [CMS88]. Abbildung 4.20 zeigt ein Beispiel für solch eine zweidimensionale Transferfunktion, bei dem in eine Richtung die Gewichte auf konstant 1 gesetzt werden und entlang der zweiten Achse eine Gauss-ähnliche Transferfunktion angewandt wird.

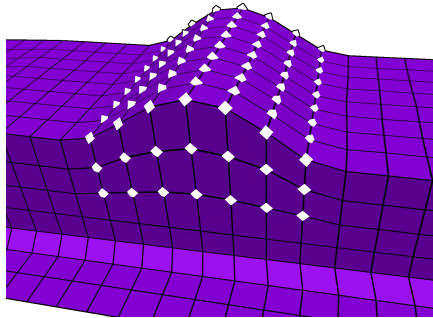


Abbildung 4.20: Gleichmäßige Ausbeulung durch Anwenden einer zweidimensionalen Transferfunktion bei der Modifikation.

Biegeoperationen sind ebenfalls möglich, indem der Benutzer zunächst eine Rotationsachse definiert, welche entweder manuell mittels numerischer Eingabe eines Vektors oder bei der Wahl eines Punktes auf der Bauteiloberfläche durch die lokale Flächennormale festgelegt wird. Die Rotationsachse wird daraufhin durch den Pfeil des Interaktionswidgets repräsentiert. Der Rotationswinkel wird – analog zur im letzten Absatz erläuterten Rotationsinteraktion – durch Beschreiben eines entsprechend weit gezogenen Kreisbogens definiert. Die Wichtungsfaktoren werden dabei jedoch nicht durch geodätische Abstände, sondern durch die in der Rotationsebene euklidisch gemessenen Distanzen ermittelt und legen fest, welcher Anteil des Rotationswinkels für die

Drehbewegung der entsprechenden Punkte berücksichtigt wird (siehe auch [Zö04]). Da in den Randbereichen eines Selektionsgebiets dieser Anteil gegen 0 geht, findet dort keine Verschiebung der Knoten statt, so dass ein stetiger Übergang zu den angrenzenden Elementen garantiert ist (Abb. 4.21(a)). Durch Anwenden einer entsprechenden Transferfunktion können zudem glatte, d.h. G^1 -stetige Übergänge erzeugt werden, wie in Abb. 4.21(b) demonstriert. Analog zur Anwendung der Transferfunktionen bei translatorischen Modifikationen können auch hier zwei unterschiedliche Abbildungsfunktionen zum Einsatz kommen, wovon eine in Richtung der Rotationsachse und die andere senkrecht dazu angewandt wird.

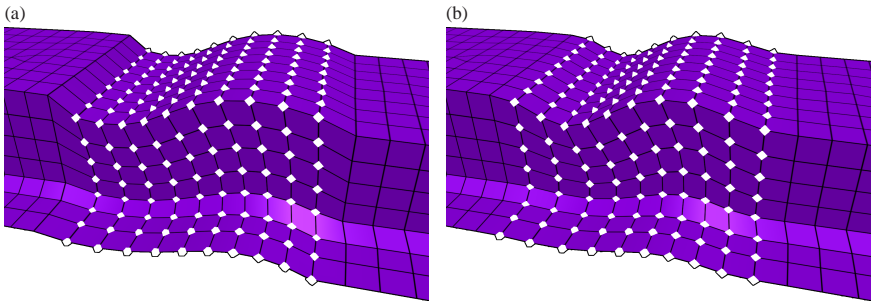


Abbildung 4.21: Rotationsdeformationen: (a) Ohne Anwendung einer Transferfunktion ist lediglich ein stetiger Übergang zu den angrenzenden Elementen außerhalb des Selektionsgebiets garantiert; (b) Durch eine geeignet gewählte Transferfunktion lassen sich weiche Übergänge erzielen.

An freien Enden, d.h. an Stellen, an denen der Selektionsrand und der Bauteilrand identisch sind, ergibt die Einhaltung eines stetigen Übergangs keinen Sinn. Ein entsprechendes Fixieren der Randpunkte ist in der Regel sogar unerwünscht. Stattdessen sollen solche freien Ränder im Allgemeinen eine ungedämpfte Verschiebungs- bzw. Drehwirkung erfahren. Daher werden im Bereich zwischen Angriffspunkt des Widgets bzw. ausgehend von der Drehachse bis zur betreffenden Randlinie⁸ sämtliche Wichtungsterme auf 1 gesetzt. Auf diese Weise ist ein Umbiegen des Bauteils wie in Abb. 4.22(b) oder eine Verlängerung z.B. eines Trägers (Abb. 4.22(c) und 4.22(d) sehr einfach realisierbar. Soll hingegen ein Bauteilrand nicht verändert werden, so dürfen dementsprechend die betreffenden Randknoten nicht selektiert werden.

Zur Feinabstimmung der Verschiebungsbewegungen könnten Schattenwürfe des dreidimensionalen Widgets dienen, wie in [HZR⁺92] vorgeschlagen. Dort werden Schatten eines Interaktionswidgets auf drei Ebenen geworfen und der Anwender kann mit diesen statt dem dreidimensionalen Widget agieren. Da die Bewegungen somit auf einer ebenen Fläche erfolgen, ist

⁸d.h. für alle Knoten, bei denen die kürzeste Entfernung zu einem bauteilinneren Selektionsrand größer ist als die Distanz zwischen Angriffspunkt und innerem Selektionsrand

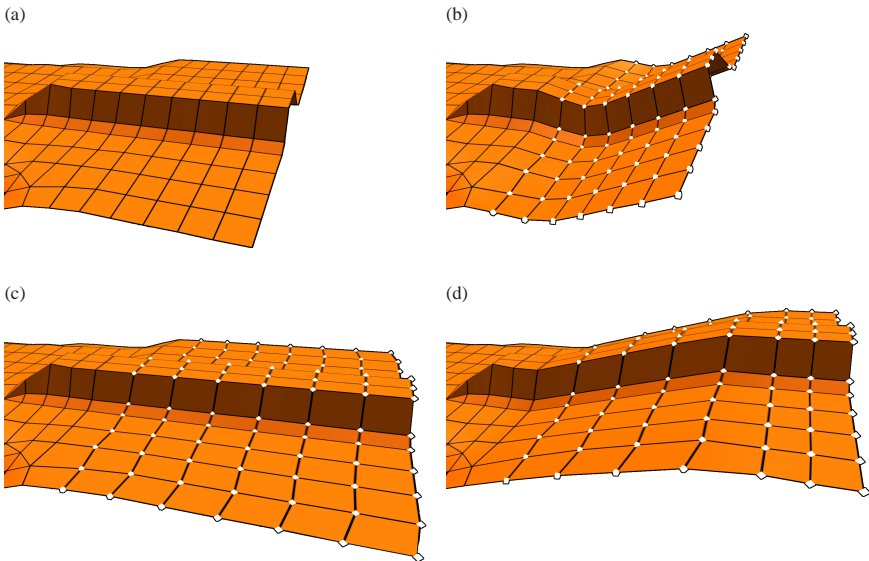


Abbildung 4.22: Unterschiedliche Deformationsoperationen an einem freien Bauteilende: (a) unverformtes FE-Netz; (b) Umbiegen durch Rotation; (c) ebene Verlängerung; (d) Verlängerung mit senkrechtem Versatz.

eine Steuerung mittels eines zweidimensionalen Eingabegeräts entsprechend einfacher und ggf. exakter. Jedoch würden zusätzlich eingeblendete Ebenen die Sicht auf dahinter liegende Komponenten verdecken. Als Kompromiss und um einen verbesserten Eindruck über die momentane Lage des dreidimensionalen Widgets relativ zur ursprünglichen Bauteiloberfläche zu vermitteln, könnte das Widget einen Schatten auf diese Oberfläche werfen. Trotz einer einfachen Möglichkeit zur hardware-nahen Implementierung eines solchen Schattens [Hei91, EK02] ist dieser eher verwirrend und im Vergleich dazu der Nutzen durch die zusätzliche Information gering bzw. nur bei ungünstigen Kameraperspektiven hilfreich. Zudem ist die Möglichkeit eines weiteren, auf die Bauteiloberfläche limitierten Interaktionsmechanismus – neben der Widgetgrundfläche, welche ebenfalls Bewegungen in dieser Ebene erlaubt – nicht sinnvoll. Wünscht der Anwender größtmögliche Genauigkeit, so ist hierfür eine direkte numerische Eingabe des zentralen Deformationsvektors bzw. alternativ dazu eine zusätzliche Skalierung der durchgeführten Freihandverschiebung das zweckmäßigste Mittel.

4.3.3 Interaktive Netzvalidierung

An den oben abgebildeten Beispielen ist zu erkennen, dass die einzelnen finiten Elemente, aus denen die verschiedenen Komponenten eines Fahrzeugs modelliert werden, durch große Modifikationen teilweise stark verzerrt werden. Diese Elementdeformationen sind bis zu einem gewissen Grad unkritisch, jedoch führen sie zu einem zunehmend ungenauen Simulationsergebnis. Dies kann zum einen zurückgeführt werden auf – im Vergleich zu quadratisch geformten Elementen – schlechter konditionierte FE-Matrizen und eine damit verbundene instabile numerische Lösung des FE-Gleichungssystems. Zum anderen wird, je nach Verzerrungsart, die geometrische Struktur eines Bauteils nicht mehr hinreichend genau abgetastet. Im Gegensatz dazu kann ebenso eine Überabtastung erfolgen, da die damit verknüpfte minimale Elementkantenlänge einen entscheidenden Einfluss auf den für aussagekräftige Simulationsergebnisse benötigten zeitlichen maximalen Abstand zwischen zwei Simulationszuständen hat. Dieses Zeitschrittkriterium [CFL28] wird neben anderen *Bedingungen für ein zuverlässiges Simulationsergebnis* im folgenden kurz erläutert.

Minimale Kantenlängen sollten nicht unterschritten werden, da sonst aufgrund des simulations-technisch vorgegebenen, materialabhängigen Zeitschrittkriteriums

$$\Delta t < \frac{l}{c} \quad (4.4)$$

die zeitliche Schrittweite zwischen aufeinanderfolgenden Crashzuständen verkürzt werden muss. Dabei ist die zeitliche Mindestschrittweite Δt abhängig von der minimalen Elementkantenlänge l und der Schallgeschwindigkeit c des eingesetzten Werkstoffes. Bei zu kurzen Kantenlängen muss daher häufiger der Momentanzustand des Fahrzeugs berechnet werden, wodurch die Rechenzeit für die Crashtsimulation erhöht wird.

Die *Elementinnenwinkel* sollten innerhalb gewisser Grenzen liegen. Sind sie zu groß, so fällt der Diskretisierungsfehler entsprechend aus, was eine schlechte Interpolation der Ableitungen nach sich zieht. Sind die Winkel zu klein, so neigt die der FE-Simulation zu Grunde liegende Steifigkeitsmatrix zu einer schlechten Konditionierung.

Das *Verhältnis der Elementkantenlänge* sollte sich ebenfalls in einem bestimmten Rahmen bewegen. Im Allgemeinen ist ein Verhältnis bis 2:1 auch in Regionen, welche starken Belastungen ausgesetzt werden, unkritisch. Noch höhere Längenverhältnisse führen – je nach Belastungsfall – zu zunehmend inakkuraten Simulationsergebnissen.

Nicht planare Viereckselemente liefern mit zunehmender Verkipfung unzuverlässige Resultate. Dieses sogenannte „*Warping*“ wird ermittelt, indem das Element entlang seiner beiden Diagonalen in jeweils zwei Dreiecke aufgespalten wird und die Winkel der Dreiecke zueinander bestimmt werden. Dabei ist zu beachten, dass bei nicht gleichseitigen Elementen das „*Warping*“ um die beiden Diagonalen unterschiedlich stark ausgeprägt ist.

Des Weiteren sind *viereckige Schalenelemente gegenüber Dreiecken vorzuziehen*, da Dreiecksnetze im Vergleich zu vierecksbasierten Netzen häufiger zu numerischen Instabilitäten neigen und das Verformungsverhalten i.A. steifer ausfällt.

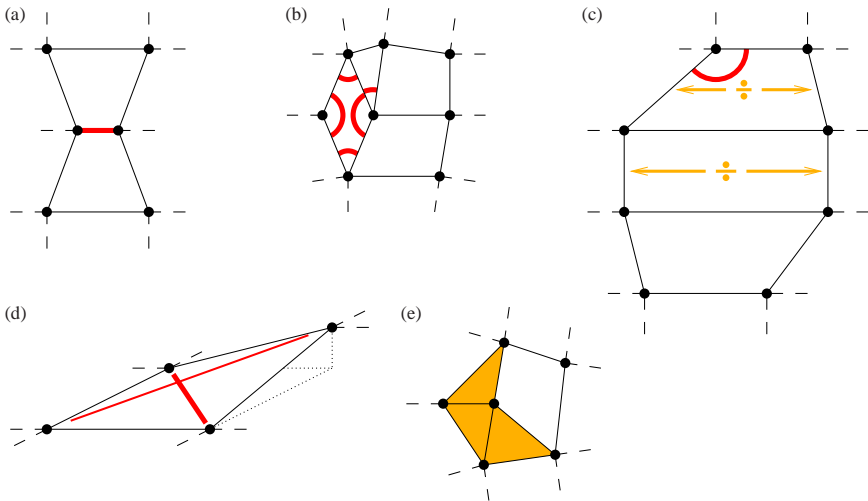


Abbildung 4.23: Markierung fehlerhafter Elemente durch Glyphen: (a) Elementkante zu kurz; (b) Innenwinkel zu groß bzw. zu klein; (c) ungünstiges Kantenverhältnis (hier: größer 2:1), gepaart mit einem zu großen Innenwinkel; (d) unebenes Element bzw. „Warping“ – die Diagonale, um welche der größere Verkippungswinkel auftritt, wird dicker dargestellt; (e) Der gehäufte Einsatz von Dreiecken sollte vermieden werden.

Bereits während einer benutzergesteuerten Modifikation werden die betroffenen Elemente auf eine Verletzung der aufgeführten Qualitätsbedingungen hin untersucht. Die Validierung erfolgt dabei elementweise mit Ausnahme der Überprüfung auf die Einhaltung der minimalen Elementkantenlänge, welche kantenweise durchgeführt wird. Fehlerhafte Elemente bzw. Kanten werden daraufhin mit entsprechenden Glyphen markiert und geben dem Anwender bei der Interaktion eine instantane Rückkopplung über die momentane Qualität des Netzes. Eine Übersicht über die einzelnen Glyphen vermittelt Abb. 4.23. Je nach Zustand des Netzes können sich dabei ggf. auch mehrere Fehlerfälle überlagern, so dass entsprechend mehrere Glyphen im selben Element angezeigt werden (siehe Abb. 4.23(b) bzw. 4.23(c)).

Neben der instantanen Netzvalidierung während Modifikationsoperationen ist es ebenfalls möglich, bestehende Netze auf Fehler hin zu überprüfen. Abbildung 4.24 zeigt ein Beispiel für ein fehlerhaftes FE-Netz, bei welchem unterschiedliche Kombinationen der in Abb. 4.23 skizzierten Fehlerfälle auftreten. Der Anwender kann dann entsprechend darauf reagieren und die Gestalt der fehlerbehafteten Elemente entweder manuell korrigieren oder auf einen der im folgenden Abschnitt präsentierten Algorithmen zurückgreifen.

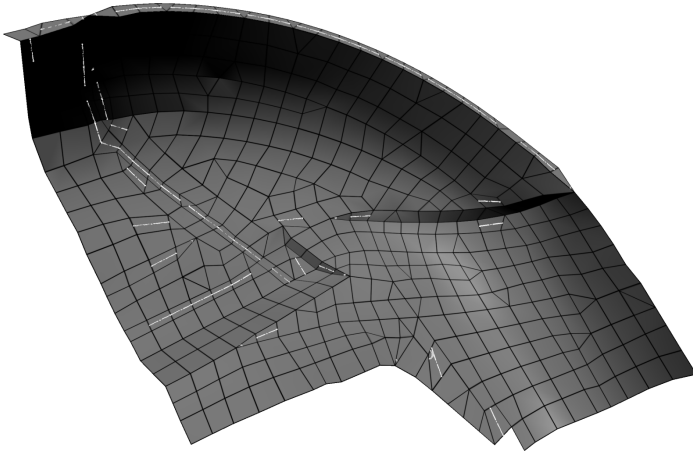


Abbildung 4.24: Glyphenbasierte Fehleranzeige an einer Teilkomponente aus dem Radhaus eines Fahrzeugs.

4.3.4 Netzoptimierung

Zur lokalen Optimierung eines ggf. mit simulationsuntauglichen Elementen behafteten FE-Netzes können unterschiedliche automatisierte Verfahren eingesetzt werden. Häufig werden verschiedene Glättungsalgorithmen angewandt, welche die Knotenkoordinaten eines Netzes leicht verändern und dadurch versuchen, möglichst gleichseitige Elemente zu erhalten. Ein Großteil dieser Relaxationsalgorithmen basiert auf einem iterativen Prozess, bei denen ein Ausgleich zwischen den Kantenlängen lokal benachbarter Elemente gesucht wird. Bekannte Vertreter dieser Algorithmen sind Laplace-Glättungsverfahren und optimierungsbasierte Ansätze, welche spezielle Verzerrungsmetriken bei der Anpassung der Knotenkoordinaten mit in Betracht ziehen. Davon abgeleitet existieren mehrere Ansätze, welche diese beiden Methoden kombinieren [Fre97, ABE97, CTS98].

Eine alternative Vorgehensweise dazu bieten physikalisch basierte Glättungsverfahren [LMZ86], bei welchen die Elementkanten als Federn und die FE-Knoten als Massepunkte betrachtet werden. Unter der Voraussetzung, dass alle Federn dieselbe Ruhelänge besitzen, ergibt sich im energetisch günstigsten bzw. Ruhezustand dieses Feder-Masse-Modells ebenfalls ein relaxiertes FE-Gitter mit möglichst gleichseitigen Elementkanten.

Durch solche Glättungsverfahren verschwinden jedoch auch wichtige Kantenmerkmale der FE-Bauteile. Daher wurden in der Arbeitsgruppe die Glättungsverfahren – und hier insbesondere der physikalisch basierte Ansatz – insofern erweitert, dass scharfe und bauteilbegrenzende Kanten erhalten werden, indem die Bewegungsfreiheit der Knoten auf den Verlauf der Kantenlinie

eingeschränkt ist [BRE04]. Hierbei finden die in Kapitel 3.1.2 vorgestellten Kantendetektionsalgorithmen erneut Verwendung. Außerdem werden Viereckselemente durch zusätzliche, entlang ihrer Diagonalen eingefügte Federn in eine vorzugsweise rechtwinklige Form gelenkt. Um eine möglichst geringe Abweichung von der Originaloberfläche zu gewährleisten, werden abschließend die verschobenen Knoten auf die Ausgangsfläche projiziert.

Abbildung 4.25 demonstriert, wie durch solche Relaxationsmethoden eine deutliche Verbesserung der Netzqualität erzielt werden kann. Im Falle von großen Deformationen aufgrund umfassender Modifikationsoperationen durch den Benutzer ist eine Entzerrung der Elemente mittels Relaxation nicht immer vollständig möglich. Diese Situation tritt vor allem bei starker Verlängerung eines Bauteils oder beim Anlegen einer tiefen Sicke ein. Daher muss in den betroffenen Gebieten eine Neuvernetzung der Oberfläche durchgeführt werden. Um den ggf. langwierigen Umweg über die u.U. zudem noch nicht existente CAD-Geometrie zu vermeiden, wird die Vernetzung direkt auf Basis des bestehenden Netzes durchgeführt. Dabei werden zunächst nur die am stärksten verzerrten Elemente betrachtet, und eine dem Fehlerfall entsprechend gewählte lokale Netztopologieänderung auf die direkte Nachbarschaft der betroffenen Elemente beschränkt. Auf diese Weise ist garantiert, dass die notwendigen FE-Vernetzungsoperationen auf einen sehr kleinen Bereich eingegrenzt werden und die Konnektivität des Ausgangsnetzes möglichst beibehalten wird.

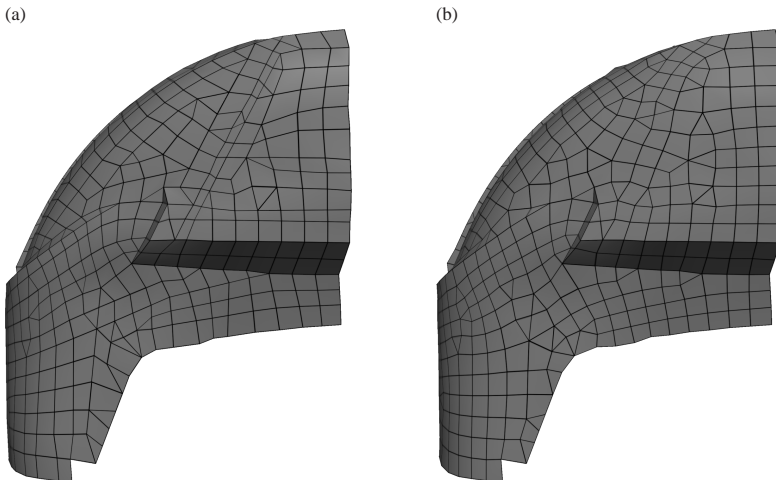


Abbildung 4.25: Relaxation eines FE-Netzes: (a) Ausgangsnetz mit verzerrten Elementen; (b) Geglättetes Netz durch Anwendung eines Relaxationsalgorithmus, der merkmalsgebende Kanten bewahrt.

In Abb. 4.27 sind die für die Fehlerfälle aus Abb. 4.23 gewählten lokalen Vernetzungsmodifikationen skizziert. Dabei stellt sich die vermeintlich einfache Operation aus Abb. 4.27(a), welche eine kurze Kante zu einem Punkt schrumpfen lässt, als relativ komplex heraus, da die Position des entstehenden Knotens nicht generell auf den Mittelpunkt der ehemaligen Kante gelegt werden kann. Stattdessen hängt die Wahl der Knotenkoordinate von der Lage der kurzen Kante zu ggf. angrenzenden scharfen Innenkanten bzw. Außenkanten ab. Um maßgebliche Kanten des Bauteils zu erhalten, müssen daher gewisse Einschränkungen getroffen werden, welche in Abb. 4.26 verdeutlicht sind. In diesem Beispiel darf ein Knoten entlang der fett eingezeichneten Kantenabschnitte nicht verschoben werden, da sich ansonsten die Gestalt des Bauteils nachhaltig verändern würde. Lediglich innerhalb der Bauteilfläche, und nicht direkt angrenzend an gestaltgebende Kanten, ist daher die freie Wahl der Position des Verschmelzungsknotens erlaubt und wird auf den entsprechenden Mittelpunkt der entfernten Kante gesetzt – wie in Abb. 4.27(a) als Auflösung des in Abb. 4.23(a) gezeigten Fehlerfalls durchgeführt.

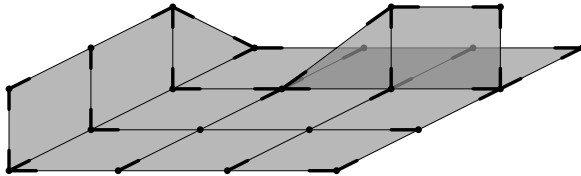


Abbildung 4.26: Randbedingungen beim Verschmelzen kurzer Kanten zu einem gemeinsamen Knoten: Entlang der fett markierten Kantenabschnitte dürfen Knoten nicht verschoben werden.

Bei den in Abb. 4.27(b), 4.27(c) und 4.27(d) illustrierten Fehlerbehebungen handelt es sich ausschließlich um Zerlegungsoperationen. Dabei werden einzelne Elemente aufgespalten, falls an einem oder mehreren Knoten zu große Winkel auftreten, bei einer zu hohen Streckung eines Elements oder bei zu starker Verkippung eines Viereckselements. Diese Zerlegungsvorgänge können u.U. zu hängenden Knoten am Übergang zu Nachbarelementen führen. In solchen Situationen müssen daher die entsprechenden Nachbarelemente ebenfalls geteilt werden, wie dies z.B. beim unteren, für sich fehlerfreien Viereckselement in Abb. 4.27(c) durchgeführt wurde.

Die vorangehend vorgestellten Netzmodifikationen bergen die potenzielle Gefahr, ein vermehrt dreieckslastiges Netz zu erzeugen. Aus diesem Grund wird als letzter Schritt versucht, benachbarte Dreiecke sinnvoll zu Viereckselementen zusammenzufassen. Eine entsprechende Anwendung dieser Dreiecksverschmelzung findet sich in Abb. 4.27(b) und für den Fall eines bereits existenten, dreiecksüberladenen Netzes in 4.27(e).

Detektiert der Algorithmus nach einer abschließenden Relaxation weiterhin fehlerbehaftete Elemente, so werden unter Rücknahme der Relaxationsverschiebungen erneut weitere stark verformte Elemente entsprechend dem beschriebenen Vorgehen bereinigt, bis eine Relaxation des neuen Netzes zum Erfolg führt und keine Elemente mangelhafter Qualität zurückbleiben. Diese Vorgehensweise ist lediglich bei einer starken Anhäufung spitz zulaufender Dreiecke zum

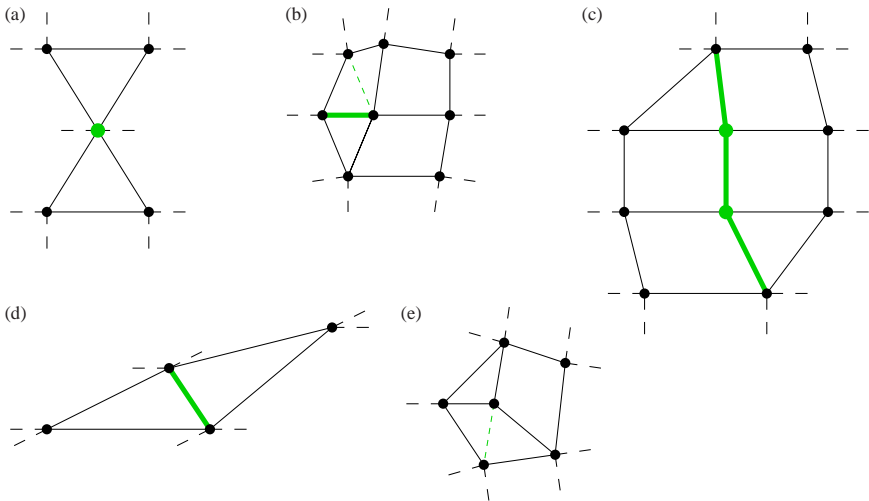


Abbildung 4.27: Lokale Neuvernetzung der fehlerhaften Elementgebiete aus Abb. 4.23: (a) Die kurze Kante wird zu einem Punkt verschmolzen; (b) Elemente werden an großen Winkeln in zwei Dreiecke aufgetrennt, anschließend werden ggf. benachbarte Dreiecke mit spitzen Winkeln zu Viereckselementen verschmolzen; (c) Gestreckte Elemente werden entlang der Mitte langer Kanten aufgetrennt, sofern durch diese Operation nicht zu kurze Kanten entstehen; (d) Unebene Vierecke werden entsprechend der stärksten Verkippungsachse in zwei Dreieckselemente gespalten; (e) Abschließend werden benachbarte Dreiecke verschmolzen, sofern die entstehenden Viereckselemente sämtlichen Qualitätskriterien genügen.

Scheitern verurteilt, da hier nicht garantiert werden kann, dass benachbarte Elemente zu fehlerfreien Viereckselementen verschmolzen werden können. Aus diesem Grund muss in solchen Fällen die lokale Vernetzungsstrategie abgebrochen werden. Als Terminierungskriterium eignet sich hierfür die Bedingung, dass bei jedem Durchgang durch die Selektionsmenge mindestens eine Operation zur Netzverbesserung durchgeführt wird. Für weiterführende Informationen und die geeignete Abfolge der in Abb. 4.27 präsentierten Neustrukturierungen, um eine möglichst schnelle Konvergenz des Verfahrens bei guter Netzqualität zu erreichen, sei an dieser Stelle auf [Zö04] verwiesen.

Werden diese Neuvernetzungsoperationen während der im letzten Unterkapitel vorgestellten interaktiven Modifikation durchgeführt, so erhält der Anwender stets ein direktes Bild vom endgültigen Zustand des Netzes nach Abschluss der Manipulation. Um ein stabiles Verformungs-

verhalten zu garantieren, ohne dabei die benötigten Wichtungsfaktoren laufend neu zu berechnen, werden bei veränderten oder neu eingefügten Knoten diese Faktoren aus der lokal gegebenen Gewichtsverteilung interpoliert. Abbildung 4.28 zeigt, wie zur Vermeidung gestreckter Elemente interaktiv zusätzliche Elemente in ein analog zu Abb. 4.22(c) deformiertes Bauteil eingefügt werden.

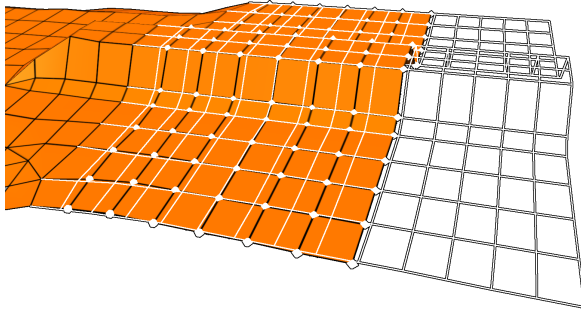


Abbildung 4.28: Verlängerung des Bauteils aus Abb. 4.22(a) mit interaktiver Neuvernetzung: Die gestreckten Elemente aus Abb. 4.22(c) wurden jeweils in zwei neue Elemente aufgespalten.

The whirling, turning height span him and twisted his brain in upon itself till he found himself, eyes closed, whimpering and hugging the hideous wall of towering rock.

He slowly brought his breathing back under control again. He told himself repeatedly that he was just in a graphic representation of a world. A virtual universe. A simulated reality. He could snap back out of it at any moment.

He snapped back out of it.

He was sitting in a blue leatherette foam filled swivel-seated office chair in front of a computer terminal.

Douglas Adams: Mostly Harmless

Kapitel 5

Virtuelle Realität am Arbeitsplatz

5.1 Stereosehen

Der stetig wachsende Komplexitätsgrad erschwert zunehmend das unmittelbare Verständnis eines Fahrzeugmodells. Die Vielzahl überlappender und miteinander verbundener Einzelkomponenten erfordert trotz unterschiedlicher Farbgebung der Einzelteile (siehe auch Abb. 2.9) ein gewisses Maß an Vorwissen, um die Lage bestimmter Komponenten im dreidimensionalen Raum auf einen Blick erkennen zu können. Ohne dieses Wissen muss der Anwender in der Regel zunächst um das Modell herum navigieren und das interessierende Teil von mehreren Seiten betrachten.

Diese Problematik rührt hauptsächlich von der Tatsache her, dass es sich zwar um dreidimensionale Modelle handelt, diese jedoch nur auf einem zweidimensionalen Ausgabegerät als Projektion dargestellt werden. Daher stehen dem Betrachter eines Standbilds lediglich Verdeckungs- und Größenverhältnisse als Hilfsmittel für die Abschätzung der räumlichen Tiefe zur Verfügung. Beide setzen allerdings voraus, dass der Betrachter in etwa über die Form und Größe der verschiedenen Bauteile Bescheid weiß. Verstärkend kommt im Umfeld von FE-Simulationen hinzu, dass Gesamtfahrzeuge teilweise ausgedünnt repräsentiert werden und nichttragende Strukturen vernachlässigt werden. Unter Umständen fehlen somit aus dem CAD oder der Realität bekannte Anhaltspunkte, um z.B. abschätzen zu können, welche genaue Position im Motorraum der Motorblock oder ggf. weitaus komplexer geformte Fahrzeugkomponenten einnehmen.

Die zweidimensionale Darstellung hemmt somit die schnelle, visuelle Aufnahmefähigkeit und bremst u.U. den raschen Fahrzeugentwicklungsprozess. Stereobilddarstellung kann hier Abhilfe schaffen und ermöglicht mittels zwei – unter leicht unterschiedlicher Perspektive – generier-

ten Szenenbildern dreidimensionales Sehen und damit eine relative Tiefenzuordnung der verschiedenen Fahrzeugkomponenten untereinander. Auf die Problemstellung, wie die beiden Bilder dem jeweiligen Auge zugeordnet werden können, wird später eingegangen. Im Folgenden soll zunächst ein kurzer Einblick in die Stereoprojektion gegeben werden.

Im Wesentlichen existieren zwei unterschiedliche Methoden, das Stereokamera paar im Raum auszurichten. Die Positionierung ist bei beiden Ansätzen gleich und erfolgt entsprechend der Koordinaten des Augenpaars des Betrachters. Im Allgemeinen wird dazu, jeweils ausgehend von der mittleren Blickrichtung, eine Kamera um den halben Augenabstand nach links und die zweite Kamera entsprechend nach rechts verschoben.

Bei dem ersten Ansatz, der sogenannten „Toe-In“-Methode, in Abb. 5.1(a) skizziert, werden die beiden Kameras gegeneinander verdreht, so dass sie auf einen gemeinsamen Punkt ausgerichtet sind. Obwohl diese Methode intuitiv richtig erscheinen mag, so ist sie jedoch falsch, da hierbei die perspektivische Verzerrung zum linken und rechten Szenenrand hin unnatürlich stark ausgeprägt ist. Dies rührt daher, dass zum einen durch die Kamerarotation eine Verzerrung eingebracht wird und zum anderen durch die optische Strahlgeometrie auf Betrachterseite eine weitere perspektivische Verzerrung zum Bildschirmrand hin auftritt. Dies äußert sich z.B. beim Betrachten einer vertikalen Kante einer parallel zur Bildebene verlaufenden Fläche, welche auf dem einen Auge perspektivisch zu lang erscheint und auf dem anderen zu kurz. Der Vorteil dieser Vorgehensweise liegt jedoch in der einfachen und anschaulichen Implementierung der Stereoparameter und in der Verwendung von symmetrischen Frusta.

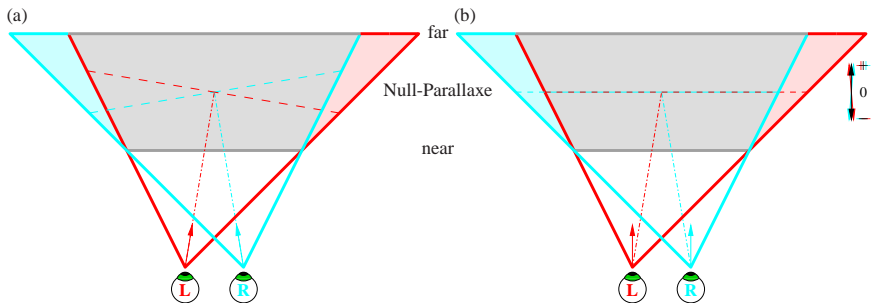


Abbildung 5.1: Kameraeinstellungen für die Stereoprojektion: (a) symmetrisch, auf einen Punkt fokussiert, (b) asymmetrisch.

Bei der zweiten Methode werden hingegen asymmetrische Frusta eingesetzt und die beiden Kameras nicht gedreht. Damit lassen sich die oben erwähnten Verzerrungsprobleme beheben. Nachteil hierbei ist die vergleichsweise geringe Anschaulichkeit der Stereoparameter. Streng genommen liefert auch diese Methode kein korrektes Ergebnis. Dies ist auf die Art der Projektion zurückzuführen. Aufgrund der Optik des Auges müsste eine Kugelprojektion angewandt werden.

Wegen der einfacheren Mathematik und keinem zusätzlichen Bedarf an Unterteilungen in räumlich ausgedehnten Flächen wird i.A. allerdings nur eine lineare Projektion hardware-beschleunigt unterstützt. Der Fehler ist bei üblichen Blickwinkeln bzw. Perspektiven jedoch minimal, so dass im Folgenden die übliche, hardware-unterstützte Variante zum Einsatz kommt.

Ein schiefwinkliges Frustum kann erzeugt werden, indem in der allgemeinen perspektivischen Projektionsmatrix

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (5.1)$$

für die linke und rechte Begrenzung (l bzw. r) des Frustums unterschiedliche Werte eingesetzt werden. Der Offset von der halben Breite w des Frustums sollte dabei den halben Augabstand e des Betrachters betragen. In der Vertikalen ist das Frustum i.A. nicht schiefwinklig, daher entspricht die obere und untere Begrenzung betragsmäßig der halben Höhe h des Frustums.

$$l = -w/2 \pm e/2 \quad r = w/2 \pm e/2 \quad t = -b = h/2 \quad (5.2)$$

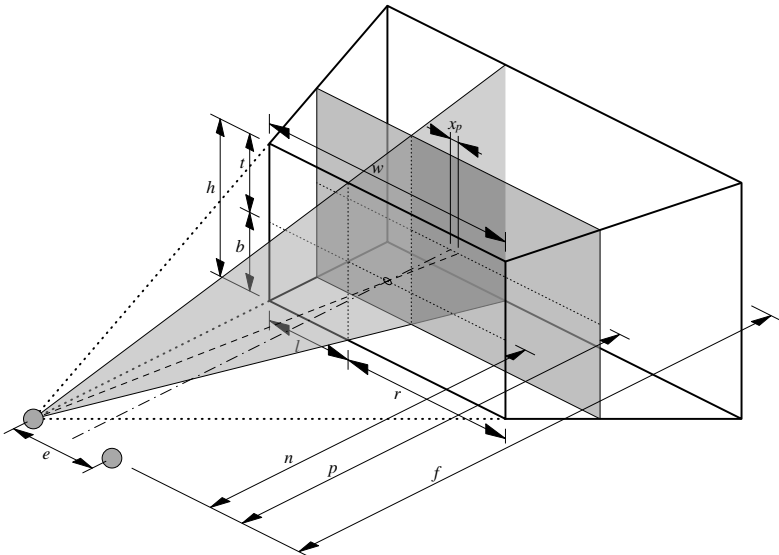


Abbildung 5.2: Parameter eines schiefwinkligen Frustums.

Die vordere und hintere Begrenzung („near clipping plane“ n bzw. „far clipping plane“ f , siehe auch Abb. 5.1 und 5.2) werden im Rahmen der auch bei Nicht-Stereo-Anwendungen üblichen Parameter gewählt.

Die Null-Parallaxe definiert dabei die Tiefe der Szene, in welcher die beiden Stereodarstellungen deckungsgleiche Bilder liefern und liegt unter obigen Annahmen auf der vorderen Begrenzung des Frustum, welche gleichzeitig die Projektionsebene ist. Dies ergibt Sinn, da sich somit – vom Tiefeneindruck her – die gezeichneten Objekte hinter der Projektionsebene und damit innerhalb des Frustum befinden müssen. Soll der Eindruck entstehen, dass sich die Objekte teilweise oder vollständig vor der Projektionsfläche bzw. dem Anzeigegerät befinden, so erreicht man dies mittels einer Änderung des Abstands p der Null-Parallaxe zum Betrachter durch eine horizontale Translation x_p der Szene je nach Augenseite in jeweils gegensätzlicher Richtung senkrecht zur Blickrichtung:

$$x_p = \pm \frac{1}{2} e \left(\frac{p}{n} - 1 \right) \quad (5.3)$$

Gute visuelle Ergebnisse werden dabei erzielt, wenn die Null-Parallaxe ungefähr im vorderen Drittel der Gesamtszene liegt. Liegen die vordere und hintere Begrenzungsebene dicht am Objekt, was auch wegen einer maximalen Ausnutzung der Tiefenpuffergenauigkeit ratsam ist, so ist somit $x_p = \frac{1}{6} e \left(\frac{f}{n} - 1 \right)$ zu wählen. Unter Berücksichtigung von (5.1), (5.2) und (5.3) ergibt sich für die gesamte Projektionsmatrix:

$$\mathbf{R}_{\text{Stereo}} = \begin{pmatrix} \frac{2n}{w} & 0 & s \frac{e}{w} & s \frac{e}{w} (p-n) \\ 0 & \frac{2n}{h} & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \text{mit } s = \begin{cases} -1 & : \text{ rechtes Auge} \\ 1 & : \text{ linkes Auge} \end{cases} \quad (5.4)$$

Unter Anwendung dieser Projektionsmatrix kann entsprechend der Parameter ein Stereobildpaar erzeugt werden. Dieses Bildpaar kann auf unterschiedliche Arten dargestellt werden. Im Wesentlichen wird dabei zwischen aktiven und passiven Ansätzen unterschieden. Bei ersteren ist immer anwenderseitig eine aktive Komponente beteiligt, z.B. „Shutter“-Brillen, bei denen in schneller Abfolge das linke und rechte Auge abwechselnd zugedeckt wird. In Synchronisation dazu wird auf dem graphischen Ausgabegerät das entsprechende Stereobild präsentiert. Diese Synchronisation kann u.U. sehr aufwändig werden und man benötigt im Allgemeinen vier Bildschirmpuffer – jeweils einen doppelten Puffer für jedes Auge – für eine flackerfreie Darstellung.

Passive Stereoanwendungen arbeiten in der Regel mit speziellen Filtern, welche eine eindeutige Trennung des Stereobildpaares ermöglichen. Eine kostengünstige Variante sind hierbei Rot-Grün- oder Rot-Cyan-Brillen, welche mit einem Darstellungsgerät auskommen, auf welchem im Rot-Kanal das Bild für das linke Auge und im Grün- sowie ggf. im Blau-Kanal das Bild für das rechte Auge wiedergegeben wird. Der Nachteil hierbei sind jedoch gravierende Einbußen in der Wiedergabemöglichkeit des Farbraums, so dass man sich i.A. auf Farben beschränken muss, welche beide Filterfarben zumindest teilweise beinhalten.

Polarisationsfilter schaffen hier Abhilfe, benötigen jedoch zwei Displaygeräte. Ein typisches Anwendungsbeispiel sind z.B. die Videoprojektoren einer im passiven Stereomodus betriebenen PowerWall, bei der zwei¹ Videoprojektoren das Stereobildpaar mit entsprechend orthogonaler Lichtpolarisation in der Regel von hinten auf eine lichtdurchlässige und polarisationserhaltende Scheibe projizieren. Tragen die Betrachter Brillen mit entsprechenden Polarisationsfiltern, so erreicht das jeweilige Auge nur das entsprechend zugeordnete Bild [FDF⁺94, HMD97]. Dieses Verfahren wird allerdings meist nur bei großflächigen Projektionseinrichtungen eingesetzt und stellt aufgrund der Videoprojektoren und der speziellen Anforderungen an die Projektionsfläche eine vergleichsweise teure Lösung dar. Im Gegensatz dazu soll im Folgenden eine alternative, platzsparende Passiv-Stereo-Technologie vorgestellt werden, welche ohne weitere Hilfsmittel – wie Brillen – auskommt. Diese stellt daher eine ideale Arbeitsplatzlösung dar und bildet eine gute Ergänzung zu bestehenden Visualisierungsmethoden für FE-Modelle.

5.1.1 Autostereoskopie

In den vergangenen Jahren erlangte eine zunehmende Anzahl an autostereoskopischen Displaygeräten die Marktreife. Im technischen Detail unterscheiden sich die verschiedenen Ansätze [PPK00, PPK⁺01, See04]. Bei allen gängigen Lösungen werden jedoch die beiden Stereobilder gleichmäßig über der Anzeigefläche verteilt. Häufig wird eine ineinander verschachtelte spaltenweise Anzeige des Stereobildpaares eingesetzt, bei welcher in ungeraden Pixelspalten das Bild für das linke Auge dargestellt und in den geradzahigen Spalten entsprechend die Szene mit der Perspektive des rechten Auges gezeichnet wird. Ein vor dem Bildschirm angebrachtes Blenden- oder Prismensystem schirmt oder lenkt das Licht der einzelnen Pixelspalten dabei so ab, dass der Betrachter mit dem jeweiligen Auge nur die dafür bestimmte Bildinformation sieht. Wegen der positionsgenauen Abbildung einzelner Pixel eignen sich für diesen Zweck insbesondere digitale TFT- bzw. DFP-Monitore. Abbildung 5.3 zeigt die Prinzipskizze eines autostereoskopischen Monitors, bei dem die gerichtete Aufteilung der Pixelspalten mittels eines Prismensystems bewerkstelligt wird [See04]. Seit kurzem ist solch ein dreidimensionales Display auch für tragbare Computer erhältlich [HJWE00], was die Flexibilität und das große Einsatzgebiet dieser Darstellungsmöglichkeit unterstreicht.

Die optische Zuordnung der Stereobilder zum betreffenden Auge funktioniert bei solchen Ansätzen allerdings nur in einem sehr engen, horizontal beschränkten Bereich. Daher ist in der Regel bei solchen Geräten über der Darstellungsfläche ein zusätzliches Dual-Kamera-System integriert, welches über die Auswertung der aufgenommenen Bildinformation – meist durch einen eingebauten Prozessor – verzögerungsarm die Position der Augen des Betrachters erfasst. Die Prisma- oder Blendeneinheit wird dementsprechend nachjustiert, so dass eine korrekte Stereowahrnehmung innerhalb eines deutlich größeren Bereichs erreicht werden kann.

Um ein solches Display betreiben zu können, muss ein Stereobildpaar der Szene horizontal alternierend in jede Pixelspalte des Bildschirmspeichers gezeichnet werden. Einige Graphikkartenhersteller stellen bereits spezielle Treiber für diesen Zweck zur Verfügung, allerdings sind die-

¹oder entsprechende Vielfache davon bei gekachelten Großprojektionen

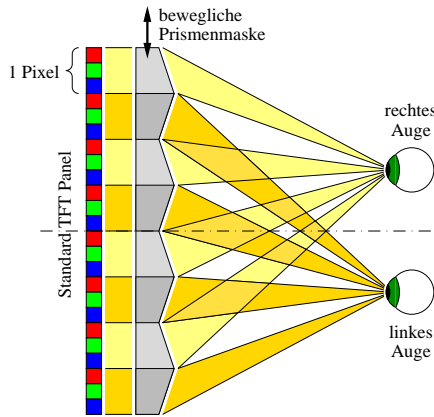


Abbildung 5.3: Funktionsprinzip eines autostereoskopischen Displays, welches ein in den einzelnen Pixelspalten abwechselnd verteiltes Stereobildpaar mittels eines Prismensystems auftrennt und zum entsprechenden Auge ablenkt. Die Prismenmaske ist horizontal beweglich und wird entsprechend der Augposition mittels eines Schrittmotors verschoben.

se entweder teuren High-End Graphikkarten vorbehalten oder unterstützen lediglich Windows-Plattformen. Daher wurde in die entwickelte FE-Preprocessing-Anwendung eine native Unterstützung dieser Art von Ausgabegeräten implementiert.

Eine performante und einfache Möglichkeit, die abwechselnde Verteilung der beiden Szenenbilder über die Breite des Bildschirmspeichers zu realisieren, basiert auf der Anwendung einer entsprechenden Stanzmaske. In einem speziellen Puffer („Stencil Buffer“) werden dazu die geradzahligen Pixelspalten mit einer Eins und die ungeraden Spalten mit einer Null markiert. Beim Zeichnen des Bilds für das linke Auge werden durch Anwenden eines entsprechenden Tests nur die Bereiche im Farb- und Tiefenpuffer gesetzt, welche mit einer Null gekennzeichnet wurden. Dementsprechend wird die Szene für das rechte Auge nur dort gezeichnet, wo in dem Stanz-Puffer eine Eins zu finden ist. Je nach den gegebenen Möglichkeiten der Graphikkarte ist die Verwendung dieses zusätzlichen Puffers mit mehr oder weniger gravierenden Geschwindigkeitseinbußen verbunden. In diesem Fall ist es zweckmäßig, den betreffenden Stanz-Test nur im zweiten Durchgang – beim Zeichnen der rechtsäugigen Szene bzw. beim Test auf Eins – durchzuführen, so dass die Bildinformation des komplett gezeichneten Bilds für das linke Auge in den entsprechenden Spalten überschrieben wird.

Diese einfache Vorgehensweise liefert jedoch nur für grobe Strukturen bzw. gefüllte Flächen akzeptable Ergebnisse. Beim Zeichnen dünner Linien, um z.B. die Berandungen der finiten Elemente darzustellen, können jedoch deutliche Artefakte auftreten. Abbildung 5.4 verdeutlicht diese Problematik an einer schräg verlaufenden Linie, welche rasterisiert wird [Bre65] und un-

ter Berücksichtigung der Sperrung jeder zweiten Spalte in den Bildschirmspeicher geschrieben wird (Abb. 5.4(a)). Durch das vor den Bildschirm geschaltete Prismensystem wird das Stereobild entsprechend getrennt und der Betrachter sieht auf seinem linken Auge das in Abb. 5.4(b) dargestellte Muster. Wie man sieht, wird die Linie hierbei – wie auch für das rechte Auge – nur bruchstückhaft wiedergegeben.

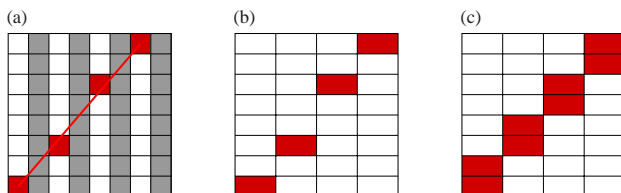


Abbildung 5.4: Zeichnen einer dünnen Linie zur Wiedergabe auf einem autostereoskopischen Bildschirm: (a) Rasterisierung unter Berücksichtigung der gesperrten Pixelzeilen (grau hinterlegt); (b) Durch das Prismensystem betrachtet ergibt sich für das linke Auge eine lückenhafte Darstellung der Linie; (c) Korrekte Darstellung durch Aufweiten eines zwischengespeicherten Bilds, welches in einem auf die halbe Auflösung gestauchten Bildschirmspeicher erzeugt wurde (Sicht des linken Auges durch das Prismensystem).

Bessere Ergebnisse werden erzielt, wenn die Szene jeweils mit einer in der Breite halbierten Auflösung in den verdeckten Bildschirmspeicher gezeichnet wird (Abb. 5.4(c)). Allerdings muss zumindest ein Teil des Stereobildpaars zwischengespeichert werden, um eine entsprechende Verteilung der Bildinformation auf die verschiedenen Spalten des endgültigen Bilds zu ermöglichen. Im Detail kann die spaltenweise Aufteilung unterschiedlich realisiert werden. So kann z.B. die linke Ansicht in einer Textur zwischengelagert werden², anschließend wird die rechte Ansicht gezeichnet und die einzelnen Spalten unter Verwendung von `glCopyPixels` an die endgültige Position kopiert. Durch Zeichnen texturierter Linien oder ein Pixel breiter Rechtecke kann das zwischengespeicherte Bild für das linke Auge zwischen das Bild der rechten Ansicht eingestreut werden. Hierfür ist jedoch auch eine Lösung durch Zeichnen eines bildschirmfüllenden, und mit der benötigten Bildinformation texturierten, Polygons unter Anwendung eines geeigneten Stanz-Tests denkbar, dessen Einsatz in diesem Fall keine Darstellungsartefakte nach sich zieht. Die jeweils performanteste Strategie zur Generierung einer für autostereoskopische Bildschirme geeigneten Darstellung ist stark Hardware-abhängig, so dass keine universelle, gemeingültige Lösung möglich ist. Abbildung 5.5 zeigt ein mittels dieser prinzipiellen Vorgehensweise generiertes Bild eines Gesamtfahrzeugmodells.

Zusätzlich liefert der vorgestellte Ansatz über die Rasterisierung der Szene in halber Auflösung auch beim Einsatz von Anti-Aliasing-Techniken bessere Ergebnisse als die zuvor vorgestellte

²unter Verwendung einer speziellen OpenGL-Erweiterung kann auch direkt in eine Textur gezeichnet werden

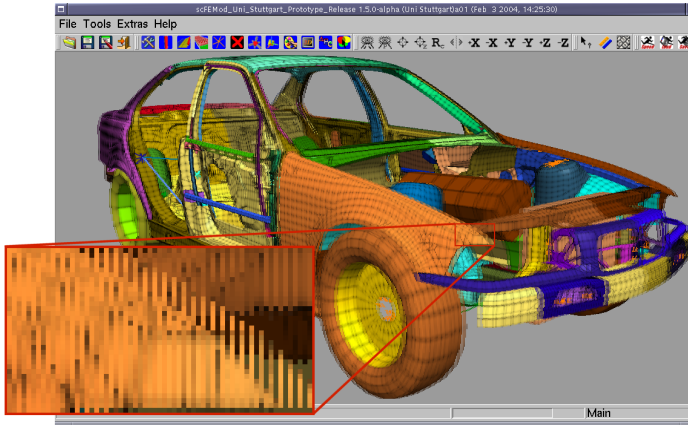


Abbildung 5.5: Aufbereitung einer Stereobilddarstellung eines Gesamtfahrzeugmodells zur Ausgabe auf einem autostereoskopischen Monitor. Im vergrößerten Ausschnitt ist die spaltenweise Verteilung der Bildinformationen für das linke und das rechte Auge zu erkennen.

Lösung über den „Stencil Buffer“ und ist daher trotz des erhöhten Aufwands in jedem Fall vorzuziehen. Unabhängig von der gewählten Aufbereitungsstrategie müssen jedoch in jedem Fall zwei Ansichten des Modells generiert werden, so dass sich hierfür der Einsatz des in Kapitel 3.2.3 vorgestellten Simplifizierungsverfahrens empfiehlt.

Einige autostereoskopische Displaygeräte ermöglichen über eine serielle Schnittstelle o.Ä. die Abfrage der mittels der eingebauten Kameras ermittelten Kopf- bzw. Augposition. Durch Berücksichtigung dieser Angaben kann der Blickwinkel und ggf. die Entfernung, unter dem das Modell betrachtet wird, entsprechend variiert werden, so dass auf diese Weise eine einfache Art von optischem Tracking realisiert werden kann. Ohne eine solche Nachführung der Projektionsparameter wird ansonsten beim Anwender die Vermutung erweckt, dass sich das betrachtete Objekt mitbewegt, da es relativ zum Beobachter immer gleich ausgerichtet erscheint.

Da sich ein Bildpaar die gleiche Bildschirmfläche teilen muss, ist die Auflösung – unabhängig vom autostereoskopischen Darstellungsverfahren – zwangsläufig halbiert. Dies stellt einen gewissen Nachteil gegenüber herkömmlichen, zweidimensionalen Ausgabegeräten dar. Mit der rasanten Entwicklung auf dem Gebiet der Flachbildschirmtechnologie ist jedoch in naher Zukunft eine deutliche Verbesserung der diesbezüglichen Darstellungsqualität zu erwarten.

5.1.2 Generische Erweiterung von OpenGL-Programmen um Stereo-Funktionalität

In der Regel unterstützen lediglich speziell dafür konzipierte Anwendungen die native Erzeugung von Stereobildern für eine dreidimensionale Darstellung auf geeigneten Ausgabegeräten. Jedoch kann es u.U. wünschenswert sein, einer monoskopischen Anwendung die Fähigkeit zur stereoskopischen Bildaufbereitung zu verleihen. Auf Windows Plattformen existieren zu diesem Zweck – wie bereits erwähnt – einige treiberbasierte Lösungen. Im Folgenden soll ein Ansatz für UNIX-Plattformen vorgestellt werden, welcher mittels einer generischen Methode die Erweiterung eines Programms um die Fähigkeit zur Stereodarstellung ermöglicht. Die prinzipielle Vorgehensweise ist dabei nicht auf UNIX-Umgebungen beschränkt, sondern kann ebenso auf andere Plattformen übertragen werden.

Die Idee hinter dem gewählten Ansatz ist das Einfügen einer zusätzlichen Bibliothek, welche bestimmte OpenGL-Funktionen überschreibt und durch eigene ersetzt. Dieses Überschreiben wird im UNIX-Jargon als „Preloading“ bezeichnet und erfolgt während des dynamischen Einbindens der Funktionsbibliotheken beim Laden eines Programmes [HO91]. Die zusätzlich eingefügten Routinen haben dabei Zugriff auf die ursprünglichen Bibliotheksfunktionen und können diese weiterhin verwenden. Abbildung 5.6 zeigt, wie sich dabei die zusätzlichen Funktionsroutinen zwischen der eigentlichen Applikation und den originären Bibliotheken einnisten. Auf diese

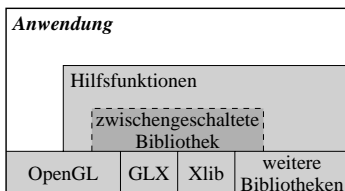


Abbildung 5.6: Die Einbindung der verwendeten Programmbibliotheken zur Laufzeit erlaubt das Einschleusen zusätzlicher Bibliotheksroutinen (Beispiel für eine UNIX-Umgebung nach [Mag04]).

Weise kann das ursprüngliche Verhalten beim Aufruf von OpenGL-Routinen gezielt verändert werden, ohne Änderungen im Quellcode der Anwendung vornehmen zu müssen. Dieser Ansatz funktioniert daher auch mit Applikationen von Drittanbietern, bei denen keine Zugriffsmöglichkeit auf den Programmcode besteht. Ähnliche Vorgehensweisen sind ebenfalls unter Windows umsetzbar und werden in diesem Umfeld meist mit „DLL Hooking“ betitelt [HB99].

Um eine Anwendung unter Einsatz dieser Möglichkeiten stereofähig zu machen, müssen verschiedene Randbedingungen erfüllt werden. Zunächst muss eine Möglichkeit gegeben sein, die Zeichenroutine des Programms zu initiieren, so dass ein zweimaliges Zeichnen zur Erzeugung des Stereobildpaares möglich ist. Hierfür sind verschiedene Ansätze denkbar. Zum einen kann – ausgehend von der Aufrufadresse eines häufig verwendeten OpenGL-Befehls –

versucht werden, im Funktionsaufruf-Stapelspeicher die übergeordnete Routine zu finden. Bei einfach strukturierten Programmen handelt es sich dabei um den Programmabschnitt zum Darstellen der Szene, und die so gewonnene Einsprungsadresse kann dazu verwendet werden, die Szene zweimal zeichnen zu lassen. Für komplexere Programme wird diese äußerst simple Heuristik allerdings fehlschlagen (siehe auch [SRE02]).

Eine weitere Möglichkeit besteht im Mitschneiden sämtlicher OpenGL-Aufrufe, so dass diese anschließend entsprechend wiedergegeben werden können. Obwohl diese Methode – was exakte Reproduzierbarkeit der Szenenbeschreibung angeht – sehr zuverlässige Ergebnisse liefert, so zieht sie jedoch einen enormen Aufwand zur Verwaltung der anfallenden Daten nach sich, welche für große Szenen ggf. viel Speicherplatz benötigen und folglich den Programmablauf entsprechend verlangsamen. Außerdem muss zu diesem Zweck der komplette Befehlssatz von OpenGL überschrieben werden, ein Unterfangen, welches lediglich mittels automatisch generiertem Code durch entsprechende Auswertung der OpenGL Musterimplementierung [Silb] realisierbar ist.

Weitere Methoden sind denkbar [SRE02], auf sie soll jedoch nicht weiter eingegangen werden. Stattdessen wird ein vergleichsweise einfacher Ansatz vorgestellt, welcher mit einer Vielzahl von Programmen zuverlässig arbeitet und gute Ergebnisse liefert. Dabei ist der genaue Aufbau der Zeichenroutine und die Abfolge der einzelnen OpenGL-Befehlssequenzen zweitrangig. Man verlässt sich vielmehr auf die Tatsache, dass ein Programm den Fensterinhalt erneut zeichnet, wenn es von einem anderen Fenster verdeckt wurde und wieder aufgedeckt wird. Das Ereignis „Fenster wieder sichtbar“ lässt sich künstlich erzeugen [Nye95] und wird durch das Fenstersystem der Applikation übersandt. Handelt es sich um eine sauber programmierte Anwendung, so sollte diese den Fensterinhalt rekonstruieren, was bedeutet, dass die komplette Szene erneut gezeichnet werden muss. Durch Erzeugen eines vorgetäuschten Ereignisses lässt sich die Applikation somit beliebig oft zu einem Neuzeichnen bewegen.

Die zweite wesentliche Randbedingung ist die Kenntnis der momentanen Kameraperspektive. Durch Abfangen der für eine Veränderung der Kameraparameter in Frage kommenden Funktionsaufrufe (`glFrustum`, `glLoadMatrix*`, `glMultMatrix*`) lassen sich unter Auswertung der resultierenden perspektivischen Projektionsmatrix anhand (5.1) die benötigten Variablen zum Aufbau einer entsprechend der Stereoparameter nach (5.4) modifizierten Projektionsmatrix bestimmen:

$$n = \frac{r_{34}}{r_{33} - 1} \quad l = \frac{(r_{13} - 1) r_{34}}{r_{11} (r_{33} - 1)} \quad r = \frac{(r_{13} + 1) r_{34}}{r_{11} (r_{33} - 1)} \quad (5.5)$$

Ein Stereobild lässt sich unter Einsatz der gewonnenen Erkenntnisse aus einer Applikation ohne deren Wissen wie folgt erzeugen: Zunächst zeichnet die Applikation das Bild für das linke Auge. Die überschriebenen Perspektiven-relevanten Routinen sorgen dabei für eine entsprechende Änderung des Frustum. Nach Vollendung des Zeichenvorgangs löst die Anwendung zur Anzeige des gezeichneten Bilds einen Pufferaustausch der beiden Bildschirmpuffer aus³ (`glXSwapBuffers`). Dieser Austausch wird verhindert und stattdessen die perspektivischen

³hierbei wird davon ausgegangen, dass die Anwendung einen doppelten Bildschirmpuffer verwendet, was im Allgemeinen zutrifft

sowie ggf. sonstigen Einstellungen für das rechte Auge vorgenommen und das erläuterte Ereignis erzeugt, welches die Anwendung zum Neuzeichnen veranlasst. Nach Abschluss dieses zweiten Zeichenvorgangs wird der Pufferaustausch nicht verhindert, sondern ordnungsgemäß durchgeführt. Das Ergebnis ist eine stereoskopische Darstellung der Szene.

Unter Anwendung dieser Vorgehensweise lassen sich unterschiedliche passive Stereotechniken realisieren. Neben den im letzten Abschnitt diskutierten Techniken zur Erzeugung autostereoskopischer Bilder können auch ursprünglich nicht dafür vorgesehene Anwendungen auf stereofähige Großbildprojektionen, wie z.B. PowerWalls, übertragen werden. Es können aber auch kostengünstige Alternativen der Stereodarstellung, wie Anaglyphe gewählt werden. Abbildung 5.7 zeigt ein Rot-Cyan-Anaglyph, welches mittels des vorgestellten Ansatzes und einer älteren Version des in der Arbeitsgruppe entwickelten FE-Werkzeugs generiert wurde, die noch keine native Stereodarstellung unterstützt.

Letzteres Beispiel ist attraktiv für die Darstellung von Stereobildern auf mobilen Kleincomputern, welche keine leistungsfähige Graphikhardware besitzen und außerdem nicht über die Technik für komplexe Stereodarstellungsmethoden verfügen. Daher werden solche Anaglyphe in der Regel auf einem leistungsfähigen Rechner erzeugt und drahtlos an das mobile Gerät zur Anzeige übertragen [SME02, SRE02].

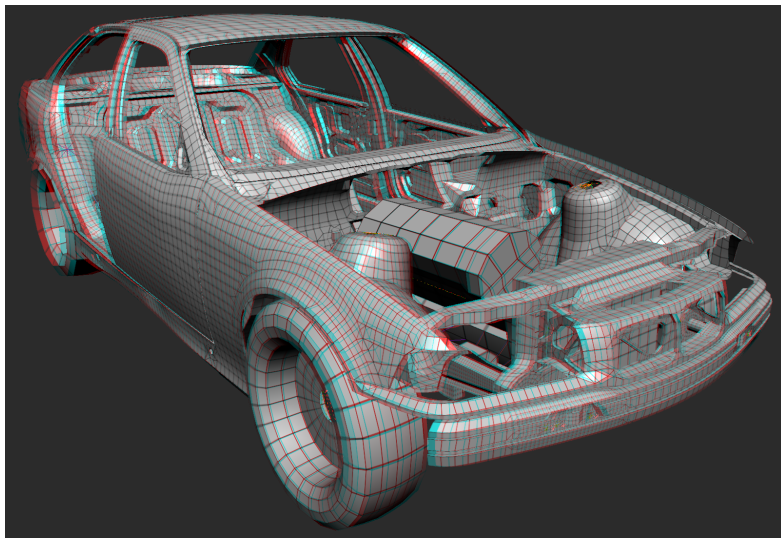


Abbildung 5.7: Rot-Cyan-Anaglyph eines Fahrzeugmodells, erzeugt durch generische Erweiterung einer Anwendung um Stereofähigkeit.

5.2 Interaktion in der Virtuellen Realität

Dreidimensionale Visualisierungs- und Anzeigetechniken können das Verständnis und die Begreifbarkeit eines Datensatzes deutlich verbessern und folglich den Zeitaufwand für die Analyse und Bewertung des dargestellten Objekts minimieren. Im Gegensatz dazu stellt sich eine Interaktion bzw. die Durchführung diffiziler Aufgaben in solch dreidimensionalen Welten teilweise problematisch dar. Diese Schwierigkeiten sind häufig darauf zurückzuführen, dass die Steuerung der Interaktion mit gängigen zweidimensionalen Eingabemethoden und -geräten durchgeführt wird.

Ein Beispiel ist die bereits im letzten Hauptkapitel diskutierte Steuerung von Modifikationen mittels einer zweidimensionalen Maus. Diese liefert lediglich zwei kontinuierliche Eingabeparameter, so dass eine direkte Angabe dreidimensionaler Werte damit nicht möglich ist. Der auf dem Bildschirm in der Regel eingeblendete Mauszeiger zerstört zudem den dreidimensionalen Eindruck des dargestellten Modells, da er je nach eingesetzter Stereotechnik nur auf einem Auge zu sehen, oder auf beiden Augen an derselben Position sichtbar ist und daher den Anschein erweckt, auf der Null-Parallaxen-Ebene fixiert zu sein. Meist passt diese Annahme nicht mit der Tiefeninformation der Szene zusammen, und u.U. müsste der Mauszeiger, dem Tiefeneindruck nach, durch ein Objekt verdeckt sein, ist aber dennoch im Vordergrund sichtbar. Diese virtuelle Bindung des Mauszeigers an die Null-Parallaxe erschwert zudem die exakte Platzierung neuer Objekte – wie z.B. von Schweißpunkten – auf der Oberfläche eines nicht in dieser Ebene liegenden Bauteils.

Verbesserungen im Umgang mit dreidimensionalen Modellen können daher nur erreicht werden, indem sowohl die Eingabe selbst als auch die Visualisierung der dabei übergebenen Parameter, z.B. einer Position im Raum, komplett dreidimensional erfolgt. Die Aussage beschränkt sich dabei nicht auf die bloße, mausgesteuerte Positionsangabe, sondern gilt auch für weitere Kontrollmethoden wie Menüs oder Piktogramme. Diese können ebenso den dreidimensionalen Eindruck negativ beeinflussen und schwierig anzusteuern sein, vor allem wenn dem Anwender keine im direkten Sinne mausähnlichen Eingabegeräte zur Verfügung stehen. Um diese Problematik zu verdeutlichen, sollen an dieser Stelle zwei denkbare Szenarien beschrieben werden.

Im ersten Szenario möchte der Anwender eine PowerWall- oder CAVE-gestützte Anwendung bedienen. Als Eingabemedium soll dabei ein dreidimensionales Tracking-Gerät dienen, welches Positions- und Richtungsangaben liefert. Auf welche Art das Tracking realisiert wird – z.B. optisch oder magnetisch – ist hierbei irrelevant. In der Regel wird die Position und Richtung direkt in die virtuelle Szene übertragen und häufig derart visualisiert, dass der Anwender eine Art Licht-Schwert in der Hand hält, mit dem er im dreidimensionalen Raum umherzeigen kann [RL98]. Um mit diesem Schwert eine Auswahl in einem Menü treffen zu können, muss letzteres zweckmäßigerweise in den dreidimensionalen Raum übertragen werden. [RFL⁺98]. Ein dreidimensionales Menü hat – neben der dadurch gelösten Parallaxen- bzw. Tiefenproblematik – mehrere Vorteile. So ist es einfacher, den angewählten Menüpunkt festzustellen, da es einen eindeutigen Durchstoßpunkt des Strahls auf der Menüoberfläche gibt. Zum anderen kann solch ein Menü im dreidimensionalen Raum bewegt und ggf. zur Seite geschoben werden, falls es interessante Szenenbereiche verdeckt.

Im zweiten Szenario wird eine dreidimensionale Anwendung über einen mobilen Kleincomputer ferngesteuert [PGGZ04]. Dabei werden sämtliche Interaktionsstrukturen – und somit auch die Menüs – an den Hand-held-Computer übergeben. Die direkte Übertragung von Menüs auf solche Kleingeräte ist dabei meist wenig sinnvoll, da hier nur sehr wenig Platz zur Verfügung steht und die Eingaben des Anwenders über einen Zeigestift erfolgen müssen, mit welchem der Benutzer durch Tippen auf die Displayfläche einen Mausklick auslösen kann. In Desktopcomputer-Terminologie ausgedrückt, steht dem Anwender somit nur eine Art Maus mit einer Schaltfläche zur Verfügung. Dass bei der direkten Übertragung von Menüstrukturen aus dem Desktopbereich auf mobile Kleincomputer daher zwangsweise Defizite entstehen, wurde frühzeitig festgestellt [Kuu99, HK94]. Eine entsprechende Anpassung an Kleingeräte ist allerdings je nach Anwendungsfall und -zweck ggf. mühsam, wird aber dennoch angestrebt [WB00].

Im folgenden Abschnitt soll gezeigt werden, wie bestehende Menüs auf generischem Weg an eine dreidimensionale Umgebung angepasst werden können. Dabei wird insbesondere das zuletzt skizzierte Szenario aufgegriffen und ein Ansatz vorgestellt, der es erlaubt, auch ausgedehnte Menüstrukturen platzsparend zu repräsentieren, so dass sie dabei mit einem Zeigestift leicht navigierbar bleiben [RSR⁺03].

5.2.1 Menüs

Der hier vorgestellte Ansatz, Menüs neu zu strukturieren, basiert auf der gleichen Vorgehensweise, wie sie bereits in Kapitel 5.1.2 erläutert wurde. Das heißt, auch hier wird eine zusätzliche Funktionsbibliothek in die bestehende Anwendung injiziert, so dass bestimmte Programmroutinen abgeändert werden können. Dies erlaubt zum einen, die zu Grunde liegende Menülogik wiederzuverwenden, und zum anderen eine leichte und schnelle Austauschbarkeit des Menüsystems bei Programmen, welche auf derselben Menübibliothek aufbauen. Damit eignet sich ein solch generischer Ansatz besonders zur schnellen Entwicklung von Prototypen, welche auf einer breiten Basis von Anwendungen getestet werden können, so dass eventuelle Fehler im Design oder der Umsetzung sehr schnell erkannt werden.

Soll ein bestehender Menübaum abgewandelt werden, so muss die eingefügte Bibliothek im Wesentlichen drei Aufgabenbereiche abdecken. Zunächst müssen die Funktionen zur Erzeugung der Menüpunkte abgefangen werden. Anschließend muss die dabei gewonnene Information über den Menüaufbau analysiert und ggf. zur späteren Verwendung abgespeichert und zuletzt Routinen bereitgestellt werden, welche die entsprechenden Teile der Benutzerschnittstelle überschreiben und durch eigene Ausgabe- und Interaktionsmethoden ersetzen. Die beiden ersten Schritte sind dabei einander verknüpft.

Um die Anwendbarkeit dieses Ansatzes zur generischen Anpassung eines Desktop-orientierten Menüs an die Bedürfnisse von Kleinstgeräten demonstrieren zu können, wurde exemplarisch die Menüfunktionalität der bei OpenGL-Anwendungen häufig eingesetzten GLUT-Bibliothek [Kil96] überschrieben. Der überschaubare Funktionsumfang dieser Bibliothek ermöglichte den schnellen Entwurf einer prototypischen Lösung, welche im Folgenden an einem bewusst einfach gehaltenen Programm zur Partikelsimulation demonstriert werden soll. Prinzipiell ist je-

doch auch für umfangreichere GUI-APIs – wie z.B. Qt oder MFC – diese Adaption ohne weiteres möglich, wenngleich auch der Aufwand deutlich höher liegt. Somit ist der Transfer des hier gezeigten Ansatzes auf entsprechend komplexere Programme – wie ein FE-Preprocessing-Werkzeug – ebenfalls denkbar.

Abbildung 5.8(a) verdeutlicht, wie GLUT beim Aufklappen einer verschachtelten Menüstruktur den begrenzten Platz auf einem mobilen Endgerät großflächig für die Anzeige des Menüs beansprucht. Die über mehrere Prototypenstadien optimierte Lösung zur Anzeige und Bedienung von Menüs auf mobilen Kleincomputern in Abb. 5.8(b) hingegen zeichnet sich durch einen konstant geringen Platzbedarf aus. Die Größe und Position der Menüfläche kann durch Ziehen an den

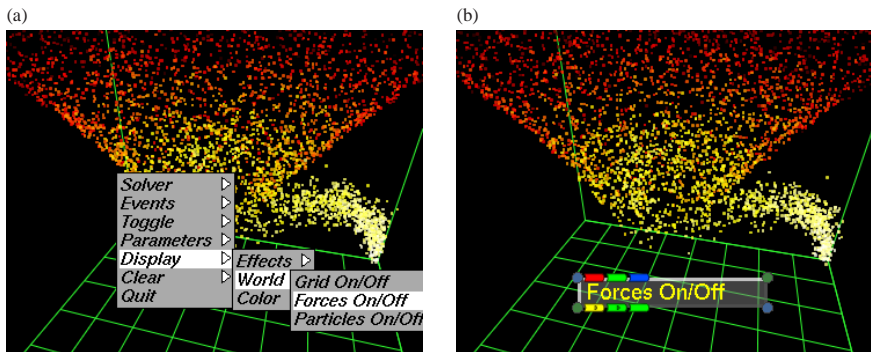


Abbildung 5.8: Menü mit drei Ebenen auf einer kleinen Ausgabefläche: (a) originäres GLUT-Menü; (b) alternative Menüdarstellung durch Einschleusen einer zusätzlichen Bibliothek.

Eckpunkten variiert werden. Die Menünavigation ist an [DE01] angelehnt und ermöglicht das zyklische Durchlaufen der einzelnen Menüpunkte, d.h. an das Ende der Menüliste schließt sich der Beginn derselben Liste zyklisch an, so dass die benötigte Zeit zur Anwahl eines Menüpunkts minimiert werden kann. Die Menü-Exploration erfolgt, indem der Anwender mit dem Zeigestift auf das schaltflächenähnliche Menü drückt und mit dem Stift nach oben oder unten fährt und sich dabei durch die einzelnen Einträge bewegt. Durch doppeltes Antippen des Menüs wird der momentan gewählte Eintrag selektiert. Handelt es sich dabei um den Titel eines Untermenüs, was durch ein kleines Pfeil-Piktogramm hinter der Beschriftung verdeutlicht wird, so wird dieses anstatt des aktuellen Menüs dargestellt. Die Anwahl eines Untermenüs kann der Benutzer alternativ auch durch Ziehen des Zeigestifts nach rechts erreichen. Zum Verlassen des Untermenüs muss er den Stift entsprechend nach links bewegen.

Eine zweite Navigationsmöglichkeit richtet sich vor allem an Benutzer, welche die betreffende Anwendung häufig bedienen. Unter jedem Menüeintrag ist dazu eine eindeutige Abfolge von Farbcodes abgebildet, welche die Position der einzelnen Einträge in der Menüliste farblich sym-

bolisieren. Die Länge des Farbcodes gibt dabei die Tiefe in der Menühierarchie wieder. Durch Tippen auf eines der weiter vorne angesiedelten Farbkästchen kann der Anwender direkt zu der entsprechenden Menütiefe springen. Nach längerer Benutzung wird sich ein Wiedererkennungseffekt bei häufig angewählten Menüpunkten einstellen, und der Anwender kann die eingeprägte Farbfolge durch Auswählen der entsprechenden Farben in der Zeile oberhalb des Menükastens eingeben und auf diese Art sehr schnell einen beliebigen, ggf. tief in der Hierarchie gelegenen, Menüeintrag selektieren.

Abbildung 5.9 zeigt den Einsatz des modifizierten Menüs auf einem mobilen Endgerät, welches mittels drahtloser Netzanbindung eine Anwendung auf einem graphisch leistungsfähigeren Computer steuert [SME02].



Abbildung 5.9: Angepasste Benutzeroberfläche in der Anwendung auf einem iPAQ Pocket PC.

Abbildung 5.10 veranschaulicht einen bereits vereinfachten Evolutionsbaum, welcher sich bei der prototypischen Entwicklung der alternativen Menünavigation ergeben hat. Die Vielfalt demonstriert die einfache Austauschbarkeit und schnelle Erweiterbarkeit bestehender Menüansätze, welche daher optimal und zügig auf Benutzer- und Eingabe- sowie Ausgabe-gerätsansprüche zugeschnitten werden können. So wurden teilweise bereits während der ersten Implementierung weitere Verbesserungen zur ursprünglichen Idee hinzugefügt, weitere folgten auf entsprechende Benutzeranregungen. Die ähnliche Art der Menüpräsentation in [DE01] unterstreicht dabei, dass die gewählte Navigationsmethode nicht auf tragbare Kleincomputer limitiert und z.B. auch für großflächige stereoskopische Projektionen geeignet ist und unter Anwendung der generischen Erweiterungsmöglichkeiten leicht umgesetzt werden kann. Ebenso sind mittels

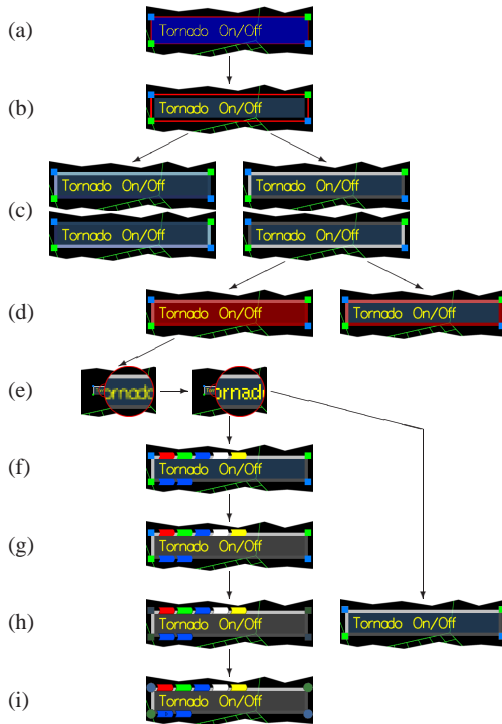


Abbildung 5.10: Evolutionsbaum der alternativen Menü-Interaktion: (a) erster Prototyp; (b) Farbnuance und Sättigung repräsentieren die Position des Eintrags sowie die Hierarchiestufe; (c) bunter sowie grauer Schaltflächenrahmen im entlasteten und gedrückten Zustand; (d) Aufblitzen bzw. dauerhafter Farbrahmen bestätigen die Selektion; (e) Bitmap-Schriften verbessern die Lesbarkeit bei kleiner Skalierung; (f) farbkodierte Abkürzungspunkte; (g) Farbnuancen und Sättigung im Menükörper verwirren häufig, daher entfernt; (h) dunkle Umrandung verbessert Erkennbarkeit; (i) Änderung der Skalierungs- und Bewegungsflächen, kleine Dreiecke weisen Untermenüs aus.

dieser Technik z.B. Anpassungen bestehender Oberflächen an die Eigenheiten bei autostereoskopischer Darstellung möglich. Es hat sich gezeigt, dass Schrift und Piktogramme aufgrund der spaltenbasierten Bildunterteilung nur sehr schwer erkennbar sind. Dieses Problem könnte in diesem Fall durch eine Verdoppelung der Schriftbreite und Piktogrammgröße leicht mittels der vorgestellten Vorgehensweise behoben werden.

5.2.2 Haptische Ein- und Ausgabe

In der Einleitung zu diesem Kapitel sowie in Kapitel 4.3.2 wurde bereits erläutert, dass die Durchführung von Interaktionsaufgaben im dreidimensionalen Raum durch die Verwendung von auf zwei Freiheitsgrade beschränkten Eingabegeräten nur umständlich erfolgen kann und mit Limitierungen verbunden ist [HDG94]. Im Folgenden werden daher Eingabegeräte vorgestellt, welche sechs oder sogar sieben Freiheitsgrade besitzen und die freie Positionierung im Raum sowie Drehungen um mehrere Achsen unterstützen. Haptische Geräte bieten dabei zusätzlich die Möglichkeit zur Erzeugung einer Gegenkraft an, welche als taktile Rückkopplung zum Anwender genutzt werden kann. Daneben existieren noch weitere Ansätze, um unter Verwendung herkömmlicher Eingabegeräte eine drei- oder mehrdimensionale Interaktionsfreiheit zu erreichen, z.B. durch den Einsatz von zwei Mäusen gleichzeitig [ZFS97], auf welche jedoch hier nicht näher eingegangen wird, da sie im Allgemeinen lediglich als provisorische Lösungen anzusehen sind (siehe [RBE04]).

In verschiedenen Veröffentlichungen wird ebenfalls die Bedeutung von mehrdimensionalen Eingabegeräten betont. In [TG98] wird zur Steuerung von Charakteranimationen eine SpaceMouse eingesetzt. Dieses in Abb. 5.11 gezeigte Gerät ist im Gegensatz zu einer herkömmlichen zweidimensionalen Maus fest auf dem Tisch platziert und es erfolgt keine direkte Abbildung der Mausexbewegung auf das Objekt, sondern die Erzeugung eines statischen, gerichteten Drucks oder einer Torsionskraft wird in eine entsprechend orientierte Bewegung umgewandelt auf ein virtuelles Objekt übertragen. In der allgemeinen Anwendung ist sie als Ergänzung zu einer zweidimensionalen Maus gedacht und wird meist linkshändig zur Modellexploration bzw. -navigation eingesetzt. Für exakte Modifikationsaufgaben ist sie jedoch weniger geeignet, da eine genaue Positionierung bzw. Reproduktion einer Bewegung nur mit sehr viel Übung gelingt. Ohne eine gewisse



Abbildung 5.11: Die SpaceMouse[®] der Firma 3Dconnexion besitzt drei translatorische und drei rotatorische Freiheitsgrade, welche durch Drücken bzw. Drehen an der zentralen Zylinderkappe angesprochen werden.

Erfahrung und eine ruhige Hand überlagern sich ansonsten meist Drehung und Positionierung. Dies lässt sich zwar vermeiden, indem – je nach Aufgabe – bestimmte Bewegungsmöglichkeiten vorübergehend deaktiviert bzw. von der Anwendung ignoriert werden, jedoch widerspricht diese Einschränkung der Forderung nach mehreren Freiheitsgraden.

Ähnlich zu bewerten sind spezielle Joysticks, welche mehrdimensionale Bewegungen erlauben. Einige besitzen jedoch zusätzlich die Möglichkeit zur Erzeugung von Gegenkräften, durch welche ein sensorischer Rückkopplungskanal zum Benutzer geöffnet wird. Eine direkte Anwendung hierfür ist die Vermeidung von Durchdringungen, d.h. würde der Anwender durch die Erzeugung einer zu großen Deformation einer FE-Oberfläche eines oder mehrere benachbarte Bauteile durchstoßen, so kann er durch Erzeugen einer entsprechend dimensionierten Gegenkraft zurückgehalten bzw. daran gehindert werden.

Schon einfache haptische Geräte, wie taktile Mäuse, können helfen, schnell und intuitiv Unebenheiten in einer Oberfläche zu erfühlen [HF96]. Da FE-Oberflächen in der Regel keine glatte Struktur besitzen, soll dieses Anwendungsgebiet für haptische Ausgabe im Folgenden nicht näher diskutiert werden.

In den vergangenen Jahren hat die Auswahl an haptischen Eingabegeräten stark zugenommen [LD03]. Aus den erwähnten Gründen sind hiervon besonders diejenigen von Interesse, die eine hohe Anzahl an Freiheitsgraden anbieten, wie sie z.B. von der Firma SensAble Technologies [Sen05] hergestellt werden. Diese und ähnliche Geräte unterstützen je nach Ausführung drei oder vier translatorische und in der Regel bis zu drei rotatorische Freiheitsgrade. In Abb. 5.12(a) rechts bzw. 5.12(b) Mitte ist ein PHANTOM Desktop Gerät zu sehen, welches sowohl die Position als auch die Ausrichtung des am vorderen Ende befestigten Interaktionsstifts bzw. Stylus erfassen kann. Wie auch dieses verfügen die meisten Geräte jedoch lediglich über die Fähigkeit, direktionale Druckkräfte aufzubauen. Die Erzeugung von Torsionskräften bleibt den High-End-Geräten dieser Familie vorbehalten. Die sensorische Rückkopplung ist ohne die Möglichkeit zur Generierung von Drehmomenten somit eingeschränkt. In Verbindung mit einem stereofähigen Anzeigegerät kann jedoch auch trotz dieser geringfügigen Einbuße von einem immersiven Arbeitsplatz gesprochen werden. Die Firma Reachin Technologies AB hat sich auf die Montage solcher Arbeitsplätze spezialisiert und bietet entsprechende Komplettpakete (Abb. 5.12(b)) für verschiedenste Anwendungszwecke an.

Um ein ungestörtes taktiles Empfinden zu ermöglichen, muss die durch das Gerät entwickelte Kraft ständig der Position des Stylus und den äußeren Kräften angepasst werden. Die typische Mindestanforderung an die Anpassungsrate liegt hierbei im Bereich von etwa 1kHz. Bei der haptischen Exploration eines Modells mit komplexer Geometrie sind daher optimierte Verfahren zur Kollisionsdetektion notwendig, um diese Anforderung erfüllen zu können. Diese Problematik wurde jedoch bereits ausreichend wissenschaftlich adressiert, und so existieren neben den Hilfsbibliotheken und der Ansteuersoftware des Herstellers zahlreiche Erweiterungen, welche für spezielle Anwendungen optimierte Lösungen bieten [JW03, JW04]. Die Ansteuerung durch den sogenannten haptischen Prozessor geschieht dabei in einem eigenen, parallel ablaufenden Programmteil, so dass diese durch das Hauptprogramm nicht ausgebremst wird. Dies bedeutet jedoch auch, dass eine direkte Aktualisierung der ertastbaren Geometrie nicht möglich ist, sondern mit Hilfe der Programmbibliothek des Herstellers erfolgen muss.

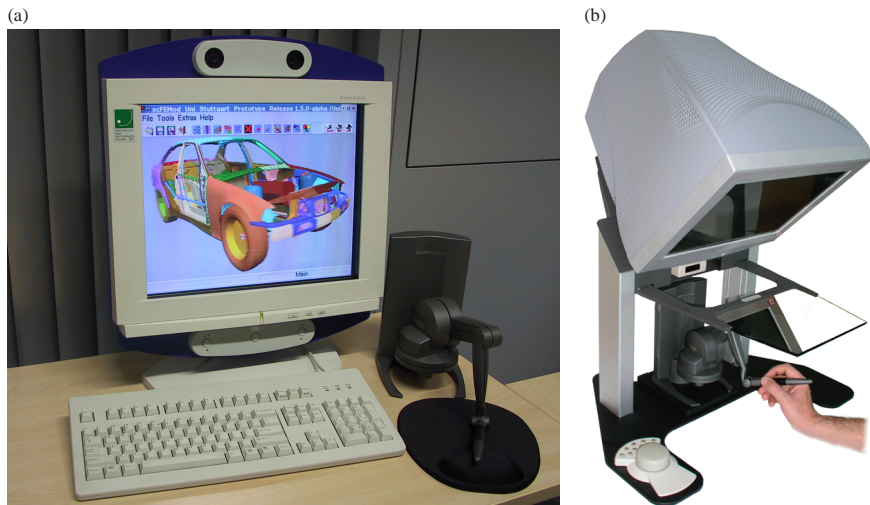


Abbildung 5.12: Immersive Arbeitsplätze: (a) autostereoskopischer Monitor (C-i 3D Display der Firma SeeReal Technologies GmbH) und haptisches Eingabegerät (PHANTOM[®] Desktop[™] der Firma SensAble Technologies, Inc.); (b) Komplettsystem der Firma Reachin Technologies AB mit halbtransparenter Anzeigefläche.

Durch Abfrageroutinen lässt sich der Zustand, d.h. die Position und die Ausrichtung des Stylus ermitteln, und dieser kann dementsprechend durch einen virtuellen Stift in der Szenerie des FE-Modells wiedergegeben werden. Mit Hilfe eines solchen Stifts gestalten sich die in Kapitel 4.3.2 erläuterten Modifikationsaufgaben deutlich einfacher und intuitiver. Eine Aufspaltung der Bewegung in mehrere Verschiebungen und Rotationen niedriger Dimensionalität ist nicht notwendig, stattdessen kann ein Punkt auf der FE-Oberfläche direkt mit der Stiftspitze ertastet und durch Drücken eines in den Stift eingebauten Schaltknopfs gegriffen werden. Dabei sind für die Wahl dieses Angriffspunkts die taktil erfahrbaren Oberflächen auf diejenigen FE-Bauteile begrenzt, welche selektierte Knoten enthalten. Andernfalls wäre es für den Anwender aufgrund andauernder Kollisionen zu mühsam und zu unübersichtlich, sich durch die Vielzahl der – um das interessierende Auswahlgebiet gruppierten – Fahrzeugkomponenten zu navigieren. An dieser Stelle sei erwähnt, dass für die Selektion einer Menge an Knoten das einfache Einkreisen mit einer zweidimensionalen Maus einer in diesem Fall vergleichsweise umständlichen Auswahl mittels dreidimensionaler Eingabemethoden vorzuziehen ist. Eine Selektion im dreidimensionalen Raum ist zwar prinzipiell leistungsfähiger, bringt jedoch im Vergleich zum zweidimensionalen Einkreisen nur dann einen Vorteil, wenn dem Betrachter abgewandte Knoten selektiert werden

sollen, was jedoch zwangsläufig in einer unzuverlässigen und nicht exakt reproduzierbaren Auswahlmenge resultiert.

Eine Bauteilmodifikation bedarf durch den Einsatz von einem virtuellen Interaktionsstift keines zusätzlichen Widgetmechanismus und die in Kapitel 4.3.2 vorgestellten Editieroperationen können direkt und intuitiv durch Ziehen und Drehen am Stylus des haptischen Eingabegeräts durchgeführt werden. Stereoskopische Projektionstechniken garantieren dabei, dass die Verschiebung im dreidimensionalen Raum entsprechend erkennbar wiedergegeben werden kann. Auch hierbei ist es zweckmäßig, die haptischen Fähigkeiten des Eingabegeräts zu nutzen, um bei der Bauteilverformung das Durchstoßen benachbarter Teile zu vermeiden, indem ständig auf Kollision mit diesen überprüft wird. Dabei werden nicht alle angrenzenden Teile in Betracht gezogen, sondern zu Beginn der Interaktion innerhalb eines bestimmten Radius die potenziellen Kollisionselemente bestimmt, wie in Abb. 5.13 gezeigt. Dabei wird der selektierte bzw. modifizierte, hier grau dargestellte Bereich nicht in diese Menge aufgenommen, da zum einen eine Kollision der deformierten Fläche mit sich selbst nur bei sinnlosen Manipulationsoperationen erfolgen kann und zum anderen eine dadurch notwendige, andauernde Aktualisierung der Kollisionsflächen den haptischen Prozessor belastet. Wird der Angriffspunkt um mehr als einen bestimmten Betrag (z.B. die Hälfte des Radius) verschoben, so wird die Menge der potenziellen Kollisionselemente entsprechend um die Umgebung der neuen Position erweitert.

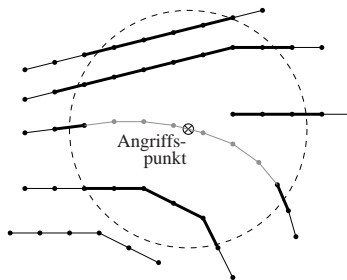


Abbildung 5.13: Die während der Modifikation des grau gezeichneten Abschnitts berücksichtigten Kollisionsflächen sind fett markiert, die Abgrenzung des dabei betrachteten, kugelförmigen Umgebungsbereichs ist gestrichelt skizziert.

Während einer Torsion oder Verschiebung des Auswahlgebiets kann es ggf. zu Durchdringungen mit benachbarten Bauteilen kommen (siehe Abb. 5.14(a)). Dies kann großteils behoben werden, indem in Durchdringungsbereichen der Verschiebungsvektor der betreffenden Knoten soweit verkürzt wird, dass dieser die Oberfläche des perforierten Bauteils nicht mehr durchstößt. Dabei kann eine geeignete Bounding-Volume-Hierarchie die Detektion durchdringungsverursachender Verschiebungen und die Auffindung der Durchstoßpunkte deutlich beschleunigen. Allerdings werden durch diesen Ansatz die Durchdringungen nur einseitig und somit nicht vollständig und in Randbereichen meist sehr mangelhaft gelöst wie Abb. 5.14(b) zu entnehmen ist.

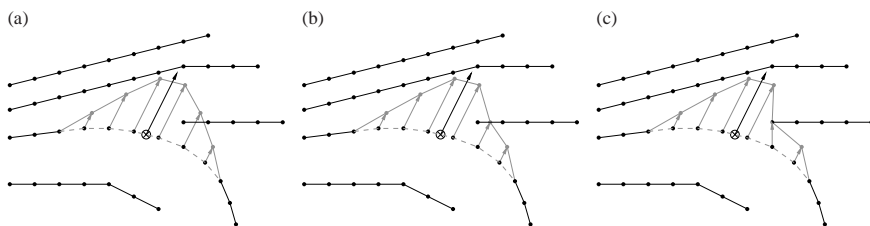


Abbildung 5.14: Durchdringungsproblematik: (a) Durch die Verformung des grauen Selektionsbereichs kommt es zur Interferenz mit dem Bauteil rechts; (b) Knotenbasierte, einseitige Bereinigung der Perforation, Elementflächen können sich jedoch ggf. weiterhin durchdringen; (c) Vollständige Aufhebung der Durchdringung bewirkt u.U. eine lokale Änderung der Deformationsrichtung.

Eine vollständige Auflösung der Durchdringung ist mit einem entsprechenden Mehraufwand verbunden [Kü01] und vermindert dadurch die Interaktivität. Gravierender in diesem Zusammenhang ist jedoch die u.U. nicht erwünschte, lokale Änderung der Deformationsrichtung und eine damit verbundene, starke Verzerrung der Elemente sowie die Einschränkung der Entscheidungsfreiheit, welches Bauteil in solchen Randgebieten nachgeben soll. Gegebenenfalls kann es bei diesem, in Abb. 5.14(c) verdeutlichten Fall, durchaus wünschenswert sein, das rechts eingezeichnete Bauteilelement nach oben zu biegen. Die Entscheidung, welcher Teil der Fahrzeugkomponenten zurückweicht, sollte daher der Expertise des Anwenders überlassen werden und nach Vollendung der Deformationsoperation in einem zweiten Schritt unter Anwendung bereits früher in der Arbeitsgruppe entwickelter, halbautomatischer Perforationsbehebungsalgorithmen [Kü01, Fri04] durchgeführt werden.

Alternativ ist es ebenso denkbar, sämtliche Durchdringungen generell zu verbieten und unter Einsatz der haptischen Möglichkeiten in einem solchen Fall die weitere Bewegung in die betreffende Richtung zu stoppen. Eine derart restriktive Vorgehensweise hat sich jedoch nicht bewährt und wirkt häufig dem gewünschten Verformungsprozess entgegen. Stattdessen ist es – auch aus Gründen einer performanten Ausnutzung des limitierten Funktionsumfangs der haptischen Prozessorbibliothek – zweckmäßig, lediglich den Verschiebungsvektor des Angriffspunkts, in Abb. 5.14 jeweils schwarz in der Mitte eingezeichnet, auf Kollision bzw. Durchdringung mit Nachbarbauteilen zu überprüfen.

Um dem Anwender dennoch eine Rückkopplung über die Qualität des Netzes zu geben, wird für jede durch die Deformation hervorgerufene Durchdringung und für jeden der in Kapitel 4.3.3 aufgeführten Vernetzungsfehler, welcher weiter hinzukommt, eine kleine, der Verformung entgegenwirkende Kraft erzeugt. Diese lässt sich durch Definition von Kraftzonen, wie sie in Abb. 5.15 skizziert sind, performant an die Softwareschnittstelle des haptischen Geräts übermitteln. Eine gewisse Ausdehnung sowie eine senkrecht zum Verformungsvektor verlaufende Ausrichtung garantieren dabei eine stabile Kraftwahrnehmung, da eine Aktualisierung dieser Kraftfelder seitens

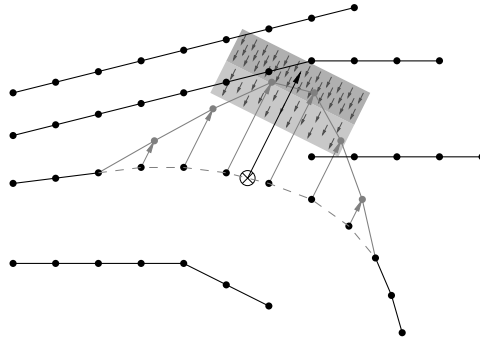


Abbildung 5.15: Gegenkräfte ermöglichen eine Rückkopplung über den aktuellen Zustand der Netzqualität.

des Hauptprogramms nicht unbedingt mit der geforderten Rate von circa 1kHz erfolgen kann. Eine entsprechende Vorberechnung der Kraftfeldbereiche, welche diese interaktive Generierung und Übermittlung der Zonen ersetzen kann, ist zwar denkbar, jedoch ohne Einschränkung der Bewegungsfreiheit nicht realisierbar, da ansonsten die Vielzahl der möglichen Deformationsfälle den Speicherbedarf sprengen würde. Mit zunehmender Verschlechterung der Netzqualität erfährt der Anwender somit eine ebenfalls zunehmende Gegenkraft und erhält dadurch neben der visuellen Anzeige der Vernetzungsfehler eine zusätzlich taktil wahrnehmbare Warnung.

Abbildung 5.16 zeigt zwei einfache Beispiele für Verformungen, welche mit einem mehrdimensionalen Eingabegerät erzielt werden können. In Abb. 5.16(a) wird eine reine Verschiebung durchgeführt, in Abb. 5.16(b) hingegen eine reine Drehung. Durch gleichzeitige translatorische und rotatorische Bewegung des Stylus können im selben Arbeitsschritt beliebig komplexe Kombinationen der beiden Deformationsarten auf direkte und intuitive Art und Weise erreicht werden.

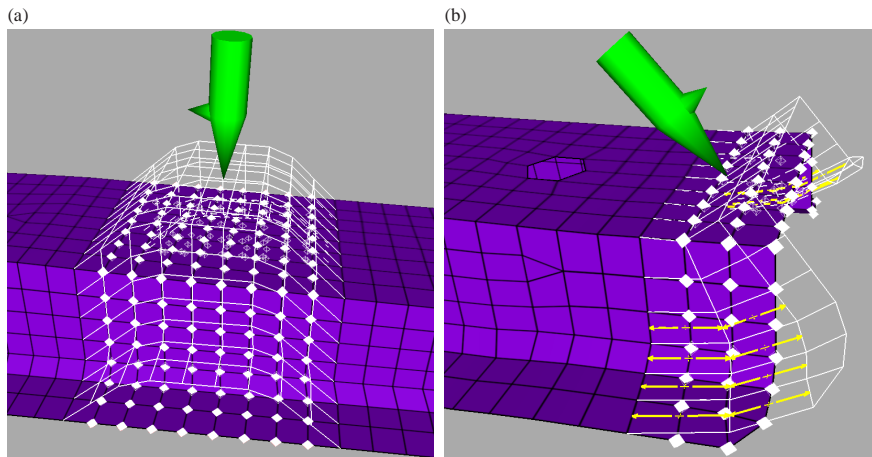


Abbildung 5.16: Deformationsoperationen mit virtuellem Stylus: (a) Erzeugen einer Beule in einem Blechträger durch Hochziehen; (b) Biegeoperation durch Drehen des Stylus und interaktive Markierung langgestreckter Elemente aufgrund der starken Verzerrung; bei Einsatz eines entsprechend ausgerüsteten haptischen Eingabegeräts können zusätzlich der Fehlermenge angemessene Rückhaltekräfte bzw. -momente erzeugt werden.

Kapitel 6

Ergebnisse

“Alright,” said Deep Thought.
“The Answer to the Great Question ...”
“Yes ...!”
“Of Life, the Universe and Everything ...”
said Deep Thought.
“Yes ...!”
“Is ...” said Deep Thought, and paused.
“Yes ...!”
“Is ...”
“Yes ...!!!...?”
“Forty-two,” said Deep Thought,
with infinite majesty and calm.

Douglas Adams: The Hitch Hiker’s Guide to the Galaxy

6.1 Zusammenfassung

In den vorangegangenen Kapiteln wurden unterschiedliche Verfahren vorgestellt, welche ein performantes, interaktives und intuitives Bearbeiten von Finite-Element-Datensätzen ermöglichen. Dabei hat sich gezeigt, dass eine gute Qualität der graphischen Darstellung trotz der Forderung nach hohen Bildwiederholraten besonders im Umgang mit Crashsimulationsmodellen wichtig ist, da hier bereits – aufgrund des in der Regel chaotischen, nichtlinearen Crashverlaufs – kleinste Abweichungen im FE-Netz große Auswirkungen im Ergebnis bewirken können. Bei der Entwicklung der in Kapitel 3 vorgestellten Ansätze zur Optimierung der graphischen Darstellung in einem FE-Werkzeug wurde deshalb besonderer Wert auf die Bildqualität und Detailtreue gelegt. Der Einsatz von Normalentexturen kristallisiert sich hier als speichereffiziente Methode zur pixelexakten Beleuchtungsberechnung auch in anderen Anwendungsgebieten (u.a. Spiele) heraus und ist einer herkömmlichen Vorgehensweise, wie sie z.B. in Standard-OpenGL-Lichtmodellen vorgesehen ist, insbesondere in der Darstellung kleiner Details und Unebenheiten in einer Oberfläche deutlich überlegen. Unter Berücksichtigung der bei FE-Modellen anfallenden Datenmengen wurde eine besonders sparsame Speicherung der Normaleninformationen gewählt, welche zudem auf eine möglichst exakte Wiedergabe der Normalenrichtung und -länge optimiert wurde. Ebenso hat es sich gezeigt, dass eine zusätzliche Hervorhebung von typischen Merkmalslinien – wie scharfen oder begrenzenden Kanten – das Verständnis der Bauteilgeometrie erleichtert und diese im Allgemeinen sogar genügend Informationen enthalten, um eine Fahrzeugkomponente eindeutig identifizieren zu können.

Um auch bei größeren FE-Modellen interaktive Bildwiederholraten zu erreichen, wurde ein spezielles Simplifizierungsverfahren entwickelt, welches die besonderen Eigenschaften von flächigen FE-Gittern gleichfalls beachtet und ausnutzt. Dabei werden sowohl qualitativ als auch quantitativ bessere Ergebnisse erzielt, als sie mit in der Praxis gängigen – jedoch nicht für FE-Netze optimierten – Simplifizierungsalgorithmen erreicht werden. Da für die Beurteilung der Qualität eines FE-Gitters neben der schattierten Darstellung der Oberfläche besonders die Umrisse der

einzelnen finiten Elemente wichtig sind, wurde eine Lösung entwickelt, welche es erlaubt, diese Umrisse auch auf einem reduzierten Netz in einer hohen Auflösungsgenauigkeit wiederzugeben. Die dazu benötigten Informationen werden – im Sinne einer hohen Speicherausnutzung – indirekt über Distanzfelder repräsentiert, welche analog zu den Normalen in Texturen abgelegt werden. Kombiniert man diese Methoden, so erlauben diese die Reduktion der Geometrie auf etwa ein Fünftel der Ausgangsdaten, ohne dass dadurch ein sichtbarer Verlust entsteht. Die dadurch gewonnene Entlastung der geometrieverarbeitenden Einheiten der Graphikkarte resultiert in einer Beschleunigung des Bildaufbaus um den Faktor drei bis vier, und der Anwender kann daher ohne Einbußen mit deutlich höheren Interaktionsraten agieren.

Diese Methoden bilden die Voraussetzungen für ein interaktives und damit effizientes Arbeiten. Um einem Ingenieur die Durchführung typischer Aufgabenbereiche weiter zu erleichtern, wurden intuitiv zu bedienende Interaktions- und Modifikationsmechanismen zur Modellbearbeitung entworfen. Hierzu wurden zunächst schnelle und fehlerunterbindende Algorithmen zur Erzeugung von linienförmigen und flächigen Verbindungen zweier Fahrzeugkomponenten vorgestellt. Die eingesetzten Interaktionsmittel wurden dabei bewusst der Definition vergleichbarer Verbindungstypen, wie Schweißpunkten, nachempfunden, so dass trotz der ggf. andersartigen Verbindungstypen ihre Handhabung konsistent ist. Des Weiteren wurde die automatische Validierung bestehender Verbindungselemente zur Ladezeit erläutert. Durch die entsprechende farbliche Hervorhebung wird der Anwender auf fehlerhafte Bereiche aufmerksam gemacht, welche er dann direkt mittels einer automatischen Kamerapositionierung ansteuern kann, um vor Ort entscheiden zu können, ob bzw. wie die bestehenden Elemente korrigiert werden müssen. Zusätzlich zur manuellen Definition von Verbindungselementen wurde eine Import-Funktion entwickelt, welche entlang analytisch beschreibender, aus CAD-Daten gewonnenen Kurvenabschnitten entsprechende Verbindungselemente unter Berücksichtigung der vorgegebenen Randbedingungen platziert. Ebenso einfach wie bei Verbindungselementen erfolgt die manuelle Definition von Sensorpunkten. Hier ist vor allem die intuitive und realitätsnahe Ausrichtung und Repräsentation des sensorinternen Koordinatensystems von Bedeutung, so dass ohne großen Aufwand eine fundierte Basis für die Analyse von Beschleunigungsverläufen oder für Vergleiche mit real durchgeführten Untersuchungen an Fahrzeugprototypen geschaffen werden kann.

Zusätzlich zu bestehenden, voll- und halbautomatischen Methoden zur Modifikation von Bauteilgeometrie, um z.B. Durchdringungen zu beheben, wurden Werkzeuge zur vollständig manuellen Deformation einer Bauteiloberfläche entwickelt. Diese erlauben zum einen die Auflösung komplexer Durchdringungsprobleme, bei denen Automatismen entweder versagen oder nicht das gewünschte Ergebnis erzielen. Zum anderen ist damit auf eine bequeme Art die Erstellung von Bauteilvarianten möglich. Im Sinne einer intuitiven Bedienbarkeit wurden spezielle Selektionsmethoden, Widgets sowie Deformationsoperationen entwickelt, welche eine schnelle und zielsichere Manipulation dreidimensionaler Objekte auch mittels zweidimensionaler Eingabegeräte unterstützen. Da bei manuellen Deformationsoperationen ggf. sehr große Verschiebungen in der Bauteilgeometrie auftreten können, mussten Strategien erarbeitet werden, um zu jeder Zeit ein simulationsfähiges FE-Netz garantieren zu können. Gegebenfalls wird daher bereits während der Durchführung manueller Operationen eine lokale Neuvernetzung der betroffenen Bereiche initiiert.

Zur hardwareseitigen Komplettierung der Interaktions- und Darstellungsmöglichkeiten wurden spezielle, für eine Arbeitsplatzanwendung geeignete, Aus- und Eingabegeräte untersucht und eine entsprechende Anbindung bzw. Unterstützung für das eingesetzte Preprocessing-Werkzeug implementiert. Es wurde erläutert, wie autostereoskopische Bildschirme den Eindruck und das Verständnis eines komplexen, dreidimensionalen Fahrzeugmodells verbessern können. Die Einbindung in eine bestehende Applikation kann dabei entweder nativ oder mittels einem generischen Ansatz erfolgen, welcher es erlaubt, jede OpenGL basierte Applikation – auch ohne direkten Zugriff auf den Quellcode – auf solch einem Ausgabegerät in Stereoprojektion anzuzeigen. Da der Zeichenaufwand bei Stereoprojektionen entsprechend steigt, lassen sich auch hier die in Kapitel 3 vorgestellten Algorithmen für einen deutlichen Performanzgewinn einsetzen.

Die Interaktion in einer virtuellen Realitätsumgebung ist u.U. nicht immer mit herkömmlichen Arbeitsplatzmöglichkeiten vergleichbar. Insbesondere die Navigation innerhalb menüartiger Strukturen kann in einer dreidimensionalen Umgebung beschwerlich sein. Auch hier können generische Ansätze eine Verbesserung der Bedienbarkeit schaffen. Die Interaktion mit dreidimensionalen Objekten hingegen wird durch entsprechende Eingabemöglichkeiten wie z.B. einem haptischen Eingabegerät mit sechs Bewegungsfreiheitsgraden stark vereinfacht und erfolgt deutlich intuitiver und direkter als mit einer für zweidimensionale Aufgaben ausgelegten Ausrüstung. Hier hat sich gezeigt, dass die Ausnutzung eines zusätzlichen Sinnes – trotz noch nicht vollständig ausgereifter Hardware – eine Verbesserung und Beschleunigung in der Durchführung bestimmter Aufgaben gegenüber einer rückkopplungslosen Manipulation bewirkt.

6.2 Bewertung

Sämtliche in diesem Rahmen präsentierten Methoden wurden zunächst in prototypischer Form in ein Preprocessing-Werkzeug integriert und großteils im praktischen Einsatz von Berechnungsingenieuren der Crashesimulationsabteilung bei BMW evaluiert. Aufgrund der intuitiven Bedienbarkeit der implementierten Interaktionsmethoden wurden diese sehr schnell akzeptiert und daher meist direkt in die kommerziell vertriebene Programmversion übernommen. Viele der im Rahmen dieser Arbeit entwickelten Ansätze haben daher bereits eine gewisse Verbreitung im Bereich der FE-Modellierung erlangt.

Im zu Beginn vorgestellten Simulations- und Entwicklungszyklus ergeben sich durch die Anwendung der beschriebenen Techniken insbesondere in der frühen Konstruktionsphase eines neuen Fahrzeugmodells deutliche Zeitersparnisse. Da hier häufig auf die FE-Netze eines Vorgängerfahrzeugmodells zurückgegriffen wird und teils durch den Einbau neuer Teile, teils durch das Abändern bestehender Fahrzeugkomponenten die Konstruktion an den neuen Fahrzeugprototypen angenähert wird, kommt es häufig zu einer Mixtur aus unterschiedlichsten Bauteilen, welche an den Naht- bzw. Verbindungsstellen und häufig auch im Bauteilinneren nicht mit ihrer Umgebung harmonieren. Daher müssen an solchen Stellen eines oder mehrere beteiligte Bauteile entsprechend angepasst werden. Bei der Wahl der für eine Modifikation am geeignetsten erscheinenden Fläche kann eine qualitativ hochwertige Darstellung und die zusätzliche Einblendung der ggf. stabilitätsentscheidenden Merkmalskanten hilfreiche Anhaltspunkte liefern. Da mit der

Entwicklung neuer Fahrzeugtypen die Komplexität der zugehörigen Simulationsmodelle schnell wächst, wird trotz der parallel zunehmenden Leistungsfähigkeit der Graphikhardware der Bedarf an den vorgestellten Simplifizierungsansätzen oder ähnlichen Methoden zur schnellen und dennoch qualitativ hochwertigen Darstellung eines Gesamtfahrzeugs gegeben bleiben.

Die Modifikationen an den gewählten Bauteilen muss dank der beschriebenen Manipulationstechniken nicht mehr in einem externen Vernetzungsprogramm durchgeführt werden, sondern kann zusammen mit der einfach zu handhabenden Definition der verschiedenen neuartigen Verbindungselemente in einer einzigen Applikation erfolgen. Ebenso ergeben sich Vorteile bei der Erzeugung einfacher Varianten, wie z.B. der Einbau zusätzlicher Versteifungssicken o.Ä., da ein langwieriger Umweg über CAD-gestützte Überarbeitungen mit anschließender Neuvernetzung entfällt. Im Vergleich zur üblichen Vorgehensweise bei der Generierung solcher Varianten in Vernetzungsprogrammen sind die präsentierten Methoden deutlich intuitiver und einfacher zu handhaben, da eine aufwändige Auftrennung des Netzes und das manuelle Einfügen neuer Elemente entfällt. Die interaktive Validierung mit sofortiger Anzeige etwaiger Netzfehler und die Möglichkeit zur automatischen lokalen Neuvernetzung verbunden mit Glättungsalgorithmen garantieren dabei den Erhalt einer hohen Netzqualität. Die innerhalb der Arbeitsgruppe teilweise parallel dazu entwickelten Methoden zur halbautomatischen Behebung von Bauteildurchdringungen behalten jedoch zur Korrektur kleinerer Fehler beim Zusammenführen prototypischer Fahrzeugkomponenten weiterhin ihre Relevanz. Die außerdem eingeführte interaktive Platzierung virtueller Sensorpunkte kann in der frühen Entwicklungsphase dazu dienen, erste wichtige Anhaltspunkte im Vergleich mit früheren Fahrzeugmodellen zu liefern.

Aufgrund der in dieser Phase auftretenden Mischung zwischen neuen und alten, teilweise modifizierten Fahrzeugteilen wird eine schnelle Orientierung innerhalb des dreidimensionalen Modells anfänglich erschwert. Die Dimensionen und Verhältnisse können während einer gewissen Einarbeitungsphase vom Anwender i.A. nicht korrekt beurteilt werden. Hier können die behandelten stereographischen Projektionstechniken große Vorteile verschaffen. Haptische Eingabegeräte mit sechs Freiheitsgraden – wie das PHANTOM Desktop Gerät – ergänzen den immersiven Arbeitsplatz und bieten aufgrund der mehrdimensionalen Eingabemöglichkeit entscheidende Vorteile gegenüber herkömmlichen zweidimensionalen Mäusen oder Ähnlichem. Auch im Vergleich zu einer ebenfalls mit sechs Freiheitsgraden ausgestatteten SpaceMouse erweist sich ein solches Eingabegerät als vorteilhaft, da es zum einen eine direktere Handhabung ermöglicht und zum anderen zu einer taktil wahrnehmbaren Darstellung der Netzqualität dienen kann. Es ist daher als ideale hardwareseitige Erweiterung der mausbasierten, widgetgesteuerten Manipulationsmethoden anzusehen.

Anhand dieses weit gefächerten Einsatzgebiets im Bereich der frühen Konstruktionsphase wird deutlich, inwieweit die im Rahmen dieser Dissertation entwickelten Techniken sich gegenseitig ergänzen und es den Anwendern ermöglichen, bisher teilweise umständlich durchzuführende Arbeiten in wesentlich weniger Zeit zu erledigen. Die im Rahmen der Forschungsprojekte „AutoBench“ und „AutoOPT“ seitens der Anwender geforderten Verbesserungen in den Bereichen Interaktivität und Handhabung großer Modelle konnten über die gesteckten Ziele hinaus erfüllt werden. Eine hohe Darstellungsqualität trotz Simplifizierung – welche ohne Beeinträchtigung der Arbeitsgänge im Hintergrund erfolgen kann – erlaubt ein flüssiges Arbeiten, und im Ver-

gleich zu bisher üblichen Ansätzen werden die Benutzer nicht damit belastet, eine aktive Rolle bei der Simplifizierung einzunehmen. Vielmehr werden die Anwender mit fortschreitendem Arbeiten lediglich die zunehmende Performanz und die verbesserte Beleuchtungsberechnung positiv bemerken.

Die unkomplizierte Unterstützung neuer Berechnungs- und Verbindungselemente ist nicht nur innerhalb der beiden Forschungsprojekte auf positive Resonanz gestoßen und die zu diesem Zweck entwickelten Interaktionstechniken haben ihren Platz innerhalb des kommerziellen Produkts „sc-FEMod“ gefestigt. Die intuitiven Möglichkeiten zum Editieren dreidimensionaler FE-Flächen und die robusten Bewertungs- und Optimierungsverfahren für diese – im Vergleich zum CAD relativ grob strukturierten – Modelle erlauben den Simulationsingenieuren nicht nur eine gewisse Unabhängigkeit von der CAD-Abteilung und dadurch kürzere und damit konkurrenzfähige Entwicklungszyklen zu erlangen; sie eröffnen vielmehr auch die Möglichkeit zu stochastischen Simulationen und bieten eine attraktive Alternative zu den üblicherweise verbreiteten und teilweise umständlich zu bedienenden Werkzeugen. Aufgrund der Einfachheit und schnellen Erlernbarkeit der präsentierten Modifikationsoperationen entspricht die durchgängige Akzeptanz den Erwartungen. Mit Verfügbarkeit der Methoden innerhalb des kommerziellen Preprocessing-Werkzeugs haben diese eine gewisse Vorreiterstellung eingenommen und wurden bereits in andere Produkte übernommen. Eine Adaption dieser Interaktionstechniken auf andere Modellierungsbereiche – z.B. CAD-Freiformflächen – bleibt dementsprechend abzuwarten.

Die im Rahmen des „AutoOPT“-Projekts untersuchten Möglichkeiten zur Nutzung neuer Hardware-Technologien haben eindeutig gezeigt, dass immersive Arbeitsplätze das Potenzial zur weiteren Vereinfachung der alltäglichen Arbeitsaufgaben von Berechnungsingenieuren bieten. Aufgrund der Neuheit und den auf den ersten Blick unrentabel erscheinenden Anschaffungskosten steht die breite Akzeptanz – vor allem auf der Ebene des Finanzmanagements – noch aus. Mit fortschreitender Entwicklung der Hardware aus erster Generation und der Einbringung der im Rahmen dieser Arbeit entstandenen Erkenntnisse und Lösungen ist jedoch auch hier ein weiterer Schritt hin zu kürzeren und Fehler unterbindenden Entwicklungszyklen in naher Zukunft absehbar. Ein dauerhafter und verbreiteter Einsatz solcher Geräte verbessert und beschleunigt dabei nicht nur den Entwicklungsprozess, sondern wirkt sich auch positiv auf das Verhältnis von Kosten zu Nutzen aus. Im folgenden Abschnitt soll auf diese Problematik nochmals kurz eingegangen werden.

6.3 Ausblick

Sowohl die Größe üblicher FE-Simulationsmodelle als auch die Leistungsfähigkeit der Graphikhardware wird mittelfristig weiterhin überproportional anwachsen. Dabei ist zu beachten, dass aktuell der Geometriedurchsatz der Graphikkarten langsamer zunimmt als ihre Komplexität. Das heißt, primär getrieben durch die Anforderungen in Computerspielen, wird die Anzahl der Textur- und Rastereinheiten sowie der Umfang ihrer Programmierbarkeit schneller steigen als der reine Vertexdurchsatz. Damit erhöht sich auch der Grad der Parallelisierung auf diesen Graphikkarten. Von diesen Verbesserungen können insbesondere die texturgestützten Visuali-

sierungsalgorithmen des vorgestellten Simplifizierungsansatzes profitieren. Dadurch sind auch langfristig performante Interaktionsraten im Umgang mit immer größeren FE-Modellen gewährleistet.

Mit kürzeren Entwicklungszyklen wird voraussichtlich auch die Forderung nach intuitiven Werkzeugen zur schnellen Variantenerzeugung und Geometrieanpassung auf Basis von FE-Netzen zunehmen. Die präsentierten Interaktionsmethoden sollten daher weiter verfeinert und den aktuellen Bedürfnissen angepasst werden, so dass auch auf FE-Netzen ein ähnlicher oder sogar höherer Bedienkomfort als im Umgang mit gebräuchlichen CAD-Softwarepaketen erreicht wird. Die vorgestellten Ansätze zur Neuvernetzung eignen sich zur Zeit lediglich für die lokal begrenzte Optimierung eines deformierten Netzes. Hier sollten ggf. weiterführende Untersuchungen in Zusammenarbeit mit Kompetenzzentren im Bereich der Vernetzungsalgorithmen angestrebt werden.

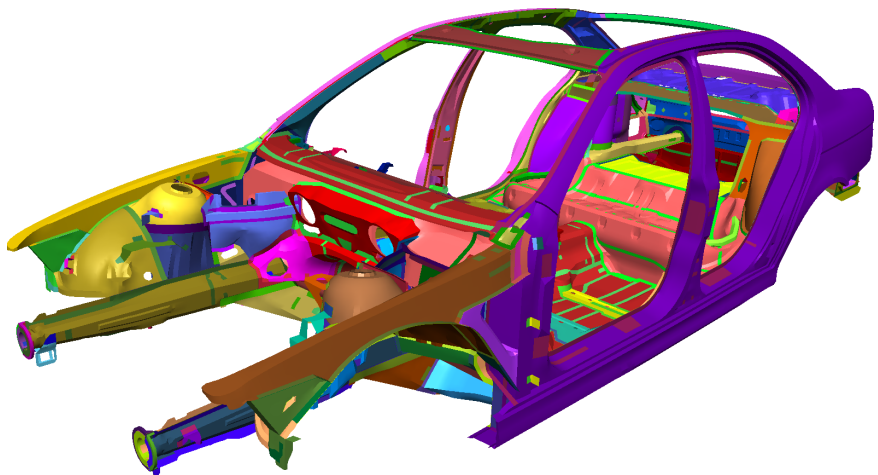
Spezielle Hardware wie autostereoskopische Monitore oder haptische Eingabegeräte befinden sich zur Zeit lediglich in Nischengebieten – z.B. in der minimal-invasiven bzw. endoskopischen Chirurgie – im Einsatz. Dies ist einerseits auf die zur Zeit noch relativ hohen Anschaffungskosten und andererseits auf Defizite aufgrund des teilweise prototypischen Produktstatus zurückzuführen. Bei haptischen Eingabegeräten ist hier die sehr limitierte Krafterzeugung, bei autostereoskopischen Anzeigegegeräten die vergleichsweise geringe Auflösung anzuführen. Der vorgestellte generische Ansatz schafft hier die Möglichkeit zur einfachen Erweiterung bestehender Anwendungen und kann so zur Förderung der Akzeptanz und Verbreitung beitragen.

Sowohl bei Graphikhardware als auch bei neuen Displaytechniken sowie haptischen Eingabegeräten wird sich der Trend zu höherer Leistung bei günstigeren Preisen voraussichtlich fortsetzen. In naher Zukunft ist zudem eine Beseitigung der erwähnten Mängel zu erwarten, so dass evtl. bald auch im konstruktiven FE-Umfeld immersive Arbeitsplätze ihren Einsatz finden werden.

Anhang A

Farbabbildungen

Kapitel 2: Berechnungsmethoden und Visualisierung strukturanalytischer Simulationen



Farbabbildung 2.9: Farbkodierung verschiedener Bauteile und Komponenten.

“What was that?” hissed Arthur.

“Something red,” hissed Ford back at him.

“Where are we?”

“Er, somewhere green.”

“Shapes,” muttered Arthur. “I need shapes.”

[...]

“What was that?” whispered Ford.

Arthur looked up.

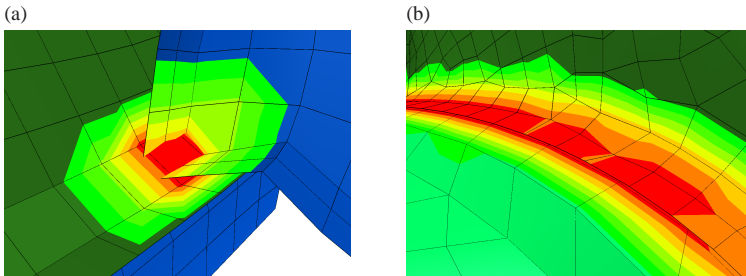
“Something blue,” he said.

“Shape?” said Ford.

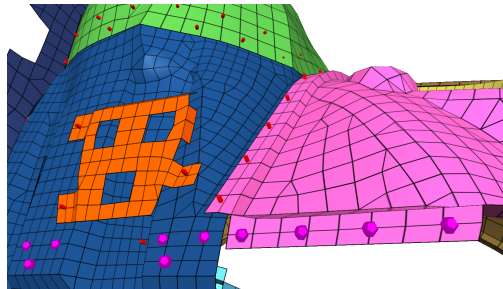
Arthur looked again.

“It is shaped,” he hissed at Ford, with his brow savagely furrowing, “like a policeman.”

Douglas Adams: Life, the Universe, and Everything

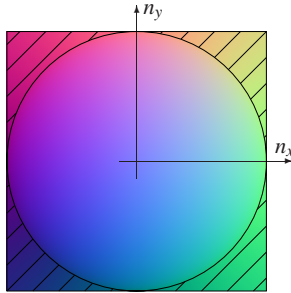


Farbabbildung 2.15: Materialdurchdringung zweier Bleche: (a) Perforation der Blechmittelebenen; (b) Penetration: Regionen, in denen der Abstand der Mittelebenen geringer als die gemittelte Blechdicke ist, sind rot texturiert.

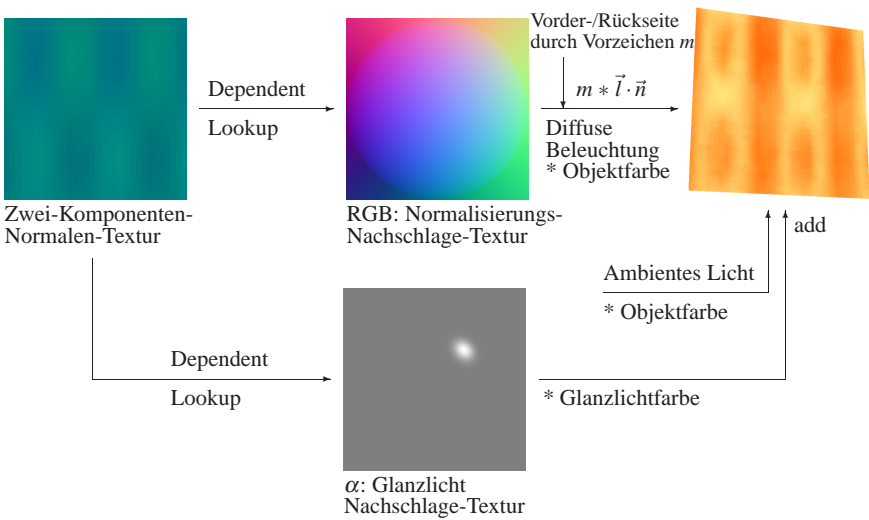


Farbabbildung 2.16: Schweißpunkte bilden die Verbindung zwischen benachbarten Bauteilen: Rote Quaderstifte stellen korrekte Schweißpunkte dar, violette Dodekaeder markieren das Fehlen einer Bauteilkomponente.

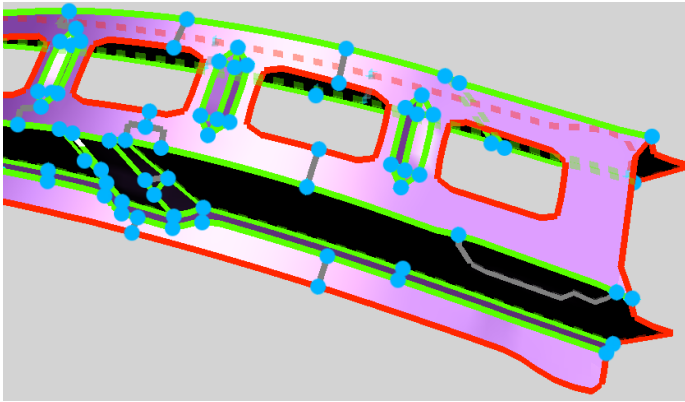
Kapitel 3: Optimierung der graphischen Darstellung



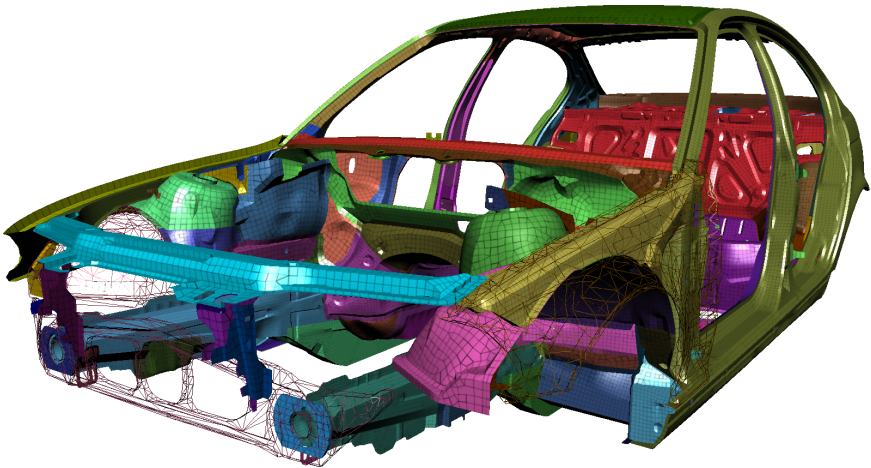
Farbabbildung 3.3: Dependent Lookup Textur zur Bestimmung der dritten Normalenkomponente bei gleichzeitiger Normalisierung.



Farbabbildung 3.4: Beleuchtungsberechnung nach dem Phong'schen Modell für eine mit einer zweikanaligen Normalentextur überzogenen Oberfläche.

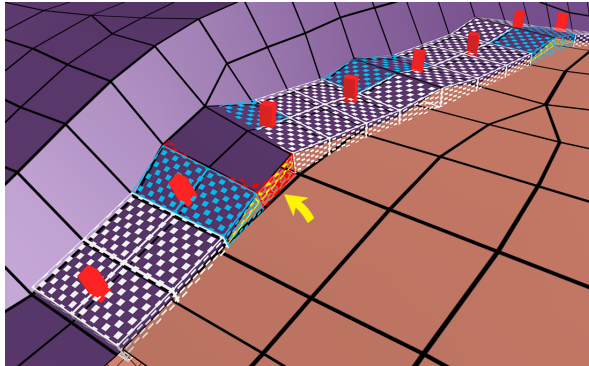


Farbabbildung 3.17(c): Gebietsgrenzen im Simplifizierungsprozess: Außenkante (rot), scharfe Innenkante (grün), vervollständigte auslaufende Kante (weiß), Untergebiets- / Ebenengrenze (grau); arretierte Kopplungspunkte (blau).

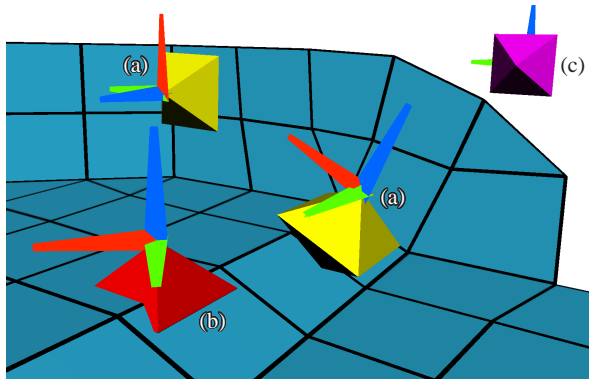


Farbabbildung 3.27: Simplifiziertes Gesamtfahrzeug mit restaurierten Elementberandungen und Phong-Beleuchtung/-Schattierung. An den in Drahtgitteransicht dargestellten Fahrzeugkomponenten im Vordergrund ist die zu Grunde liegende Simplifizierung zu erkennen.

Kapitel 4: Intuitive und interaktive Modifikationsmechanismen

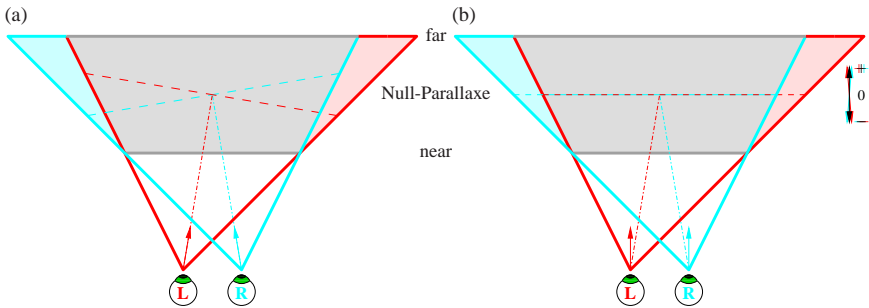


Farbabbildung 4.9: Visualisierung einer Klebeschicht. Ungültige Verbindungselemente sind farblich hervorgehoben, und um eine genauere Analyse zu ermöglichen, ist zusätzlich die exakte Lage des Klebelements gelb eingezeichnet (Pfeil).

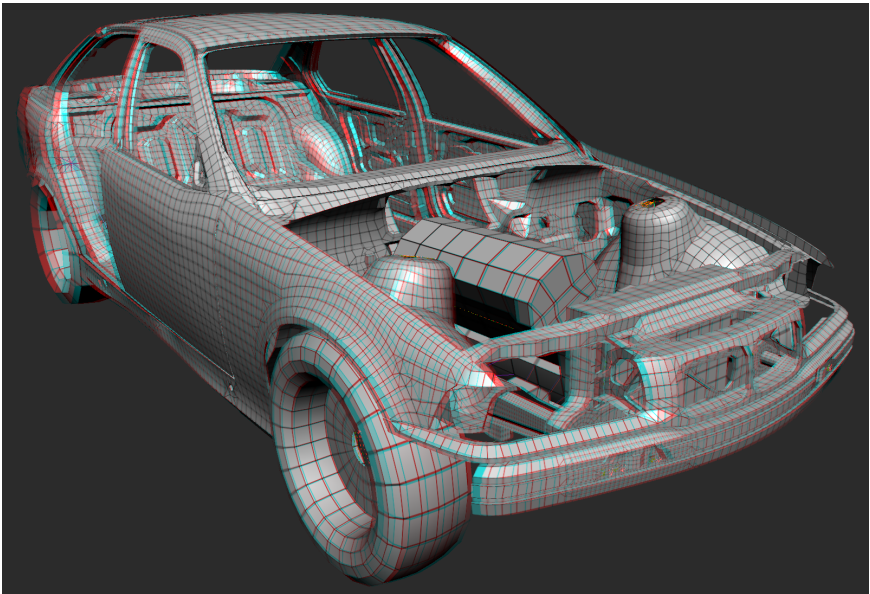


Farbabbildung 4.12: Visualisierung von Sensorpunkten: (a) korrekt platzierte Sensoren; (b) Sensorkörper (rot) durchdringt die Montagefläche; (c) Sensorpunkt (violett) mit ungültiger Bauteilreferenz.

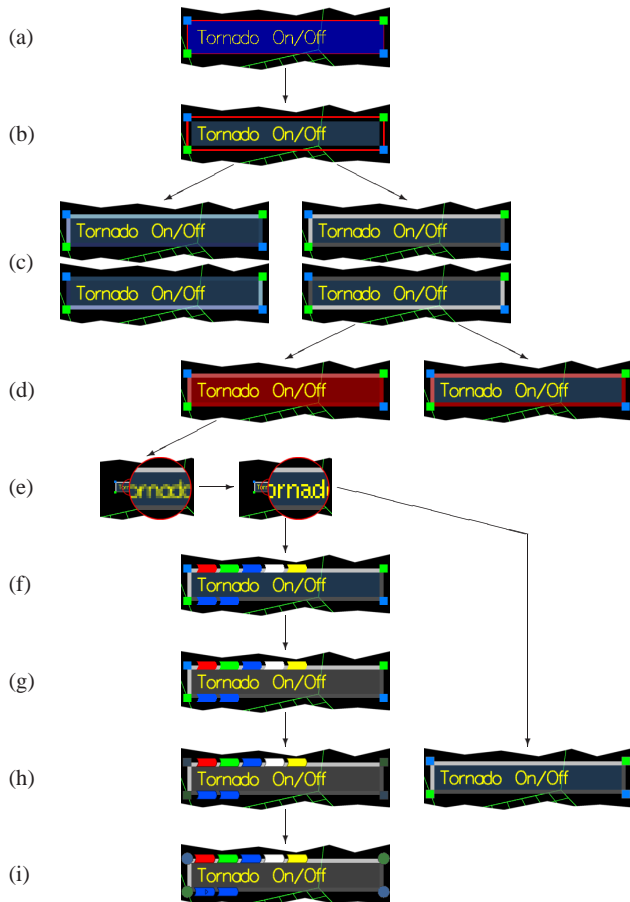
Kapitel 5: Virtuelle Realität am Arbeitsplatz



Farbabbildung 5.1: Kameraeinstellungen für die Stereoprojektion: (a) symmetrisch, auf einen Punkt fokussiert, (b) asymmetrisch.



Farbabbildung 5.7: Rot-Cyan-Anaglyph eines Fahrzeugmodells, erzeugt durch generische Erweiterung einer Anwendung um Stereofähigkeit.



Farbabbildung 5.10: Evolutionsbaum der alternativen Menü-Interaktion: (a) erster Prototyp; (b) Farbnuance und Sättigung repräsentieren die Position des Eintrags sowie die Hierarchiestufe; (c) bunter sowie grauer Schaltflächenrahmen im entlasteten und gedrückten Zustand; (d) Aufblitzen bzw. dauerhafter Farbrahmen bestätigen die Selektion; (e) Bitmap-Schriften verbessern die Lesbarkeit bei kleiner Skalierung; (f) farbkodierte Abkürzungspunkte; (g) Farbnuancen und Sättigung im Menükörper verwirren häufig, daher entfernt; (h) dunkle Umrandung verbessert Erkennbarkeit; (i) Änderung der Skalierungs- und Bewegungsflächen, kleine Dreiecke weisen Untermenüs aus.

The Hitch Hiker's Guide to the Galaxy is a wholly remarkable book. It has been compiled and recompiled many times over many years and under many different editorships. It contains contributions from countless numbers of travellers and researchers.

Douglas Adams: The Hitch Hiker's Guide to the Galaxy

Anhang B

Literaturverzeichnis

- [ABE97] AMENTA, N. ; BERN, M. ; EPPSTEIN, D.: Optimal point placement for mesh smoothing. In: *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 1997, S. 528–537
- [Arg54] ARGYRIS, J. H.: Energy Theorems and Structural Analysis. In: *Aircraft Engineering* 26 (1954), S. 347–356 (Oct.), 383–387,394 (Nov.)
- [Arg55] ARGYRIS, J. H.: Energy Theorems and Structural Analysis. In: *Aircraft Engineering* 27 (1955), S. 42–58 (Feb.), 80–94 (Mar.), 125–134 (Apr.), 145–158 (May)
- [BKS03] BENDELS, G. H. ; KLEIN, R. ; SCHILLING, A.: Image and 3D-Object Editing with Precisely Specified Editing Regions. In: *Workshop on Vision, Modelling, and Visualization VMV '03*, Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003, S. 451–460
- [Bli77] BLINN, J. F.: Models of light reflection for computer synthesized pictures. In: *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, ACM Press, 1977, S. 192–198
- [Bli78] BLINN, J. F.: Simulation of wrinkled surfaces. In: *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, 1978, S. 286–292
- [Bra03] BRAESS, D.: *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. 3. korrigierte und ergänzte Auflage. Springer Verlag, 2003
- [Bre65] BRESENHAM, J. E.: Algorithm for computer control of a digital plotter. In: *IBM Systems Journal* 4 (1965), S. 25–30
- [BRE04] BIDMON, K. ; ROSE, D. ; ERTL, T.: Intuitive, Interactive, and Robust Modification and Optimization of Finite Element Models. In: *Proceedings 13th International Meshing Roundtable*, 2004, S. 59–69

- [BS91] BLACKER, T. D. ; STEPHENSON, M. B.: Paving: A new approach to automated quadrilateral mesh generation. In: *International Journal for Numerical Methods in Engineering* 32 (1991), S. 811–847
- [BSMM97] BRONŠTEJN, I. N. ; SEMENDJAJEW, K. A. ; MUSIOL, G. ; MÜHLIG, H.: *Taschenbuch der Mathematik*. 3. überarbeitete und erweiterte Auflage. Verlag Harri Deutsch, Frankfurt am Main, Thun, 1997, S. 718–720
- [BSW83] BANK, R. ; SHERMAN, A. ; WEISER, A.: Some refinement algorithms and data structures for regular local mesh refinement. In: STEPLEMAN, R. (Hrsg.): *IE-EE First Symposium on Scientific Computing*, North Holland Publishing Company, Amsterdam, 1983, S. 3–17
- [Cam98] CAMPAGNA, S.: *Polygonreduktion zur effizienten Speicherung, Übertragung und Darstellung komplexer polygonaler Modelle*, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 1998
- [CFL28] COURANT, R. ; FRIEDRICHS, K. O. ; LEWY, H.: Über die partiellen Differenzgleichungen der mathematischen Physik. In: *Mathematische Annalen* 100 (1928), S. 32–74
- [CH02] CARR, N. A. ; HART, J. C.: Meshed atlases for real-time procedural solid texturing. In: *ACM Trans. Graph.* 21 (2002), Nr. 2, S. 106–131
- [CM93] CHORIN, A. J. ; MARSDEN, J. E.: *Texts in Applied Mathematics*. Bd. 4: *A Mathematical Introduction to Fluid Mechanics*. 3. Auflage. Springer-Verlag, 1993
- [CMS88] CHEN, M. ; MOUNTFORD, S. J. ; SELLEN, A.: A study in interactive 3-D rotation using 2-D control devices. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, 1988, S. 121–129
- [CMSR98] CIGNONI, P. ; MONTANI, C. ; SCOPIGNO, R. ; ROCCHINI, C.: A general method for preserving attribute values on simplified meshes. In: *VIS '98: Proceedings of the conference on Visualization '98*, IEEE Computer Society Press, 1998, S. 59–66
- [COM98] COHEN, J. ; OLANO, M. ; MANOCHA, D.: Appearance-Preserving Simplification. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, 1998, S. 115–122
- [CSH⁺92] CONNER, B. D. ; SNIBBE, S. S. ; HERNDON, K. P. ; ROBBINS, D. C. ; ZELEZNIK, R. C. ; VAN DAM, A.: Three-dimensional widgets. In: *Proceedings of the 1992 symposium on Interactive 3D graphics*, ACM Press, 1992, S. 183–188

-
- [CTS98] CANANN, S. A. ; TRISTANO, J. R. ; STATEN, M. L.: An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quadrilateral meshes. In: *Proc. 7th Int. Meshing Roundtable*, Sandia Nat. Lab., Oct 1998, S. 479–494
- [DE01] DACHSELT, R. ; EBERT, J.: Collapsible cylindrical trees: a fast hierarchical navigation technique. In: *Proc. IEEE Symposium on Information Visualization 2001*, 2001, S. 79–86
- [DH97] DÖLLNER, J. ; HINRICHS, K.: Object-oriented 3D Modeling, Animation and Interaction. In: *The Journal of Visualization and Computer Animation* 8 (1997), Nr. 1, S. 33–64
- [Die99] DIETRICH, S.: *Dot Product Texture Blending and Per-Pixel Lighting*. NVIDIA White Paper. 1999. – Webseite: http://developer.nvidia.com/view.asp?IO=Dot_Product_Texture_Blending
- [DLG90] DYN, N. ; LEVINE, D. ; GREGORY, J. A.: A butterfly subdivision scheme for surface interpolation with tension control. In: *ACM Trans. Graph.* 9 (1990), Nr. 2, S. 160–169
- [DS01] DOMINÉ, S. ; SPITZER, J.: *Texture Shaders*. NVIDIA Presentation. 2001. – Webseite: http://developer.nvidia.com/object/texture_shaders.html
- [EK02] EVERITT, C. ; KILGARD, M. J. *Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering*. NVIDIA White Paper. 2002
- [FDF⁺94] FOLEY, J. D. ; VAN DAM, A. ; FEINER, S. K. ; HUGHES, J. F. ; PHILLIPS, R. L.: *Computer Graphics: Principles and Practice*. New York : Addison Wesley, 1994
- [FE02] FRISCH, N. ; ERTL, T.: Deformation of finite element meshes using directly manipulated free-form deformation. In: *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, ACM Press, 2002, S. 249–256
- [Fre97] FREITAG, L.: On combining Laplacian and optimization-based mesh smoothing techniques. In: *AMD Trends in Unstructured Mesh Generation* 220 (1997), S. 37–43
- [Fri04] FRISCH, N.: *Verfahren zur Unterstützung der Arbeitsabläufe bei der Crash-Simulation im Fahrzeugbau*, Universität Stuttgart, Diss., 2004
- [FRSE02] FRISCH, N. ; ROSE, D. ; SOMMER, O. ; ERTL, T.: Visualization and Pre-processing of Independent Finite Element Meshes for Car Crash Simulations. In: *The Visual Computer* 18 (2002), Nr. 4, S. 236–249

- [GH97] GARLAND, M. ; HECKBERT, P. S.: Surface simplification using quadric error metrics. In: *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1997, S. 209–216
- [GH98] GARLAND, M. ; HECKBERT, P. S.: Simplifying surfaces with color and texture using quadric error metrics. In: *VIS '98: Proceedings of the conference on Visualization '98*, IEEE Computer Society Press, 1998, S. 263–269
- [GHSW04] GROSS, D. ; HAUGER, W. ; SCHNELL, W. ; WRIGGERS, P.: *Technische Mechanik, Band 4: Hydromechanik, Elemente der Höheren Mechanik, Numerische Methoden*. 5. Auflage. Springer Verlag, 2004
- [GLM96] GOTTSCHALK, S. ; LIN, M. ; MANOCHA, D.: OBB-Tree: A hierarchical structure for rapid interference detection. In: RUSHMEIER, Holly (Hrsg.): *SIGGRAPH'96 Conference Proceedings*, Addison Wesley, August 1996, S. 171–180
- [GP95] GRIMM, C. ; PUGMIRE, D.: Visual interfaces for solids modeling. In: *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, ACM Press, 1995, S. 51–60
- [Had98] HADLER, H.: *Evaluierung und Implementierung verschiedener Optimierungsverfahren für die effiziente Visualisierung komplexer Fahrzeugmodelle*, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diplomarbeit, 1998
- [HB99] HUNT, G. ; BRUBACHER, D.: Detours: Binary Interception of Win32 Functions. In: *Proceedings of the 3rd USENIX Windows NT Symposium*, 1999, S. 135–143
- [HD04] HASSAN, M. F. ; DODGSON, N. A.: Reverse subdivision. In: DODGSON, N. A. (Hrsg.) ; FLOATER, M. S. (Hrsg.) ; A., Sabin M. (Hrsg.): *Advances in Multiresolution for Geometric Modelling*, Springer, 2004, S. 271–283
- [HDD⁺93] HOPPE, H. ; DEROSE, T. ; DUCHAMP, T. ; MCDONALD, J. ; STUETZLE, W.: Mesh optimization. In: *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, 1993, S. 19–26
- [HDG94] HERNDON, K. P. ; VAN DAM, A. ; GLEICHER, M.: The challenges of 3D interaction: CHI '94 workshop. In: *SIGCHI Bull.* 26 (1994), Nr. 4, S. 36–43
- [HE00] HOPF, M. ; ERTL, T.: Accelerating Morphological Analysis with Graphics Hardware. In: *Workshop on Vision, Modelling, and Visualization VMV '00*, infix, 2000, S. 337–345
- [Hei91] HEIDMANN, T.: Real Shadows Real Time. In: *IRIS Universe* (1991), Nr. 18, S. 28–31

-
- [HF96] HUGHES, R. G. ; FORREST, A. R.: Perceptualisation using a tactile mouse. In: *VIS '96: Proceedings of the 7th conference on Visualization '96*, IEEE Computer Society Press, 1996, S. 181–ff.
- [HJWE00] HARROLD, J. ; JACOBS, A. ; WOODGATE, G. J. ; EZRA, D.: Performance of a Convertible 2D and 3D Parallax Barrier Autostereoscopic Display. In: *Proceedings of the SID, 20th International Display Research Conference, 2000*
- [HK94] HAN, S. H. ; KWAHK, J.: Design of a Menu for Small Displays Presenting a Single Item at a Time. In: *Proceedings of the Human Factors and Ergonomics Society*, 1994, S. 360–364
- [HMD97] HARRISON, L. ; MCALLISTER, D. ; DULBERG, M.: Stereo Computer Graphics for Virtual Reality. In: *SIGGRAPH '97, Course Notes 6* (1997)
- [HO91] HO, W. W. ; OLSSON, R. A.: An Approach to Genuine Dynamic Linking. In: *Software, Practice and Experience* 21 (1991), Nr. 4, S. 375–390
- [Hop96] HOPPE, H.: Progressive Meshes. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, 1996, S. 99–108
- [Hop97] HOPPE, H.: View-dependent Refinement of Progressive Meshes. In: *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 1997, S. 189–198
- [Hop98a] HOPPE, H.: Efficient implementation of progressive meshes. In: *Computers and Graphics* 22 (1998), Nr. 1, S. 27–36
- [Hop98b] HOPPE, H.: Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. In: EBERT, David (Hrsg.) ; HAGEN, Hans (Hrsg.) ; RUSHMEIER, Holly (Hrsg.): *IEEE Visualization '98*, 1998, S. 35–42
- [Hor02] HORMANN, K.: An Easy Way of Detecting Subdivision Connectivity in a Triangle Mesh / Department of Computer Science 9, University of Erlangen. 2002 (3). – Forschungsbericht
- [HZR⁺92] HERNDON, K. P. ; ZELEZNIK, R. C. ; ROBBINS, D. C. ; CONNER, D. B. ; SNIBBE, S. S. ; VAN DAM, A.: Interactive shadows. In: *Proceedings of the 5th annual ACM symposium on User interface software and technology*, ACM Press, 1992, S. 1–6
- [IC01] IGARASHI, T. ; COSGROVE, D.: Adaptive unwrapping for interactive texture painting. In: *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, 2001, S. 209–216

- [IIY⁺01] INOUE, K. ; ITOH, T. ; YAMADA, A. ; FURUHATA, T. ; SHIMADA, K.: Face clustering of a large-scale CAD model for surface mesh generation. In: *Computer-Aided Design* 33 (2001), Nr. 3, S. 251–261
- [JW03] JOHNSON, D. E. ; WILLEMSSEN, P.: Six Degree-of-Freedom Haptic Rendering of Complex Polygonal Models. In: *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, 2003, S. 229–235
- [JW04] JOHNSON, D. E. ; WILLEMSSEN, P.: Accelerated Haptic Rendering of Polygonal Models through Local Descent. In: *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'04)*, 2004, S. 18–23
- [Kad00] KADA, M.: *Unterteilungsverfahren zum beschleunigten Rendering von Oberflächen*, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1845, 2000
- [KHSE98] KUSCHFELDT, S. ; HOLZNER, M. ; SOMMER, O. ; ERTL, T.: Efficient Visualization of Crash-Worthiness Simulations. In: *IEEE Computer Graphics and Applications* 18 (1998), S. 60–55
- [Kil96] KILGARD, M. J.: *The OpenGL utility toolkit (GLUT) programming interface API version 3*. 1996. – Webseite: <http://www.opengl.org/developers/documentation/glut/>
- [KLS96] KLEIN, R. ; LIEBICH, G. ; STRASSER, W.: Mesh reduction with error control. In: *VIS '96: Proceedings of the 7th conference on Visualization '96*, IEEE Computer Society Press, 1996, S. 311–318
- [Kü01] KÜNZEL, J.: *Visualisierung und Beseitigung initialer Durchdringungen bei Finite Elemente Gittern*, Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1797, 2001
- [Kus98] KUSCHFELDT, S.: *Effiziente Visualisierungsverfahren zur besseren Erfassung von Crash-Simulationen im Fahrzeugbau*, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 1998
- [Kuu99] KUUTTI, K.: Small Interfaces – a blind spot of the academical HCI community. In: *Proc. HCI International '99*, Lawrence Erlbaum Associates, 1999
- [KW03] KRÜGER, J. ; WESTERMANN, R.: Linear algebra operators for GPU implementation of numerical algorithms. In: *ACM Trans. Graph.* 22 (2003), Nr. 3, S. 908–916
- [LD03] LAYCOCK, S. D. ; DAY, A. M.: Recent Developments and Applications of Haptic Devices. In: *Computer Graphics Forum* 22 (2003), Nr. 2, S. 117–132
- [Len03] LENGYEL, E.: *The OpenGL Extensions Guide*. Charles River Media, 2003

-
- [LMZ86] LÖHNER, R. ; MORGAN, K. ; ZIENKIEWICZ, O. C.: Adaptive Grid Refinement for the Compressible Euler Equations. In: *Accuracy Estimates and Adaptive Refinements in Finite Element Computations* (1986), S. 281–297
- [LPRM02] LÉVY, B. ; PETITJEAN, S. ; RAY, N. ; MAILLOT, J.: Least squares conformal maps for automatic texture atlas generation. In: *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 2002, S. 362–371
- [Mag04] MAGALLÓN, M.: *Hardware accelerated volume visualization on PC clusters*, Universität Stuttgart, Diss., 2004
- [Mei75] MEISTERS, G.: Polygons Have Ears. In: *American Mathematical Monthly* 82 (1975), S. 648–751
- [MGW98] MULDER, J. D. ; GROEN, F. C. A. ; VAN WIJK, J. J.: Pixel masks for screen-door transparency. In: *VIS '98: Proceedings of the conference on Visualization '98*, IEEE Computer Society Press, 1998, S. 351–358
- [Mis13] VON MISES, R.: Die Mechanik der festen Körper im plastischen deformablen Zustand. In: *Nachrichten der Gesellschaft der Wissenschaften, Mathematisch Physikalische Klasse, Göttingen* (1913), S. 582–592
- [Mor90] MORTENSON, M. E.: *Computer Graphics Handbook: Geometry and Mathematics*. Industrial Press, Inc., 1990
- [NC99] NOBLE, R. A. ; CLAPWORTHY, G. J.: Direct manipulation of surfaces using NURBS-based free-form deformations. In: *Proceedings of the International Conference on Information Visualization (IV 99)*, IEEE Computer Society, 1999, S. 238–243
- [NDW93] NEIDER, J. ; DAVIS, T. ; WOO, M.: *OpenGL Programming Guide*. Reading MA : Addison-Wesley, 1993
- [Nye95] NYE, A. (Hrsg.): *Volume 0: X Protocol Reference Manual*. 4. Auflage. O'Reilly & Associates, January 1995 (X Window System Series)
- [Ode87] ODEN, T.: Some historic comments on finite elements. In: *Proceedings of the ACM conference on History of scientific and numeric computation*, ACM Press, 1987, S. 125–130
- [PGGZ04] DE PAIVA GUIMARÃES, M. ; GNECCO, B. B. ; ZUFFO, M. K.: Graphical interaction devices for distributed virtual reality systems. In: *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, ACM Press, 2004, S. 363–367

- [Pho75] PHONG, B. T.: Illumination for computer generated pictures. In: *Commun. ACM* 18 (1975), Nr. 6, S. 311–317
- [PPK00] PERLIN, K. ; PAXIA, S. ; KOLLIN, J. S.: An autostereoscopic display. In: *SIG-GRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 2000, S. 319–326
- [PPK⁺01] PERLIN, K. ; POULTNEY, C. ; KOLLIN, J. ; KRISTJANSSON, D. ; PAXIA, S.: Recent Advances in the NYU Autostereoscopic Display. In: *Proceedings of the SPIE, Vol. 4297*, 2001
- [PS04] PETERS, J. ; SHIUE, L.-J.: Combining 4- and 3-direction subdivision. In: *ACM Trans. Graph.* 23 (2004), Nr. 4, S. 980–1003
- [RB93] ROSSIGNAC, J. ; BORREL, P.: Multi-resolution 3D approximations for rendering complex scenes. In: FALCIDIENO, B. (Hrsg.) ; KUNII, T. L. (Hrsg.): *Geometric Modeling in Computer Graphics*, Springer Verlag, 1993, S. 455–465
- [RBE04] ROSE, D. ; BIDMON, K. ; ERTL, T.: Intuitive and Interactive Modification of Large Finite Element Models. In: *VIS '04: Proceedings of the IEEE Visualization 2004 (VIS'04)*, IEEE Computer Society, 2004, S. 361–368
- [RE00] ROSE, D. ; ERTL, T.: Rendering Details on Simplified Meshes by Texture Based Shading. In: *Workshop on Vision, Modelling, and Visualization VMV '00*, inflix, 2000, S. 239–245
- [RE03] ROSE, D. ; ERTL, T.: Interactive Visualization of Large Finite Element Models. In: *Workshop on Vision, Modelling, and Visualization VMV '03*, inflix, 2003, S. 585–592
- [RFL⁺98] RANTZAU, D. ; FRANK, K. ; LANG, U. ; RAINER, D. ; WÖSSNER, U.: COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data. In: *Proceedings of the 2nd Workshop on Immersive Projection Technology (IPT '98)*, 1998
- [RKE01] ROSE, D. ; KADA, M. ; ERTL, T.: On-the-Fly Adaptive Subdivision Terrain. In: *Workshop on Vision, Modelling, and Visualization VMV '01*, inflix, 2001, S. 87–93
- [RL98] RANTZAU, D. ; LANG, U.: A scalable virtual environment for large scale scientific data analysis. In: *Future Gener. Comput. Syst.* 14 (1998), Nr. 3-4, S. 215–222
- [RR96] RONFARD, R. ; ROSSIGNAC, J.: Full-range approximation of triangulated polyhedra. In: *Computer Graphics Forum (Proceedings of Eurographics)* 15 (1996), Nr. 3, S. 67–76

-
- [RSD⁺02] ROBIN, V. ; SANCHEZ, A. ; DUPUY, T. ; SOIGNEUX, J. ; BERGHEAU, J. M.: Numerical Simulation of Spot Welding with Special Attention to Contact Conditions. In: H. CERJAK, H. K. D. H. B. (Hrsg.): *Mathematical Modelling of Weld Phenomena, Proceedings of the 6th International Seminar on the Numerical Analysis of Weldability* Bd. 6, 2002, S. 997–1013
- [RSR⁺03] ROSE, D. ; STEGMAIER, S. ; REINA, G. ; WEISKOPF, D. ; ERTL, T.: Non-invasive adaptation of black-box user interfaces. In: *CRPITS '18: Proceedings of the Fourth Australian user interface conference on User interfaces 2003*, Australian Computer Society, Inc., 2003, S. 19–24
- [See04] SEEREAL TECHNOLOGIES GMBH: *3D Displays*. 2004. – Webseite: http://www.seereal.com/DE/products_cnt.de.htm
- [Sen05] SENSABLE TECHNOLOGIES: *Phantom Desktop Device*. 2005. – www.sensible.com/products/phantom_ghost/phantom.asp
- [Sha49] SHANNON, C. E.: Communication in the presence of noise. In: *Proc. Institute of Radio Engineers* 37 (1949), Jan., Nr. 1, S. 10–21
- [She02] SHEFFER, A.: Model Simplification for Meshing Using Face Clustering. In: *Computer-Aided Design* 33 (2002), Nr. 13, S. 925–934
- [Sho85] SHOEMAKE, K.: Animating Rotation with Quaternion Curves. In: *SIGGRAPH'85 Conference Proceedings*, 1985, S. 245–254
- [Sila] Silicon Graphics Inc.: *OpenGL Optimizer Programmer's Guide: An Open API for Large-Model Visualization*. – Webseite: <http://techpubs.sgi.com/>
- [Silb] Silicon Graphics Inc.: *OpenGL Sample Implementation*. – Webseite: <http://oss.sgi.com/projects/ogl-sample/>
- [SME02] STEGMAIER, S. ; MAGALLÓN, M. ; ERTL, T.: A Generic Solution for Hardware-Accelerated Remote Visualization. In: *Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '02*, 2002, S. 87–94
- [Som03] SOMMER, O.: *Interaktive Visualisierung von Strukturmechaniksimulationen*, Universität Stuttgart, Diss., 2003
- [SP86] SEDERBERG, T. W. ; PARRY, S. R.: Free-form deformation of solid geometric models. In: *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, 1986, S. 151–160
- [SRE02] STEGMAIER, S. ; ROSE, D. ; ERTL, T.: A case study on the applications of a generic library for low-cost polychromatic passive stereo. In: *VIS '02: Proceedings of the conference on Visualization '02*, IEEE Computer Society, 2002, S. 557–560

- [SSGH01] SANDER, P. V. ; SNYDER, J. ; GORTLER, S. J. ; HOPPE, H.: Texture mapping progressive meshes. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, 2001, S. 409–416
- [SZL92] SCHROEDER, W. J. ; ZARGE, J. A. ; LORENSEN, W. E.: Decimation of triangle meshes. In: *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, 1992, S. 65–70
- [TCRS00] TARINI, M. ; CIGNONI, P. ; ROCCHINI, C. ; SCOPIGNO, R.: Real Time, Accurate, Multi-Featured Rendering of Bump Mapped Surfaces. In: *Computer Graphics Forum (Proceedings of Eurographics)* 19 (2000), Nr. 3, S. 119–130
- [TG98] TURNER, R. ; GOBBETTI, E.: Interactive Construction and Animation of Layered Elastically Deformable Characters. In: *Computer Graphics Forum* 17 (1998), Nr. 2, S. 135–152
- [Tur92] TURK, G.: Re-tiling polygonal surfaces. In: *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ACM Press, 1992, S. 55–64
- [VZ01] VELHO, L. ; ZORIN, D.: 4–8 Subdivision. In: *Computer-Aided Geometric Design* 18 (2001), Nr. 5, S. 397–427
- [WB00] WANT, R. ; BORRIELLO, G.: Survey on Information Appliances. In: *IEEE Computer Graphics and Applications* 20 (2000), Mai/Juni, Nr. 3, S. 24–31
- [WE98] WESTERMANN, R. ; ERTL, T.: Efficiently Using Graphics Hardware in Volume Rendering Applications. In: *Computer Graphics (SIGGRAPH '98)* 32 (1998), Nr. 4, S. 169–179
- [Wer94] WERNECKE, J.: *The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley Pub Co, 1994
- [Wri04] WRIGGERS, P.: *Nichtlineare Finite-Element-Methoden*. Springer Verlag, 2004
- [YK04] YASUI, Y. ; KANAI, T.: Surface Quality Assessment of Subdivision Surfaces on Programmable Graphics Hardware. In: *International Conference on Shape Modeling and Applications 2004 (SMI'04)*, 2004, S. 129–138
- [ZFS97] ZELEZNIK, R. C. ; FORSBERG, A. S. ; STRAUSS, P. S.: Two pointer input for 3D interaction. In: *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM Press, 1997, S. 115–120
- [Zö04] ZÖLLNER, R.: *Topologieerhaltende Deformationsoperationen auf Finite Element-Oberflächen*, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 2126, 2004

Anhang C

Lebenslauf

Dirk Rose

geboren am 29. Juni 1974 in Heilbronn-Neckargartach
ledig, keine Kinder

Schulbildung:

1980 – 1984	Grundschule, Neckarsulm
1984 – 1993	Gymnasium, Neckarsulm
	Abschluss: Abitur

Hochschulbildung:

1993 – 1999	Studium der Luft- und Raumfahrttechnik, Universität Stuttgart Abschluss: Diplom-Ingenieur
1995 – 1996	Praktikum bei der Deutschen Forschungsanstalt für Luft- und Raumfahrttechnik, DLR, Lampoldshausen
1998 – 1999	Wissenschaftliche Hilfskraft am Institut für Computeranwendun- gen, Abteilung Computersimulation und Visualisierung, Universität Stuttgart
1999 – 2005	Wissenschaftlicher Mitarbeiter am Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart