Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Entwicklung eines graphischen Editors für neurometrische Untersuchungen komplexer Oberflächen auf der Basis von Polygonnetzen

Diplomarbeit

Leipzig, April, 1999

vorgelegt von

Nowotka, Thomas

Betreuer: Privatdozent Dr. Ralf Der
Universität Leipzig

Dr. Frithjof Kruggel
Max-Planck-Institut für neuropsychologische Forschung Leipzig

Zusammenfassung

Mit der Entwicklung von Verfahren zur Erzeugung dreidimensionaler Bilder ergeben sich für die Bildverarbeitung in der Medizin neue Forschungsgebiete. Bei Datensätzen, die die Hirnoberfläche darstellen, steht neben der Visualisierung die Auswertung der Datensätze im Interesse der Forschung. Die Auswertung von Oberflächendaten, die in Form von Polygonnetzen vorliegen, umfaßt unter anderem die Segmentierung und die Vermessung von Abschnitten der Hirnoberfläche. Eine Automatisierung der Segmentierung ist nur in begrenztem Umfang möglich, daher ist eine manuelle Bearbeitung der Oberflächendaten wünschenswert. Für das Max-Planck-Institut für neuropsychologische Forschung wurde ein Programmodul entwickelt, mit dem die Segmentierung und die Vermessungen in einer interaktiven Umgebung durchführbar sind. Das Programmodul wurde in die dort entwickelte Visualisierungsoberfläche *IPE* integriert.

In der Diplomarbeit wurde ein Verfahren zur Polygondatenreduktion implementiert und näher untersucht. Verfahren zur Erzeugung von Oberflächennetzen wie z.B. der Marching Cubes Algorithmus erzeugen oft Datensätze mit einer sehr großen Anzahl an Polygonen. Solche Datensätze sind für eine Visualisierung oder für die Weiterverarbeitung schlecht geeignet. Mit Hilfe von Verfahren zur Polygondatenreduktion kann die zu verarbeitende Datenmenge verringert werden.

Danksagung

Hiermit möchte ich allen Personen danken, die beim Gelingen dieser Diplomarbeit maßgeblich beteiligt waren.

Besonders danke ich meinem Betreuer Herrn Dr. Frithjof Kruggel für die Möglichkeit, diese Diplomarbeit im Max-Planck-Institut für neuropsychologische Forschung anzufertigen. Seine hilfreichen Erklärungen und Hinweise ermöglichten mir die medizinische Bedeutung verschiedener Problemstellungen zu erkennen. Weiterhin danke ich dem Max-Planck-Institut für neuropsychologische Forschung für das hervorragende Arbeitsumfeld. Ein besonderer Dank gilt meinem Betreuer Herrn PD Dr. Ralf Der, der immer ein offenes Ohr für meine Probleme hatte und mich mit zahlreichen Hinweisen unterstützt hat.

Herrn Uwe Neubert möchte ich für Korrektur- und Verbesserungsvorschläge danken.

Vor allem danke ich aber meinen Eltern, die durch ihre fortwährende Unterstützung diese Arbeit ermöglicht haben.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die graphische Datenverarbeitung ist in vielen Bereichen der Wissenschaft ein zentrales Instrument für die Forschung geworden. Begünstigt wurde das durch die Entwicklung von immer leistungsfähigeren und schnelleren Rechnern. Durch die Visualisierung von Daten können die den Daten zugrundeliegenden Gesetze und Eigenschaften besser verdeutlicht werden.

Mit der Entwicklung von Verfahren zur Erzeugung von dreidimensionalen Bildern ergeben sich für die Bildverarbeitung in der Medizin neue Forschungsgebiete. Bekannte Verfahren sind 3D-Scanner, Magnetresonanztomographie (MRT) und Computertomographie (CT). 3D-Scanner erzeugen kein vollständiges 3D-Bild, sondern tasten nur die äußere Oberfläche eines Körpers ab. Die MR-Technik, deren Grundlagen Mitte der vierziger Jahre entdeckt wurden, ist ein Verfahren, welches direkte Messungen in 3D erlaubt. In der Medizin hat die Verwendung von technischen bildgebenden Verfahren eine lange Tradition. Die Auswertung dreidimensionaler Daten unterscheidet sich in der von zweidimensionalen Daten erheblich. Während man sich bei Schichtdaten schnell einen Überblick verschaffen kann, ist ein 3D-Datensatz vom Betrachter in seiner Gesamtheit nicht auf einmal zu erfassen. Daher wird auf Schnitte bzw. Projektionen zurückgegriffen. 3D-Datensätze können als Volumendaten oder Oberflächendaten repräsentiert werden. Im Max-Planck-Institut für neuropsychologische Forschung werden 3D-Datensätze intensiv für neurometrische Untersuchungen verwendet.

Zur Visualisierung und Verarbeitung von medizinischen Datensätzen wurde im Max-Planck-Institut für neuropsychologische Forschung das Programm *IPE* – "Image Processing Environment" – entwickelt. Mit ihm ist es möglich, viele der im Max-Planck-Institut für neuropsychologische Forschung erzeugten Datensätze darzustellen und auszuwerten. Die Datensätze liegen in verschiedenen Repräsentationen (Voxelbilder, Signaldaten (EEG) und Oberflächendaten) vor. Für die Auswertung von Oberflächendaten, die in Form von Polygonnetzen vorliegen, existieren noch keine geeigneten Programmodule. Aufgabe der Diplomarbeit ist es daher, für eine interaktive Umgebung geeignete Algorithmen für einen Editor, der mit Polygonnetzen arbeitet, zu entwickeln und zu implementieren.

Einzelnen Teilen der Hirnoberfläche lassen sich bestimmte Funktionen zuordnen (z.B. Sprachzentrum). Die relative Lage der anatomischen Regionen auf der Hirnoberfläche ist durch medizinische Forschungen auf Einzelfallbasis gut bekannt und kartiert. Die Vermessung dieser Regionen in einer Population steht seit langem im Interesse der Forschung, konnte aber bis heute nicht in größerem Umfang durchgeführt werden. In den Anfängen der Hirnforschung war die Vermessung extrem aufwendig und zudem nur am extrahierten Gehirn möglich. Heute ist es zwar möglich, Schnittbilder von lebenden Personen aufzunehmen, aber die Verarbeitung der Bilder kann erst durch den Einsatz von Rechentechnik effektiv durchgeführt werden. Bisher werden die Messungen an zweidimensionalen Schnittbildern der Hirnoberfläche vorgenommen. Die Auswertung von Polygonnetzen am Computer zur Analyse der Hirnoberfläche erlaubt, diese Messungen qualitativ und quantitativ besser durchzuführen. Ein Vorteil liegt in der relativ einfachen Handhabbarkeit der Oberflächennetze, die es erlauben, Messungen an einer größeren Anzahl von Hirnoberflächen in einer akzeptablen Zeit durchzuführen. Ein anderer Vorteil liegt in der größeren Genauigkeit, mit der Messungen durchgeführt werden können, da die Polygonnetze, mit denen die Hirnoberfläche dargestellt wird, die Hirnoberfläche sehr gut approximieren. Der Editor soll es erlauben, diese Regionen aus einem Datensatz zu extrahieren und zu vermessen.

Verfahren zur Erzeugung von Polygonnetzen (z.B. der "Marching Cubes" Algorithmus; [FDFH96]) führen bei Anwendung auf Gehirndatensätze zu Oberflächendarstellungen mit mehreren Millionen Dreiecken. Eine Handhabung solcher Netze ist sehr ressourcenintensiv. Eine Visualisierung ist in akzeptabler Zeit nicht durchführbar. Für eine interaktive Umgebung spielt die erzielbare Bildrate bei der Darstellung des Polygonnetzes eine Rolle, wobei 20-30 Bilder je Sekunde wünschenswert sind. Da bei der Darstellung am Bildschirm ohnehin nur einige 100000 Pixel benutzt werden, kann ein Netz ohne sichtbaren Informationsverlust vereinfacht werden. Eine Vereinfachung ist möglich, da bei Polygonnetzen oft größere Flächenabschnitte mit geringer Krümmung vorhanden sind, die mit weniger Polygonen beschrieben werden können. Laufzeitintensive Programme profitieren von einer Reduktion der Polygonanzahl. Speicherplatz und Übertragungszeiten in Netzen werden ebenfalls verkleinert.

Kapitel 2

Grundlagen

2.1 Repräsentation von Flächen

Flächen werden in der Computergraphik oft zur Darstellung von volumenbehafteten Objekten der realen Welt benutzt. Eine direkte Visualisierung der Volumina ist wesentlich aufwendiger, als die Visualisierung der Objekte mit Hilfe von Oberflächen. Die dargestellten Flächen repräsentieren Grenzschichten des Objektes, oft nur die äußere Begrenzung. Um beliebige in der Natur vorkommende Flächen im Rechner darstellen zu können, wird eine geeignete Repräsentation benötigt. Eine direkte mathematische Beschreibung ist aber nur für wenige Objekte wie z.B. Kugeln vorhanden.

Da es in der vorliegenden Arbeit vorwiegend um Flächen geht, wird zunächst der Begriff einer Fläche im $\mathbb{R}^3$ etwas präzisiert.

Definition 2.1.1 (Flächenstück) *Sei $\mathcal{G}$ ein Gebiet des $\mathbb{R}^2$ und $(u, v) \in \mathcal{G}$. Weiterhin seien drei Funktionen f_i , $i = 1, 2, 3$ über $\mathcal{G}$ definiert. Dann bilden die Punkte*

$$P = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad mit \quad x_i = f_i(u, v)$$

ein Flächenstück im $\mathbb{R}^3$. $x_i = f_i(u, v)$ wird dabei als Parameterdarstellung bezeichnet.

Definition 2.1.2 (Fläche) *Eine zusammenhängende Punktmenge $\mathcal{F}$ im $\mathbb{R}^3$ heißt Fläche, wenn es zu jedem Punkt p aus $\mathcal{F}$ eine Umgebung $\mathcal{U}$ in $\mathcal{F}$ gibt, so daß die in $\mathcal{U}$ liegenden Punkte von $\mathcal{F}$ eine Parameterdarstellung als Flächenstück haben.*

Definition 2.1.3 (Flächennormale) *Gegeben sei eine Fläche $\mathcal{F}$ mit dem Punkt $P(x_1, x_2, x_3) = P(u, v)$. Falls $\frac{\partial \vec{x}}{\partial u}$ und $\frac{\partial \vec{x}}{\partial v}$ existieren und linear unabhängig sind, dann ist*

$$\vec{n} = \frac{\frac{\partial \vec{x}}{\partial u} \times \frac{\partial \vec{x}}{\partial v}}{|\frac{\partial \vec{x}}{\partial u} \times \frac{\partial \vec{x}}{\partial v}|}$$

die Flächennormale im Punkt P.

Flächennormalen sind in der Computergraphik wichtig für die Darstellung einer Fläche, da mit ihnen der Einfluß des Lichteinfalls berechnet wird.

Definition 2.1.4 (Orientierbarkeit von Flächen) *Sei $\mathcal{F}$ eine Fläche und $\vec{n}_0$ die zum Punkt p_0 gehörende Flächennormale. Durch Bezeichnung der Richtung von $\vec{n}_0$ als positiv, wird eine Orientierung der Fläche festgelegt. Diese Orientierung läßt sich stets auf eine hinreichend kleine Umgebung von p_0 eindeutig und stetig übertragen. Ist diese Übertragung auf ganz $\mathcal{F}$ möglich, dann heißt $\mathcal{F}$ orientierbar [GKN90].*

Ein Beispiel für eine nicht orientierbare Fläche ist das *Möbiusband*. Es kann leicht durch Zusammenkleben eines Papierstreifens hergestellt werden, bei dem ein Ende verdreht wurde. Die in der Definition angesprochene stetige Übertragung von $\vec{n}_0$ auf die gesamte Fläche ist bei einem Möbiusband nicht möglich. Die stetige Fortsetzung der Normale längs einer geschlossenen Kurve auf der Fläche führt bei Rückkehr zum Ausgangspunkt nicht in jedem Fall zum gleichen Vektor $\vec{n}_0$, sondern u.U. zu einem entgegengesetzten Vektor, das steht aber im Widerspruch zur Eindeutigkeit. Beim Trennen einer Fläche entlang eines vorgegebenen Pfades muß der Fall einer nicht orientierten Fläche berücksichtigt werden (Kapitel 2.3.1).

In Programmen zur Modellierung von dreidimensionalen Szenen wird oft das Prinzip der *konstruktiven Geometrie fester Körper* (*constructive solid geometry*, CSG) verwendet [FDFH96]. Grundkörper, meist Quader, Kugel, Torus und Zylinder, werden mit Hilfe der booleschen Mengenoperatoren Vereinigung, Durchschnitt und Differenz zu komplexeren Körpern zusammengesetzt, mit denen weitere Körper zusammengesetzt werden können (Abbildung 2.1). Der Aufbau eines Körpers kann als Baum von Grundkörpern und booleschen Operationen repräsentiert werden. Die Grundkörper lassen sich in ihrer Form verändern, z.B. kann ein Kegel aus einem Zylinder durch Verkleinern der Deckfläche

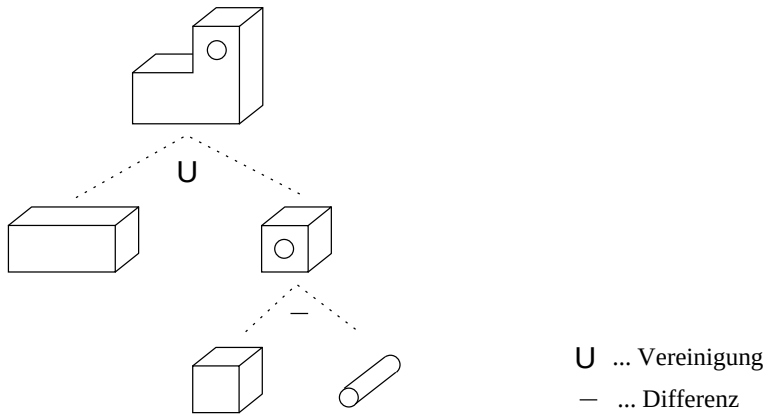


Abbildung 2.1: Beispiel für ein CSG-Objekt.

des Zylinders konstruiert werden. Für die meisten Anwendungen ist diese Methode aber ungeeignet, um reale Objekte zu beschreiben, da die Darstellung von beliebigen in der Natur vorkommenden Objekten sehr schwierig und aufwendig ist.

Durch das Zusammensetzen einer Fläche aus einfachen Flächenstücken läßt sich eine Originalfläche beliebig genau annähern. Die einzelnen Flächenstücke können dabei als einzelne auf einen eingeschränkten Parameterbereich begrenzte mathematische Funktionen angesehen werden. Für die einzelnen Flächenstücke werden häufig Polygone oder komplexere Freiformflächen verwendet. Setzt man eine Fläche zusammenhängend aus mehreren Polygonen zusammen, erhält man ein Polygonnetz, welches die am weitesten verbreitete Art ist, Flächen zu repräsentieren. Nachfolgend werden Polygonnetze und Freiformflächen kurz vorgestellt.

2.1.1 Polygonnetze

Definition 2.1.5 (Polygonnetz) *Ein* Polygonnetz[1] *ist eine stückweise lineare Oberfläche, welche aus einer Menge von planaren Polygonen besteht, die an ihren Kanten zusammengefügt sind.*

Polygonnetze eignen sich für die Darstellung beliebiger Oberflächen. Quader, Pyramiden und andere durch lineare Flächen begrenzte Körper können durch ein Polygonnetz exakt

[1] Definition für Polygone siehe Kapitel 2.2.1.3

repräsentiert werden. Die meisten darzustellenden Körper lassen sich durch Polygonnetze jedoch nur annähern und nicht exakt nachbilden. Für eine sehr genaue Approximation sind entsprechend viele Polygone notwendig. Polygonnetze lassen sich aus Polygonen beliebigen Grades aufbauen. In der Praxis werden als Polygone häufig Dreiecke verwendet. Dafür gibt es mehrere Gründe: Da Dreiecke im euklidischen Raum immer planar und konvex sind, lassen sich Algorithmen für Dreiecksnetze einfach und stabil implementieren. Weiterhin ist die Hardwareunterstützung für die Darstellung von Dreiecksnetzen am Rechnerbildschirm inzwischen etabliert und effizient. Neben Workstations besitzen auch immer mehr Personalcomputer eine Hardwareunterstützung für 3D-Darstellungen.

Polygonnetze werden meist durch automatisierte Verfahren erzeugt, z.B. mit Hilfe von 3D-Scannern oder durch Flächenextraktion aus Voxelbildern ("Marching Cubes" [FDFH96], "Wrapper Algorithm" [GR94]). Im Max-Planck-Institut für neuropsychologische Forschung wird zur Erzeugung von Oberflächen aus Voxelbildern [GR94] verwendet. Dabei wird eine Grenzschicht aus einem Voxelbild in eine Oberfläche umgewandelt. Die Komplexität der dabei erzeugten Oberflächen ist sehr hoch (Anzahl der Primitive (Dreiecke), Topologie). Eine manuelle Modellierung von Oberflächen mit Hilfe von Polygonnetzen ist sehr aufwendig, da eine Vielzahl von Primitiven zu bearbeiten sind, und die Anzahl der Primitive bei besserer Approximation stark zunimmt[2]. Eine interessante Methode, dem Problem der großen Anzahl von Primitiven zu begegnen, stellt das *Resolution-Modeling* dar, mit dem Polygonnetze in verschiedenen Qualitätsstufen bearbeitet werden können. Die verschiedenen Qualitätsstufen des Polygonnetzes werden mit Hilfe einer Darstellung eines Polygonnetzes in einem Multi-Resolution-Schema erzeugt. Wesentliche Merkmale der Methode sind selektive Verfeinerungen und selektive Vereinfachungen von Polygonnetzen, die Manipulationen an gröberen Netzen erlauben, welche sich dann auf das Gesamtnetz auswirken. Ein Beispiel für diese Methode ist in [CMRS98] zu finden. Das Verfahren des *Resolution-Modeling* ist sehr aufwendig und konnte daher im Rahmen der Diplomarbeit nicht verwendet werden.

[2]Die Vergrößerung der Datenmenge hängt vom verwendeten Verfahren ab.

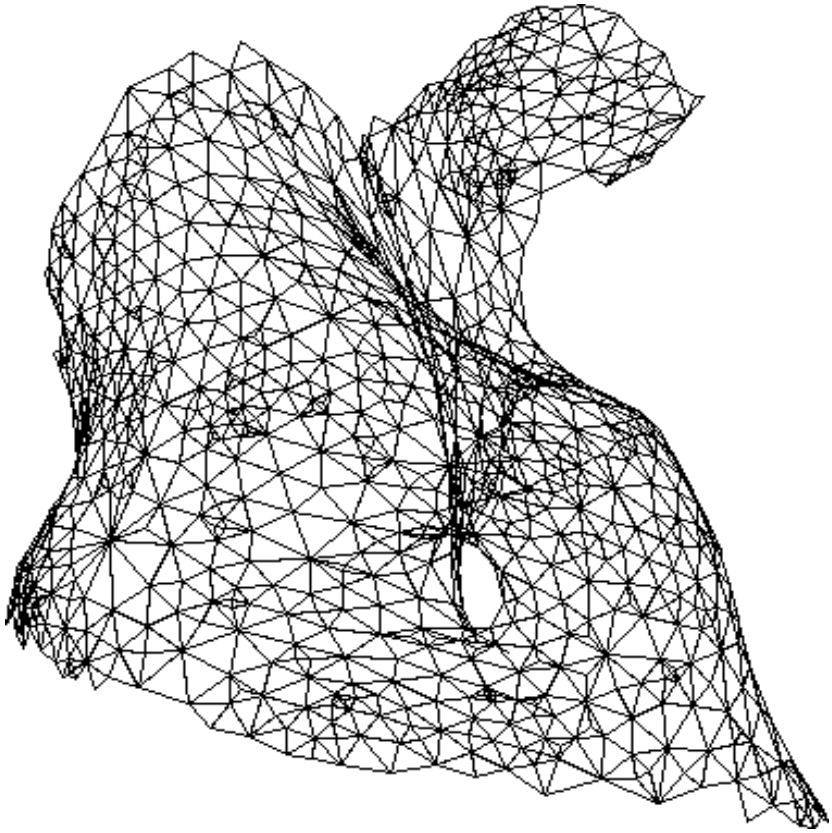


Abbildung 2.2: Beispiel für ein Polygonnetz

2.1.2 Freiformflächen

Freiformflächen sind Parameterflächen, die auf der Basis von Polynomen gebildet werden. Freiformflächen erlauben die Modellierung beliebig geformter Körper. Mit ihnen lassen sich bessere Approximationen von Flächen als mit Polygonnetzen erreichen. Bei Vergrößerungen bleibt die Oberfläche der Objekte geschmeidig, während bei Polygonnetzen die einzelnen Primitive deutlich sichtbar werden. Bei der Konstruktion von Fahrzeugen werden Freiformflächen seit vielen Jahren verwendet. Ein großer Teil der Forschungen zu Freiformflächen wurde von der Fahrzeugindustrie durchgeführt.

Die flexibelste Form von Freiformflächen sind NURBS (Nicht-Uniforme Rationale Bézier-Splines) Flächen [FDFH96], [Str96], [Hec94]. Eine NURBS-Fläche kann durch wenige

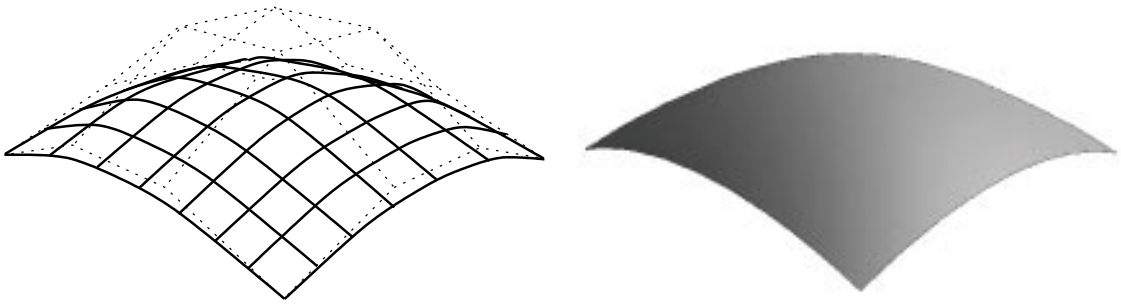


Abbildung 2.3: Beispiel für eine NURBS-Fläche mit einem Netz von 4×4 Kontrollpunkten

Kontrollpunkte beschrieben werden. Ein Kontrollpunkt beschreibt die Parameter der zugrundeliegenden Basisfunktionen. Ein Kontrollpunkt beeinflußt dabei nur einen begrenzten Bereich der NURBS-Fläche[3]. Die Menge der Kontrollpunkte von NURBS ist invariant gegenüber affinen und perspektivischen Abbildungen. Damit sind NURBS für alle wesentlichen geometrischen Transformationen geeignet. Die Verformung von NURBS ist besonders einfach und erfolgt durch Manipulation von einzelnen Kontrollpunkten. Die Beschreibung einer Fläche mit Hilfe einer einzigen NURBS-Fläche ist im allgemeinen nicht möglich. Ein Zusammensetzen aus mehreren NURBS-Abschnitten (Patch) ist dann erforderlich. Die Änderung der Topologie von NURBS-Flächen, wie z.B. das Schneiden von Löchern in einzelne oder mehrere NURBS-Abschnitte ist dagegen relativ aufwendig. Zwischen zwei NURBS-Abschnitten können durch entsprechende Wahl der Kontrollpunkte stetige Übergänge höherer Ordnung erreicht werden, wodurch im Gegensatz zu Polygonnetzen keine Kanten zu sehen sind. An diesen Übergangsstellen haben die Ableitungen auf den NURBS-Abschnitten bis zu einer bestimmten Ordnung (meist 1. oder 2.) den gleichen Wert.

Zur Darstellung von NURBS am Rechnerbildschirm wird eine sogenannte *Tesselation* (Zerlegung einer komplizierteren Fläche in mehrere einfache Flächen, üblicherweise Dreiecke) durchgeführt. Als Ergebnis der Tesselation erhält man ein Polygonnetz.
Die vorliegende Arbeit befaßt sich ausschließlich mit Polygonnetzen.

[3] abhängig vom Grad der benutzten Basisfunktionen

2.2 Geometrische Grundlagen

In diesem Kapitel werden die wichtigsten geometrischen Grundlagen der im Editor verwendeten Verfahren beschrieben.

2.2.1 Repräsentation geometrischer Objekte im Rechner

2.2.1.1 Vertex

Definition 2.2.1 (Vertex) *Ein Vertex v dient zur Repräsentation eines (Daten)punktes im dreidimensionalen Raum. Er beinhaltet die Raumkoordinaten, sowie optional zusätzliche Eigenschaften:*

- *eine Vertexnormale*
- *Farbinformation*
- *Kurvatur*

Da innerhalb eines Polygonnetzes alle Vertizes den gleichen Typ haben sollen, kann ein Feld fester Länge diese Informationen aufnehmen.

2.2.1.2 Vertexnormale

Ein Vertex kann eine *Vertexnormale* besitzen. Eine Vertexnormale gibt die Normale der durch das Polygonnetz dargestellten Fläche an der Position des Vertexes an, und steht senkrecht zu dieser Fläche (Abbildung 2.4). Kann die Vertexnormale nicht durch

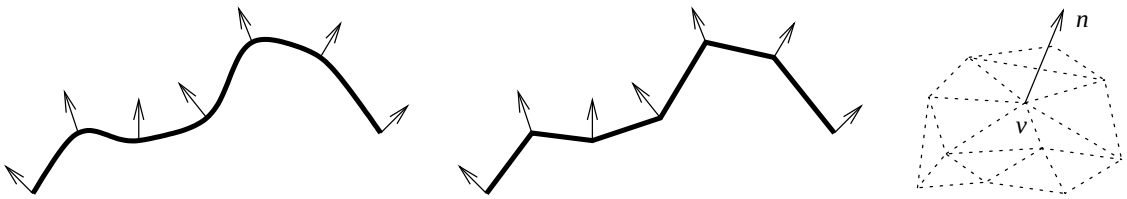


Abbildung 2.4: Vertexnormalen: Links, ein Schnitt durch eine Oberfläche; mitte, ein Schnitt durch das Polygonnetz, das die Oberfläche darstellt; rechts, Vertexnormale im Polygonnetz.

einen funktionellen Zusammenhang bestimmt werden, wird i.a. eine Näherung für die Vertexnormale benutzt. Die Normale für den Vertex v ergibt sich dann durch Mittelung

aus den Orientierungen der an den Vertex anliegenden Polygone. Eine einfache Methode zur Berechnung ist die Addition der Normalen der angrenzenden Polygone, und die anschließende Normalisierung des entstandenen Vektors [WND97]. Befindet sich der Vertex an einer Kante, bei der die Normalen der anliegenden Polygone einen relativ großen Winkel bilden, reicht eine Normale am Vertex für die Darstellung nicht aus (Abbildung 2.5). Statt dessen sollten für einen Vertex in verschiedenen Polygonen verschiedene Nor-

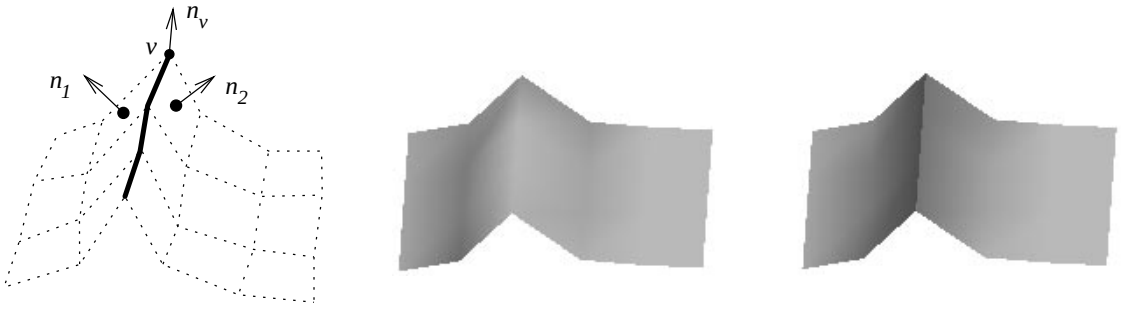


Abbildung 2.5: Die Normalen von nebeneinanderliegenden Polygonen n_1 und n_2 bilden einen relativ großen Winkel. Verwendet man am Vertex v nur die gemittelte Normale n_v, wird die Darstellung zu stark verfälscht (Mitte). Bei Verwendung verschiedener Normalen am Vertex v je Polygon bleibt die Kante (starke Linie) in der Darstellung gut erhalten (rechts).

malen verwendet werden. In der vorliegenden Arbeit wird mit einer Normale je Vertex gearbeitet, da die vorwiegend benutzten Flächen keine scharfen Kanten enthalten. Der Normalenvektor wird mit im Datenteil des Vertexes gespeichert. Vertexnormalen werden bei der Darstellung einer Oberfläche für die Beleuchtungsberechnung benötigt.

2.2.1.3 Polygon

Definition 2.2.2 (Polygon) *Ein Polygon*[4] *dient zur Darstellung einer vollständig durch genau einen geschlossenen, sich nicht selbst schneidenden Streckenzug begrenzten Fläche im dreidimensionalen Raum. Die in dieser Arbeit verwendeten Polygone sind planar.*

Die Darstellung im Computer kann als Liste von Indexverweisen auf Vertizes erfolgen. Die begrenzenden Strecken des Polygons ergeben sich aus aufeinanderfolgenden Verti-

[4]Es werden nicht alle Arten von Polygonen betrachtet, insbesondere keine Polygone mit Löchern oder sich selbst schneidende Polygone.

zes. Die letzte Strecke wird vom letzten zum ersten Vertex gezogen. Die Orientierung eines Polygons ergibt sich aus der Reihenfolge der Vertizes in der Liste. Bei konvexen Polygonen genügen zur Bestimmung der Orientierung drei aufeinanderfolgende nicht kolineare Vertizes. Der Begriff "konvexes Polygon" bedarf einer näheren Erläuterung, da sich bei weitergehender Untersuchung Unklarheiten ergeben können. Ein Polygon wird als konvex bezeichnet, wenn alle Punkte der Verbindungsstrecke zwischen zwei beliebigen Punkten des Polygons innerhalb des Polygons liegen [Hec94]. Damit läßt sich eine formale Definition der Konvexität einer Punktmenge aufstellen. Für eine Punktmenge S gilt:

$$S \text{ ist konvex} \Leftrightarrow (p \in S) \wedge (q \in S) \Rightarrow \forall \lambda : 0 \leq \lambda \leq 1 : p + \lambda(q - p) \in S$$

Bei konkaven Polygonen ist für die Bestimmung einer eindeutigen Orientierung die Auswertung aller Vertizes notwendig. Vorwiegend werden Polygone mit dem Grad 3 (Dreiecke) benutzt. Da in den verwendeten Graphen vorwiegend Polygone eines einzigen Grades benutzt werden, wird ein Polygon mit Hilfe eines Feldes fester Länge mit Indexverweisen auf Vertizes beschrieben.

2.2.1.4 Polygonnetz

Die Darstellung eines Polygonnetzes im Rechner soll für die Anwendung unterschiedlichster Algorithmen geeignet sein. Ein Polygonnetz besteht aus Vertizes, Kanten und Polygonen. Bei Algorithmen treten häufig folgende Fragestellungen auf:

- Finden aller Kanten eines Polygons,
- finden aller Polygone mit einem bestimmten Vertex,
- finden der zu einer Kante gehörenden Polygone,
- finden der benachbarten Vertizes eines bestimmten Vertexes,
- finden aller von einem Vertex ausgehenden Kanten.

Bei der Darstellung eines Polygonnetzes im Rechner muß ein Kompromiß zwischen Speicherplatzbedarf, Aufwand zur Bestimmung der oben genannten Fragen und Wahrung der

Konsistenz der Polygonnetzdarstellung gefunden werden. Für die Darstellung eines Polygonnetzes wurde eine Repräsentation gewählt, welche sich an im Max-Planck-Institut für neuropsychologische Forschung schon vorhandene Darstellungen anpaßt. Ein Polygonnetz wird repräsentiert durch:

- Eine Tabelle von Vertizes und
- eine Tabelle von Polygonen, die mit Hilfe von Indexverweisen auf die Vertizes dargestellt sind.

Jeder Vertex hat eine Liste seiner Nachbarn und eine Liste von Polygonen, in denen der Vertex referenziert wird.

Eine andere Möglichkeit, Polygonnetze zu repräsentieren, besteht darin, die Vertizes nicht als Indexverweise, sondern direkt abzuspeichern. Das erlaubt einen schnelleren Zugriff auf die Vertizes. Da bei Polygonnetzen Vertizes von verschiedenen Polygonen oft gemeinsam benutzt werden, werden viele Vertizes mehrfach gespeichert, daher ist eine Änderung eines Vertexes aufwendiger.

Werden die Kanten eines Polygonnetzes in einer Tabelle abgelegt, läßt sich ein Polygonnetz mit drei Tabellen, eine für die Vertizes, eine für die Kanten, mit Indexverweisen auf die Vertizes und Polygone und einer Tabelle für die Polygone mit Indexverweisen auf die Kanten darstellen. Bei dieser Repräsentation ist das Auffinden der zu einer Kante gehörenden Polygone besonders einfach.

2.2.1.5 Ebene

Definition 2.2.3 (Ebene im 3 dimensionalen Raum) *Eine Ebene E in $\mathbb{R}^3$ bildet einen zweidimensionalen Unterraum $\mathbb{H}$. Dargestellt wird sie durch einen Punkt p der in $\mathbb{H}$ liegt und einen Einheitsvektor $\vec{u}$, der die Orientierung der Ebene angibt.*

Die übliche Darstellung einer Ebene als $ax + by + cz + d = 0$ läßt sich aus $\vec{u}$ und p bestimmen. a, b, c sind die Komponenten von $\vec{u}$. d läßt sich durch Einsetzen von p und

$\vec{u}$ bestimmen. Die Normierung von $\vec{u}$ als Einheitsvektor vereinfacht viele Berechnungen, wie z.B. die Division durch die Länge des Vektors.

2.2.2 Affine Transformationen im dreidimensionalen Raum

2.2.2.1 Translation

Um einen Punkt p um einen Vektor $\vec{v}$ zu verschieben, wird der Vektor $\vec{v}$ Komponentenweise zu p addiert.

$$p' = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \end{pmatrix} = p + \vec{v} \tag{2.1}$$

2.2.2.2 Skalierung

Die Skalierung eines Vektors $\vec{p}$ entlang der verschiedenen Achsen erfolgt durch komponentenweise Multiplikation. Wenn der Vektor $\vec{s}$ die Faktoren für die Skalierung in $x-$, $y-$ und $z-$Richtung enthält ergibt sich folgender Zusammenhang:

$$\vec{p}\,' = \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} * \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = \begin{pmatrix} p_x s_x \\ p_y s_y \\ p_z s_z \end{pmatrix} = \vec{p}\ \vec{s} \tag{2.2}$$

Die Skalierung erfolgt dabei relativ zum Koordinatenursprung. Ist $s_x = s_y = s_z$ handelt es sich um eine gleichmäßige Skalierung in alle Raumrichtungen.

2.2.2.3 Rotation

Die Rotation in 3D kann aus der Rotation in 2D hergeleitet werden, indem die Rotation in $\mathbb{R}^3$ aus Rotationen um die Hauptachsen (x, y, z) zusammengesetzt wird. Die Rotationsmatrizen für die drei Hauptachsen lauten.
Für die X-Achse:

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{pmatrix} \tag{2.3}$$

Für die Y-Achse:

$$R_y(\phi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \tag{2.4}$$

Für die Z-Achse:

$$R_z(\phi) = \begin{pmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

Die Gesamtrotation $\tilde{R}$ erhält man durch Multiplikation der Rotationsmatrizen $\tilde{R}(\phi) = R_x(\phi_x)R_y(\phi_y)R_z(\phi_z)$. Die Reihenfolge[5] der Rotationen beeinflußt das Ergebnis und darf daher nicht beliebig sein. Die Zerlegung einer Rotation in die Winkel $\tilde{R}(\phi)$ ist nicht eindeutig. Das heißt für eine Rotation können mehrere Zerlegungen existieren. Die Rotation $\tilde{R}$ erfolgt relativ zum Koordinatenursprung.

In einer interaktiven Programmumgebung ist die alleinige Verwendung von Hauptachsenrotationen benutzerunfreundlich. Besser wäre die Rotation um eine beliebige Achse $\vec{u} = (u_x, u_y, u_z)^T$.
Die hierfür benötigte Matrix lautet:

[5]Eine Betrachtung unterschiedlicher Ausprägungen, wie "Eulersche Winkel" und "yaw-pitch-roll" (siehe dazu [Sie96, S.47]) erfolgt hier nicht.

$$R_{\vec{u}}(\phi) = \tag{2.6}$$

$$\begin{pmatrix} u_x^2 + cos(\phi)(1-u_x^2) & u_x u_y(1-cos(\phi)) - u_z sin(\phi) & u_x u_z(1-cos(\phi)) + u_y sin(\phi) \\ u_x u_y(1-cos(\phi)) + u_z sin(\phi) & u_y^2 + cos(\phi)(1-u_y^2) & u_y u_z(1-cos(\phi)) - u_x sin(\phi) \\ u_x u_z(1-cos(\phi)) - u_y sin(\phi) & u_y u_z(1-cos(\phi)) + u_x sin(\phi) & u_z^2 + cos(\phi)(1-u_z^2) \end{pmatrix}$$

Sie kann hergeleitet werden, indem man den Vektor $\vec{u}$ mit Hilfe der Rotationen R_x, R_y und R_z in eine Hauptachse rotiert. In dieser Hauptachse dreht man dann mit ϕ und kehrt die vorigen Rotationen um.

2.2.3 Quaternionen und Rotationen

Neben der Verwendung von Matrizen und Hauptachsenrotationswinkeln können Rotationen auch durch Quaternionen beschrieben werden. Bei dem Rotationsschema "Arcball Rotation Control", welches im Kapitel 3.2.3.1 beschrieben wird, werden Quaternionen für die Darstellung der Rotationen benutzt. Eine Quaternion ist gegeben durch ein 4-Tupel $q = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$, wobei w, x, y und z reelle Zahlen und $\boldsymbol{i}, \boldsymbol{j}$ und $\boldsymbol{k}$ Basiselemente sind. Für Quaternionen lassen sich die Operationen Addition und Multiplikation definieren. Die Addition von Quaternionen erfolgt komponentenweise:

$$q + q' = q_w + q'_w + (q_x + q'_x)\boldsymbol{i} + (q_y + q'_y)\boldsymbol{j} + (q_z + q'_z)\boldsymbol{k}$$

Die Multiplikation von Quaternionen wird mit Hilfe des Distributivgesetzes durchgeführt. Für die Multiplikation von $\boldsymbol{i}$, $\boldsymbol{j}$ und $\boldsymbol{k}$ gelten spezielle Gesetze:

$$\begin{aligned} \boldsymbol{i}^2 &= \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1 \\ \boldsymbol{i}\boldsymbol{j} &= \boldsymbol{k} = -\boldsymbol{j}\,\boldsymbol{i} \\ \boldsymbol{j}\boldsymbol{k} &= \boldsymbol{i} = -\boldsymbol{k}\boldsymbol{j} \\ \boldsymbol{k}\,\boldsymbol{i} &= \boldsymbol{j} = -\boldsymbol{i}\boldsymbol{k} \end{aligned}$$

Damit ergibt sich:

$$\begin{aligned} qq' = {} & (q_w q'_w - q_x q'_x - q_y q'_y - q_z q'_z) + (q_w q'_x + q_x q'_w + q_y q'_z - q_z q'_y)\boldsymbol{i} + \\ & (q_w q'_y - q_x q'_z + q_y q'_w + q_z q'_x)\boldsymbol{j} + (q_w q'_z + q_x q'_y - q_y q'_x + q_z q'_w)\boldsymbol{k} \end{aligned}$$

Die Multiplikation von Quaternionen ist nicht kommutativ, d.h. qq' ist im allgemeinen nicht gleich $q'q$. Die Norm einer Quaternion berechnet sich durch $N(q) = q_w^2 + q_x^2 + q_y^2 + q_z^2$. Hat die Norm einer Quaternion q den Wert 1, ist q eine Einheitsquaternion. Eine konjugierte Quaternion q^* ergibt sich aus q durch:

$$q^* = (w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k})^* = w - x\boldsymbol{i} - y\boldsymbol{j} - z\boldsymbol{k}$$

Konjugierte Quaternionen erfüllen die Eigenschaften $(q^*)^* = q$ und $(pq)^* = q^*p^*$.

Mit Einheitsquaternionen können Rotationen repräsentiert werden. Eine Einheitsquaternion $q = (w, x, y, z) = (\cos\theta, \vec{u}\ sin\theta)$ repräsentiert eine Rotation um den Einheitsvektor $\vec{u}$ mit dem Winkel 2θ. Folgt auf eine Rotation q eine Rotation p, wird die daraus resultierende Gesamtrotation durch das Quaternionenprodukt pq repräsentiert. Ein Vektor $\vec{v}$ aus $\mathbb{R}^3$ wird mit der Quaternion q durch die Multiplikation $R = q\vec{v}q^*$ rotiert. Die Verwendung von Quaternionen ist vorteilhaft, wenn Rotationen nacheinander ausgeführt werden, oder wenn zwischen Rotationen interpoliert werden soll.

Die von einer Einheitsquaternion $q = (w, x, y, z)$ beschriebene Rotationsmatrix lautet:

$$R(q) = \begin{pmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{pmatrix}$$

Die Rotation eines Vektors aus $\mathbb{R}^3$ mit Hilfe von Quaternionen erfordert einen größeren Rechenaufwand als die Verwendung der Matrixmultiplikation. Sollen viele Vektoren rotiert werden, ist daher die Umwandlung der Quaternion in eine Rotationsmatrix und die Durchführung der Rotation mit Hilfe dieser Rotationsmatrix günstiger.

2.2.4 Komposition von Transformationen

Für die Komposition der Transformationen Translation, Skalierung und Rotation wird eine einheitliche Darstellung benötigt. Während Skalierung und Rotation durch 3×3 Matrizen beschrieben werden können, ist die Translation als Addition von Punkt und

Vektor dargestellt. Wenn die Punkte in homogenen Koordinaten[6] dargestellt werden, können alle drei Transformationen als Matrizen dargestellt werden. Bei homogenen Koordinaten wird zu den Punkten eine 4. Koordinate hinzugefügt. Aus $p = (p_x, p_y, p_z)^T$ wird $p_h = (p_x, p_y, p_z, W)^T$. W erhält bei Punkten den Wert 1 und bei Vektoren den Wert 0. Eine Translation um den Vektor $\vec{v}$ wird dann durch

$$T(\vec{v}) = \begin{pmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

eine Skalierung durch

$$S(\vec{s}) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

und eine Rotation durch

$$R(\phi, \vec{u}) = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ R_{21} & R_{22} & R_{23} & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

beschrieben. Die Komposition der Transformationen erfolgt dabei durch Matrixmultiplikationen, wobei die erste Transformation rechts steht und die folgenden Transformationen an die linke Seite multipliziert werden.

2.2.4.1 Skalierung eines Punktes bezüglich eines beliebigen Punktes

Möchte man einen Punkt p relativ zu einem Bezugspunkt b skalieren, so sind dazu folgende drei Transformationen notwendig:

1. Verschieben des Koordinatenursprungs so, daß b der neue Koordinatenursprung ist. Dabei wird der Punkt p um den Vektor $-\vec{b}$ mit Hilfe von $T = T(-\vec{b})$ verschoben.

[6] homogene Koordinaten siehe [FDFH96] Kap 5.2

2. Durchführen der Skalierung $S = S(\vec{s})$ mit den Skalierungsfaktoren aus $\vec{s}$.

3. Umkehr der Verschiebung durch Verschiebung um den Vektor $\vec{b}$ durch T^{-1}.

Die gesamte Transformation T_g lautet dann $T_g \ = \ T^{-1}ST$.

Bemerkung

Für die Operationen des Verschiebens und Skalierens werden in der Implementierung die Formeln 2.1 und 2.2 benutzt, da eine Berechnung unter Verwendung von Matrizen aufwendiger wäre.

2.2.4.2 Rotation eines Punktes um ein beliebiges Zentrum

Die Rotation eines Punktes p um ein Zentrum b mit der Rotationsachse $\vec{u}$ und dem Drehwinkel ϕ erfolgt nach dem gleichen Prinzip, wie die Skalierung um einen beliebigen Punkt.

1. Verschieben des Koordinatenursprungs so, daß b der neue Koordinatenursprung ist. Dabei wird der Punkt p um den Vektor $-\vec{b}$ mit Hilfe von $T = T(-\vec{b})$ verschoben.

2. Durchführen der Rotation $R = R(\vec{u}, \phi)$.

3. Umkehr der Verschiebung durch $T^{-1} = T(\vec{b})$.

Die gesamte Transformation T_g lautet dann

$$T_g \ = \ T^{-1}RT = \begin{pmatrix} R_{11} & R_{12} & R_{13} & b_x - R_{11}\ b_x - R_{12}\ b_y - R_{13}\ b_z \\ R_{21} & R_{22} & R_{23} & -R_{21}\ b_x + b_y - R_{22}\ b_y - R_{23}\ b_z \\ R_{31} & R_{32} & R_{33} & -R_{31}\ b_x - R_{32}\ b_y + b_z - R_{33}\ b_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

Mit dem Ausdruck 2.7 ist eine übersichtliche Darstellung dieser Transformation gegeben. Da die Operation in der Implementierung nicht mit anderen Transformationen verknüpft wird, kann die Anzahl der Berechnungsschritte für diese Operation gegenüber der Formel 2.7 gesenkt werden. Die Transformation eines Punktes in homogenen Koordinaten mit Hilfe der Matrixmultiplikation mit einer 4×4-Matrix erfordert 16 Multiplikationen und 12

Additionen[7]. Verwendet man statt der homogenen Koordinaten dreidimensionale Koordinaten, werden die beiden Translationen der Operation als Vektoradditionen ausgeführt und die Rotation erfolgt mit Hilfe einer 3×3-Matrix. Dafür sind 9 Multiplikationen und 12 Additionen erforderlich. Da die unterste Zeile der 4×4-Transformationsmatrix in 2.7 nur mit Nullen und in der vierten Spalte mit 1 belegt ist, und die vierte Komponente des zu transformierenden Punktes bei der vorliegenden Problemstellung immer 1 ist, kann die Matrixmultiplikation auch verkürzt mit 9 Multiplikationen und 9 Additionen, quasi durch weglassen der vierten Zeile der Matrix, ausgeführt werden.

2.2.5 Spiegelung eines Vertexes an einer Ebene

Unter der Spiegelung eines Vertexes p an einer Ebene E versteht man die Verschiebung des Vertexes in Richtung der Senkrechten von p zu E um den doppelten Betrag der Länge der Strecke entlang der Senkrechten von p zu E (Abbildung 2.6). Hat der Vertex eine Normale $\vec{n}$, so wird diese ebenfalls gespiegelt (Abbildung 2.7). Für die Spiegelung der Normalen kann eine einfachere Berechnung vorgenommen werden. Wird ein Polygon gespiegelt, so muß auch die Orientierung dieses Polygons geändert werden, um die gleichen Vorder- und Rückseiten von Flächen beizubehalten.

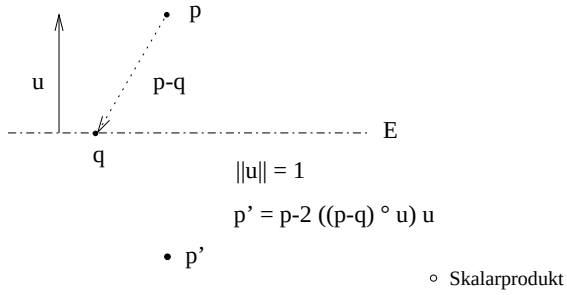


Abbildung 2.6: Spiegelung eines Vertexes p an einer Ebene E mit Orientierung $\vec{u}$ und dem Punkt q in der Ebene.

[7] In der vorliegenden Problemstellung ist die vierte Komponente W eines Punktes immer 1, Divisionen durch W sind daher nicht erforderlich.

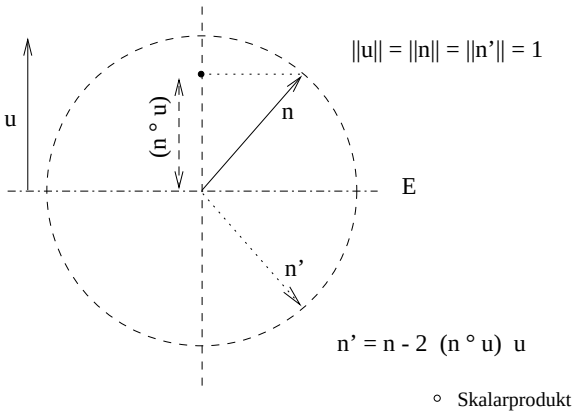


Abbildung 2.7: Spiegelung einer Vertexnormalen n an einer Ebene E mit der Orientierung $\vec{u}$.

2.2.6 Bestimmung des Schnittpunktes einer Kante mit einer Ebene

Gegeben sei eine Ebene E und eine Strecke S mit den Endpunkten s_1, s_2. Mit $\vec{g} = (s_2 - s_1)$ kann man S als $S = s_1 + t\vec{g}$ mit $0 \leq t \leq 1$ darstellen. Für die durch S laufende Gerade wird $t \in \mathbb{R}$ als beliebig angenommen. Für die Lage von S gibt es drei Möglichkeiten:

1. S schneidet E $(0 \leq t \leq 1)$
2. S liegt in E oder ist parallel zu E (t nicht berechenbar)
3. S schneidet E nicht $(t < 0$ oder $1 < t)$

Im Fall, daß s_1 oder s_2 in E liegen und nicht Fall (2) vorliegt, wird E von S nur berührt. In diesem Fall ist $t = 0$ oder $t = 1$. In den Verfahren, in denen der Schnittpunkt einer Kante mit einer Ebene eine Rolle spielt, muß dieser Fall nicht gesondert behandelt werden. Daher wird die Ebene als geschnitten betrachtet (Abbildung 2.8).

2.2.7 Verwendung der Transformationen im Editor

Die affinen Transformationen gehören zu den topologieerhaltenden Operationen im Editor. Für die Operationen *Verschieben*, *Spiegeln*, *Drehen* und *Skalieren* wird vor der eigentlichen Durchführung der Operation eine Auswahl der betreffenden Primitive getroffen. Hierzu stehen mehrere Möglichkeiten zur Verfügung. Eine Einzelauswahl läßt

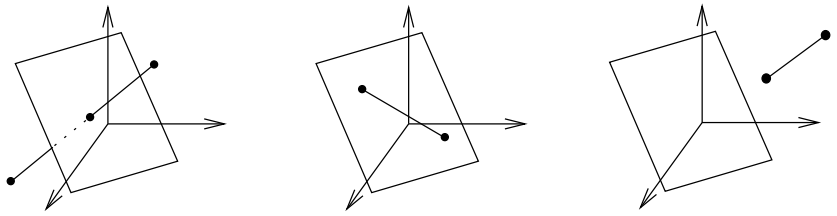


Abbildung 2.8: Verschiedene Lagen von Strecke und Ebene.

zwar alle Möglichkeiten offen, ist jedoch aufgrund der großen Anzahl der Primitive kaum praktikabel. Als Alternativen wurden folgende Selektionsmöglichkeiten implementiert:

- Vertizes einer Zusammenhangskomponente
- Vertizes innerhalb eines rechteckigen Bereiches, der alle Vertizes umfaßt, die innerhalb des Bereiches der vier Ebenen liegen, die durch die Rechteckskanten und einer Senkrechten zur Projektionsebene beschrieben werden (ähnlich einem "Tunnel").
- einzelne Polygone
- Polygone einer Zusammenhangskomponente
- Polygone innerhalb eines vom Benutzer vorgegebenen, abgegrenzten Gebietes auf der Oberfläche

Im Anschluß an die Selektion der Primitive werden die weiteren Parameter einer Operation bestimmt. Für die *Verschiebung* wird ein Verschiebungsvektor parallel zur Projektionsebene aus der Mausbewegung konstruiert (Kapitel 3.2.2). Für die *Spiegelung* wird eine Ebene benötigt, an der gespiegelt werden soll. Der Nutzer hat die Wahl zwischen einer frei positionierbaren Ebene oder einer Hauptebene des Koordinatensystems. Für die *Drehung* ist ein Rotationszentrum, eine Rotationsachse und der Drehwinkel erforderlich. Das Rotationszentrum wird zu Beginn der Rotation mit Hilfe der selektierten Primitive so bestimmt, daß es im Inneren der selektierten Primitive liegt. Die Rotationsachse ist hierbei die Achse, die senkrecht zur Projektionsebene in den Bildschirm hineinzeigt.

Der Drehwinkel wird aus einer vertikalen Mausbewegung gewonnen. Für die *Skalierung* werden ein Skalierungszentrum und ein Skalierungsfaktor benötigt. Das Zentrum wird in der gleichen Weise wie bei der Rotation bestimmt. Der Faktor ergibt sich aus der Mausbewegung in vertikaler Richtung. Für die eben beschriebenen Funktionen lassen sich Einschränkungen bestimmter Parameter auf Hauptachsen vornehmen, um dem Nutzer relativ genaue Manipulationen zu ermöglichen. Im einzelnen bedeutet das:

- Die *Verschiebung* erfolgt entlang einer einzigen Achse (x,y,z) oder parallel zur xy-, yz- oder xz-Ebene.
- Die *Spiegelung* erfolgt an der xy-, yz- oder xz-Ebene oder an der um 45° gedrehten xy-, yz-, xz-Ebene. Die Ebene geht dabei durch die Mitte der Szene.
- Die Rotationsachse bei der *Drehung* ist die x-, y- oder z-Achse.
- Die *Skalierung* erfolgt nur in Richtung einer oder zweier Achsen.

Während bei der *Verschiebung*, *Drehung* und *Skalierung* ein Parameter fortlaufend verändert werden kann (Verschiebungsvektor, Drehwinkel, Skalierungsfaktor), wird die *Spiegelung* mit einer einzelnen Aktion abgeschlossen. Das hat Auswirkungen auf die Undo-Funktion, die in Kapitel 3.3.4 beschrieben wird.

2.3 Verfahren zur Bearbeitung von Polygonnetzen

2.3.1 Trennung von Polygonnetzen

Das Herauslösen von Ausschnitten aus Polygonnetzen ist eine wichtige Editorfunktion für die Durchführung von Vermessungen an der Hirnoberfläche.
Diese Oberflächen sind sehr stark gefaltet, interessante Ausschnitte sind oft verdeckt. Das entwickelte Trennverfahren erlaubt die zielgerichtete Entfernung verdeckender Netzbestandteile. Das Prinzip der Trennung basiert auf der Erzeugung eines Randes entlang der vorgegebenen Trennstellen. Die Vertizes an den Trennstellen werden dupliziert und den Polygonen der verschiedenen Seiten zugeordnet. Da kein allgemein akzeptiertes Verfahren zur Trennung von Polygonnetzen existiert, wurden im Rahmen der Diplomarbeit die nachfolgend beschriebenen Verfahren entwickelt.

2.3.1.1 Trennen eines Polygonnetzes mit Hilfe einer Trennlinie

Eine Trennlinie besteht aus zusammenhängenden Kanten des Polygonnetzes. Um eine sinnvolle Trennoperation durchführen zu können, muß eine Trennlinie folgende Eigenschaften erfüllen:

- Sie darf sich nicht selbst schneiden und
- darf keine Kante vom Rand enthalten.
- Nur die erste oder letzte Kante der Trennlinie darf einen Randpunkt enthalten.

Bei der Eingabe der Trennlinie durch den Nutzer werden diese Kriterien automatisch geprüft.

Für die Trennungen im Polygonnetz sind zunächst vier grundlegende Fälle denkbar (Abbildung 2.9). Sie unterscheiden sich in der Verfahrensweise, die bei der Verteilung der neuen Vertizes auf die Polygone und der Behandlung der Endpunkte der Trennlinie durchgeführt wird.

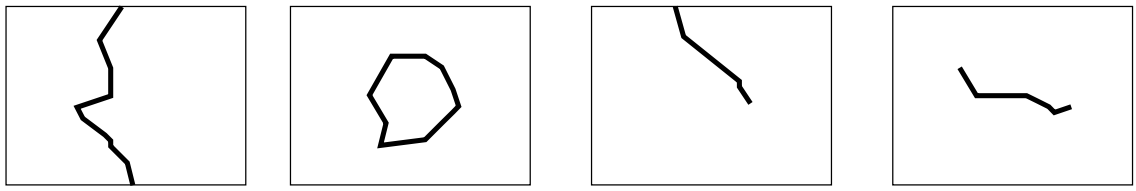


Abbildung 2.9: Vier grundlegende Fälle der Auswahl einer Trennlinie, weitere Erklärungen im Text.

(a) Anfangs- und Endpunkt der Trennlinie liegen am Rand. Alle Vertizes der Trennlinie werden dupliziert.

(b) Die Trennlinie bildet einen geschlossenen Kantenzug. Alle Vertizes der Trennlinie werden dupliziert. Der Fall einer nichtorientierbaren Fläche ist gesondert zu behandeln.

(c) Nur ein Endpunkt der Trennlinie liegt am Rand. Alle Vertizes der Trennlinie werden dupliziert außer der Endpunkt der Trennlinie, der nicht am Rand der Fläche liegt.

(d) Die Trennlinie verläuft inmitten der Fläche. Alle Vertizes der Trennlinie außer den Endpunkten werden dupliziert.

Für die Zuordnung der duplizierten Vertizes der Trennlinie zu Polygonen einer bestimmten Seite wird eine "linke" und eine "rechte" Seite bezüglich der vorgegebenen Trennlinie bestimmt. Für Netze in 2D eignet sich folgendes Verfahren: Um die Zugehörigkeit von Polygonen, die einen bestimmten Vertex p der Trennlinie gemeinsam haben, zu einer Seite zu bestimmen, wird mit Hilfe der zwei an p liegenden Kanten der Trennlinie zunächst festgestellt, ob die von beiden Kanten beschriebene Ecke konvex oder konkav ist (Abbildung 2.10). Die Ecke am Punkt p_i ist konvex, falls $\vec{k} \circ \vec{p}_i > 0$ ist (mit $\vec{k} = (p_i - p_{i-1}) \times (p_i - p_{i+1})$ und dem Skalarprodukt $\circ$, Kreuzprodukt $\times$). Falls die Ecke konvex ist, gehören alle Nachbarn des Punktes p auf die rechte Seite, die sowohl rechts von Kante k_1 und rechts von Kante k_2 sind. Falls die Ecke konkav ist, gehören alle Nachbarn auf die rechte Seite, die rechts von Kante k_1 oder rechts von Kante k_2 sind. Für 3D ist dieses Verfahren jedoch nicht ausreichend, da die verwendeten Kriterien nicht für alle Fälle gelten.

Die Bestimmung von "linker" und "rechter" Seite kann auch mit Hilfe der Netztopologie erfolgen (Abbildung 2.11). Die Nutzung der Netztopologie ist rechnerisch weniger aufwendig als geometrische Methoden. Von einem Startpolygon (p_{s_1} oder p_{s_2}) wird von der Startkante aus ein Polygon der nächsten Kante (p_{f_1} oder p_{f_2}) gesucht. Die Suche erfolgt unter den Polygonen, die den gemeinsamen Vertex v_i haben. v_i ist dabei der zweite Vertex der aktuell untersuchten Trennkante. Die Suche wird fortgesetzt, bis die letzte Kante erreicht ist. Bei geschlossenen Trennlinien werden für die Suche an der letzten Trennkante die Polygone an der Startkante benutzt. Nach dieser Prozedur sind die Polygone "links" der Trennlinie bekannt. Jetzt können die Vertizes der Trennlinie verdoppelt und die Vertizes der mit der Trennlinie verbundenen Polygone auf der "linken" Seite der Trennlinie gegen die neuen Vertizes ausgetauscht werden. Damit ist eine topologische Trennung des Polygonnetzes entlang der Trennlinie erfolgt. Das Verfahren funktioniert für alle Po-

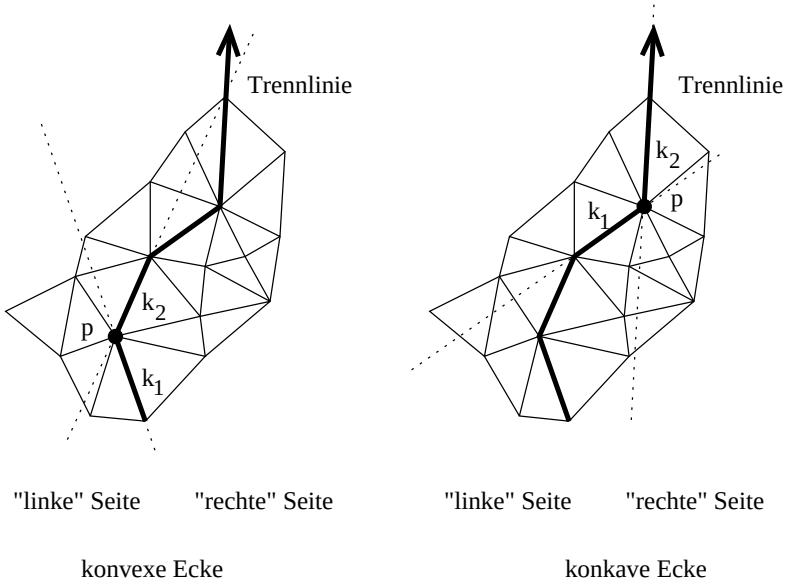


Abbildung 2.10: Bestimmung von rechter und linker Seite mit Hilfe der Geometrie.

lygonnetze mit konvexen Polygonen vom Grad $g \geq 3$, falls die Kanten, aus denen die Trennlinie besteht in jeweils genau zwei Polygonen vorkommmen. Für nichtorientierbare Flächen (siehe Definition 2.1.4), wie Möbiusbändern[8], ist beim Austausch der Vertizes eine Sonderfallbehandlung nötig (Abbildung 2.12).

2.3.1.2 Zerteilung eines Polygonnetzes mit Hilfe einer Ebene

Ziel ist es, ein bestehendes Polygonnetz mit einem "glatten" Schnitt entlang einer Ebene zu teilen (Abbildung 2.13). Nach Durchführung dieser Operation besteht das Polygonnetz aus zwei oder mehr einzeln zusammenhängenden Teilen. Um einen "glatten" Schnitt zu erreichen, ist es im allgemeinen notwendig, neue Kanten in das Polygonnetz einzufügen, was die Zerlegung von Polygonen nach sich zieht. Die Zerlegung muß so erfolgen, daß eine in der Ebene liegende Trennlinie gefunden werden kann (siehe Abbildung 2.14 und 2.17). Die Aufgabe kann in zwei Teile zerlegt werden:

[8]Beim Möbiusband erfolgt zwar eine Auflösung der Nachbarschaft von gegenüberliegenden Polygonen an der Trennlinie, es erfolgt aber keine topologische Trennung.

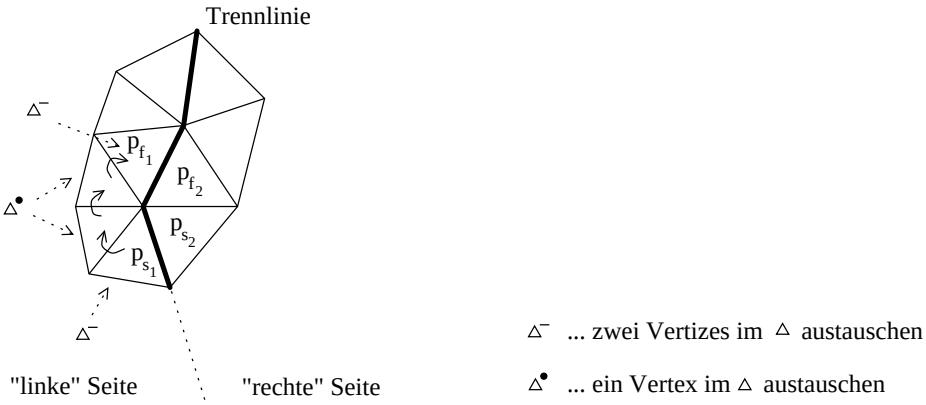


Abbildung 2.11: Bestimmung von rechter und linker Seite mit Hilfe der Topologie.

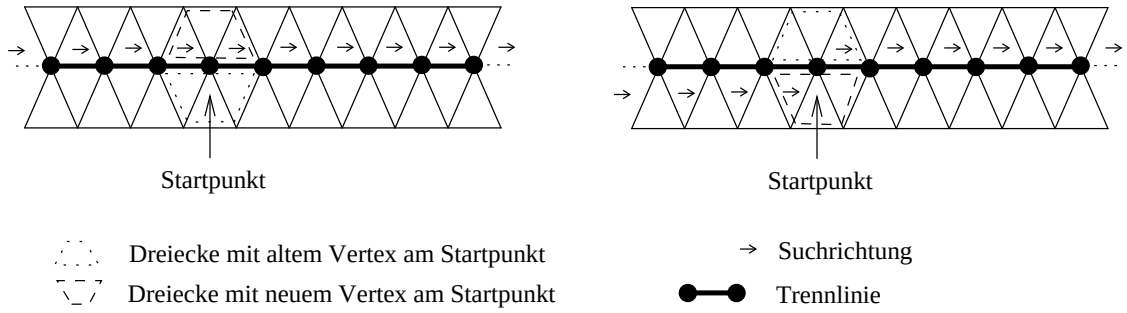


Abbildung 2.12: Vergleich einer Trennung eines geschlossenen Bandes (links) und eines Möbiusbandes (rechts). Dreiecke auf der Seite mit den Pfeilen erhalten die neuen Vertizes. Bei dem Möbiusband endet die Suche nicht am Startdreieck, sondern bei dessen Nachbarn. Daher ist der neue Vertex am Startpunkt anders zuzuordnen.

1. Erzeugen neuer Kanten im Polygonnetz, so daß sich eine möglichst gut an die Schnittebene angepaßte Trennlinie ergibt. Dazu ist im allgemeinen eine Zerlegung der Polygone entlang der Schnittebene notwendig. Das Bestimmen der Kanten, die entlang der Trennebene liegen und eine Trennlinie ergeben, erfolgt im gleichen Schritt.

2. Trennen des Polygonnetzes mit Hilfe der gefundenen Trennlinie, wobei sich die Trennlinie aus mehreren Teilstücken zusammensetzen kann.

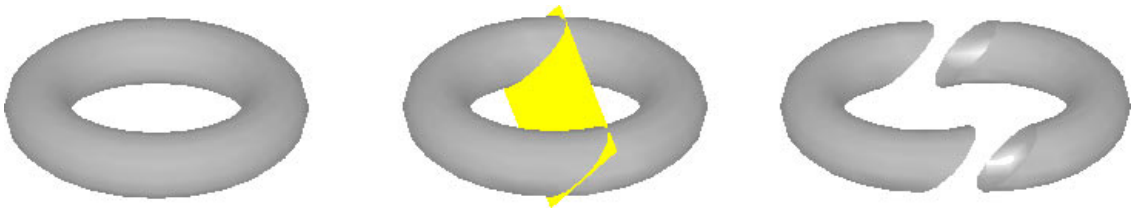


Abbildung 2.13: Beispiel für die Trennfunktion.

2.3.1.3 Zerlegung der Dreiecke, die von der Ebene geschnitten werden

Zunächst werden die entlang der Ebene liegenden Polygone (hier Dreiecke) entsprechend ihrer Lage zur Ebene so geteilt, daß Kanten entstehen, die zur Trennlinie gehören. Alle Kanten im Polygonnetz werden daraufhin untersucht, ob sie von der Ebene geschnitten werden. Wird eine Kante geschnitten, wird der Abstand des Schnittpunkts zu den Endpunkten der Kante bestimmt. Ist der Abstand des Schnittpunkts zu einem Endpunkt sehr klein, würden bei der Zerlegung des anliegenden Dreiecks nach Abbildung 2.14a sehr kleine neue Dreiecke entstehen. Liegt ein Endpunkt genau in der Trennebene, ist eine Zerlegung der Kante an dieser Position nicht möglich. Es wird daher eine Schranke e festgelegt, bei deren Unterschreitung keine Zerlegung von Kanten erfolgt. Für die Trennlinie werden dann entweder schon im Polygonnetz vorhandene Kanten (Abbildung 2.14d) oder die bei der Zerlegung nach Abbildung 2.14b entstehende Kante benutzt (siehe auch Abbildung 2.17). Um zusammenhängende Trennlinien zu erhalten, sollte e kleiner als die kürzeste Kante entlang der Trennebene sein. Ist der Abstand des Schnittpunkts zu einem Endpunkt der untersuchten Kante kleiner als e, wird der Endpunkt der Kante, der näher an der Ebene liegt für die spätere Verwendung in einer Menge $\mathcal{P}$ zwischengespeichert (Fall(b), (c) oder (d)). Liegt der Schnittpunkt außerhalb des definierten Mindestabstandes e (Fall(a) oder (b)), wird die Kante mit ihrem Schnittpunkt zur späteren Verwendung in einer vorläufigen Kantenliste $\mathcal{K}_t$ zwischengespeichert. Nach Abschluß dieser Untersuchung enthält $\mathcal{K}_t$ alle Kanten des Ausgangsnetzes, die von der Ebene geschnittenen wurden und $\mathcal{P}$ die Endpunkte der Kanten, zu denen der Abstand des jeweiligen Schnittpunkts kleiner als e ist. Die Kanten der Trennlinie mit Fall (d) werden mit Hilfe von $\mathcal{P}$ bestimmt und in die Liste $\mathcal{K}$ mit den zur Trennlinie gehörenden Kanten eingetragen. Die

Kanten mit Fall (d) stammen aus dem unveränderten Polygonnetz und müssen nicht neu erzeugt werden. Aus $\mathcal{K}_t$ werden nun alle Kanten entfernt, die einen Punkt aus $\mathcal{P}$ enthalten

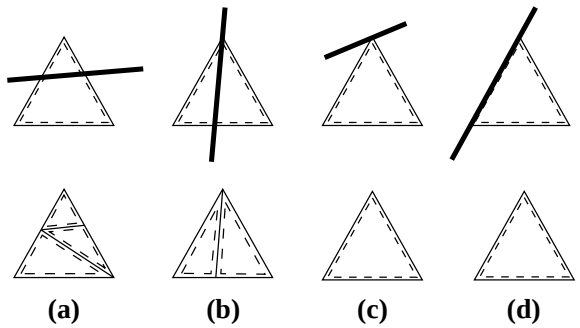


Abbildung 2.14: 4 Möglichkeiten, wie ein Dreieck durch eine Ebene geteilt werden kann. In der oberen Reihe ist die Ebene als starke Linie dargestellt. In der unteren Reihe die Zerlegung des Dreiecks. Im Fall (a) und (b) ist eine Zerlegung des Dreiecks notwendig. Im Fall (d) wird nur die Trennkante bestimmt. Im Fall (c) erfolgt weder eine Zerlegung noch die Bestimmung einer Trennkante

(siehe Sonderfall in Abbildung 2.15). Jetzt erfolgt die Erzeugung von neuen Vertizes im Polygonnetz mit den Koordinaten der Schnittpunkte der Kanten aus $\mathcal{K}_t$. Danach erfolgt die Neutriangulierung der zugeordneten Polygone.

Für die Neutriangulierung der Polygone werden alle Kanten aus $\mathcal{K}_t$ untersucht. Zu einer Kante $k \in \mathcal{K}_t$ werden die anliegenden Polygone l_i ($i \in \{1, 2\}$) gesucht und einzeln bearbeitet. Für die gefundenen Polygone l_i wird geprüft, ob Fall (a) oder (b) vorliegt (Abbildung 2.14). Es liegt Fall (b) vor, falls ein Vertex v des Polygons l_i in der Menge $\mathcal{P}$ enthalten ist $(v \in l_i) \cap (v \in \mathcal{P}) \neq \emptyset$. Falls das nicht der Fall ist, liegt Fall (a) vor. Liegt Fall (b) vor, wird ein neues Polygon P_n erzeugt und der Vertex p_o des Polygons l_i bestimmt, der kein Endpunkt von der geschnittenen Kante k ist. P_n wird der Vertex p_o, der neue Vertex k_n (Schnittpunkt) und der alte Vertex k_b (von der geschnittenen Kante) zugeordnet (Abbildung 2.16b). Im schon vorhandenen Polygon P_a wird k_b durch k_n ersetzt. Die Ersetzung erfolgt dabei so, daß sich die Orientierung des Polygons nicht ändert. In die Liste mit den zur Trennlinie gehörenden Kanten $\mathcal{K}$ wird die Kante (p_o, k_n) eingetragen.

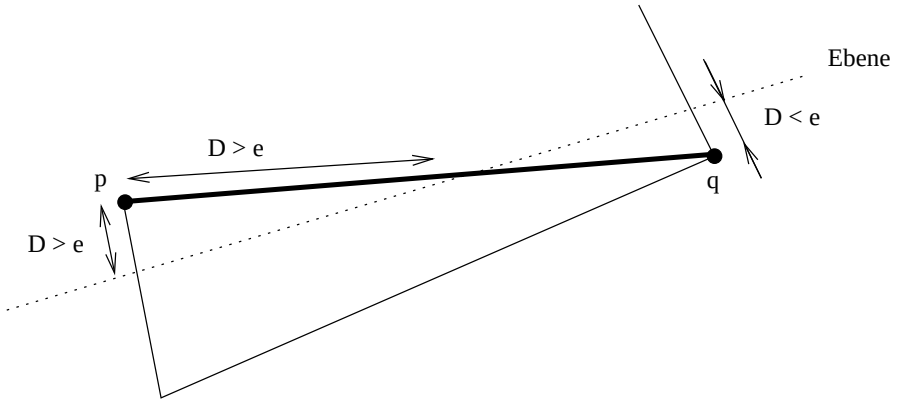


Abbildung 2.15: Die Kante (p,q) wurde als Kante gefunden, die zerlegt werden kann und in die vorläufige Kantenliste $\mathcal{K}_t$ eingefügt. Bei einer anderen von q ausgehenden Kante hat q einen kleineren Abstand D zum Schnittpunkt als die vorgegebene Schranke e. Daher kann die Kante (p,q) nicht zerlegt werden und wird aus $\mathcal{K}_t$ entfernt.

Liegt Fall (a) vor, wird zunächst die zweite von der Ebene geschnittene Kante k_A des Polygons l_i bestimmt (Abbildung 2.16 a_1, a_2). Dann werden zwei neue Polygone P_{n_1} und P_{n_2} erzeugt. Das schon vorhandene Polygon P_a erhält als Vertizes den gemeinsamen Vertex der Kanten k und k_A, sowie deren Schnittpunkte k_n und k_{A_n} (Abbildung 2.16 a_1, a_2). Für die Aufteilung der verbliebenen Fläche des zu zerlegenden Polygons gibt es zwei Möglichkeiten. Es wird die Variante gewählt, bei der die Verbindungslinie der gegenüberliegenden Vertizes am kleinsten ist. Den Polygonen P_{n_1} und P_{n_2} werden daher die Vertizes (k_a, k_n, k_{A_b}) und (k_{A_n}, k_{A_b}, k_n) oder (k_{A_b}, k_a, k_{A_n}) und (k_a, k_n, k_{A_n}) zugeordnet (Abbildung 2.16 a_1, a_2). Die Orientierung der Polygone ist die gleiche wie von P_a. In die Liste mit den zur Trennlinie gehörenden Kanten $\mathcal{K}$ wird die Kante (k_n, k_{A_n}) eingetragen.

Die Liste mit den Kanten $\mathcal{K}$ der Trennlinie ist nun vollständig und das in Kapitel 2.3.1.1 beschriebene Trennverfahren kann angewendet werden. Anschließend werden die Normalen der geänderten Polygone bestimmt. Abbildung 2.17 zeigt ein Beispiel für dieses Verfahren.

Das Trennen von Polygonnetzen gehört zu den topologieverändernden Operationen. Für

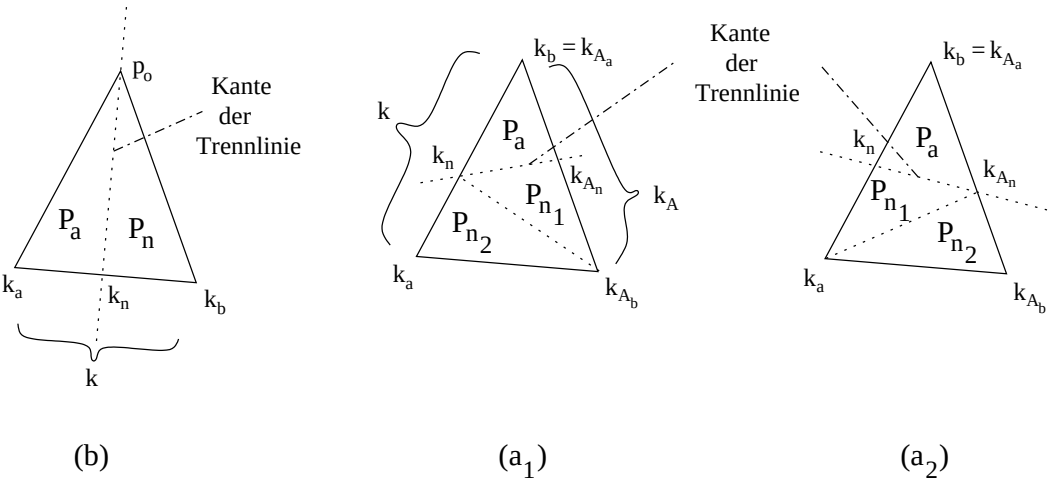


Abbildung 2.16: Erzeugung der neuen Polygone.

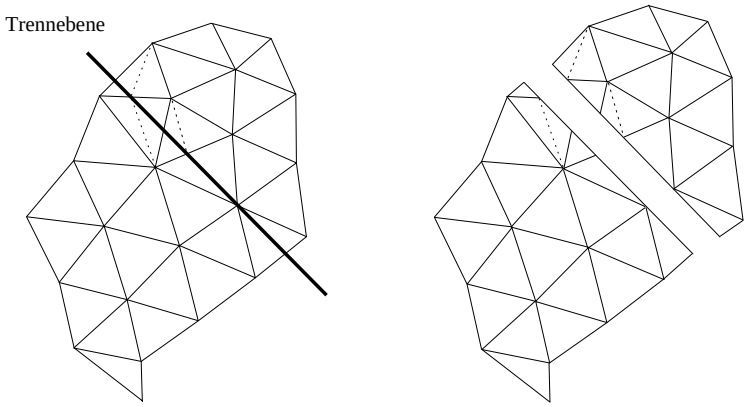


Abbildung 2.17: Ergebnis der Trennung mit einer Ebene

die Trennoperation sind als Parameter eine oder mehrere Listen von zusammenhängenden Kanten, nachfolgend als "Pfade" bezeichnet, erforderlich. Beim Trennen mit vorgegebenen Pfad, wird der Trennpfad durch den Nutzer konstruiert, wie es in Kapitel 3.2.3.3 beschrieben ist. Bei der Trennung an einer Ebene werden die Trennpfade aus den jeweiligen Schnittkanten der Polygone mit der Ebene erzeugt. Die Trennebene kann durch den Nutzer interaktiv verschoben und gedreht werden.

2.3.2 Verformen eines Polygonnetzes

Das Deformieren gehört zu den topologieerhaltenden Operationen. Für das Deformieren von Polygonnetzen sind zahlreiche grundsätzlich verschiedene Methoden denkbar. Eine Gemeinsamkeit aller Methoden ist, daß sie eine Auswahl von Vertizes verschieben. Für jeden Vertex der Auswahl kann dabei ein anderer Verschiebungsvektor benutzt werden. Die Anforderungen an eine Deformationsoperation hängen sehr stark von dem Verwendungszweck ab, so daß eine universelle Operation nicht existiert. Im Editor wird folgende Funktion benutzt: Ein Punkt d wird selektiert. Durch Verschiebung des Punktes werden die Nachbarn des Punktes in einem vorher festgelegtem Abstand r ($r > 0$) mitbewegt. Je größer die Entfernung von einem Nachbar zu d ist, desto weniger wird er mitbewegt (Abbildung 2.18). Wird der zentrale Vertex d um den Vektor $\vec{v}$ verschoben, wird ein in der Nachbarschaft von d liegender Vertex n, der den Abstand ($n_d |\ n_d < r$) von d hat um den Vektor $\vec{v}_n = \frac{r}{r+n_d}\vec{v}$ verschoben. Die Abstandsfunktion ist proportional zu $\frac{1}{n_d}$.

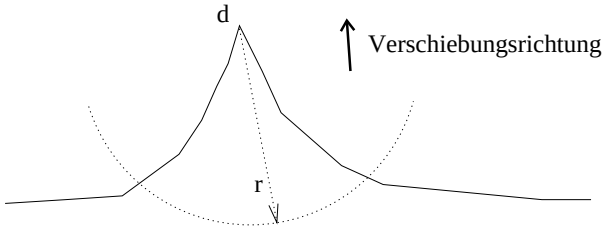


Abbildung 2.18: Deformieren von Polygonnetzen; die Vertizes innerhalb des Abstandes r werden in die Verschiebungsrichtung bewegt.

2.3.3 Netze verbinden

Das Verbinden von Polygonnetzen gehört zu den topologieverändernden Operationen. Für den Editor wurde folgendes Verfahren entwickelt: Polygonnetze können entlang zweier vom Benutzer vorgegebener Pfade verbunden werden. Haben die Pfade die gleiche Länge, werden korrespondierende Kanten einfach vereinigt. Haben die Pfade unterschiedliche Längen, werden zwischen den Pfaden neue Dreiecke erzeugt, die den Zwischenraum von beiden Pfaden ausfüllen. Dabei werden sukzessiv, beginnend an den Startpunkten der Pfa-

de, Dreiecke so erzeugt, daß die nächste, in das Polygonnetz einzufügende Kante möglichst kurz ist (Abbildung 2.19). Durch Auswahl der kürzeren Kante wird eine bessere Triangulierung erreicht. Ist die Richtung der Pfade entgegengesetzt, entstehen überlappende

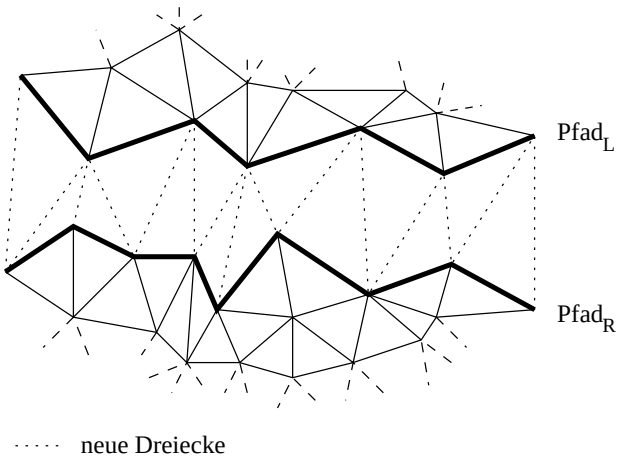


Abbildung 2.19: Verbinden von Polygonnetzen.

Polygone (Abbildung 2.20 a). Befinden sich zwischen den Pfaden andere Netzstrukturen, so werden diese von den neuen Dreiecken durchdrungen (Abbildung 2.20 b). Die Einga-

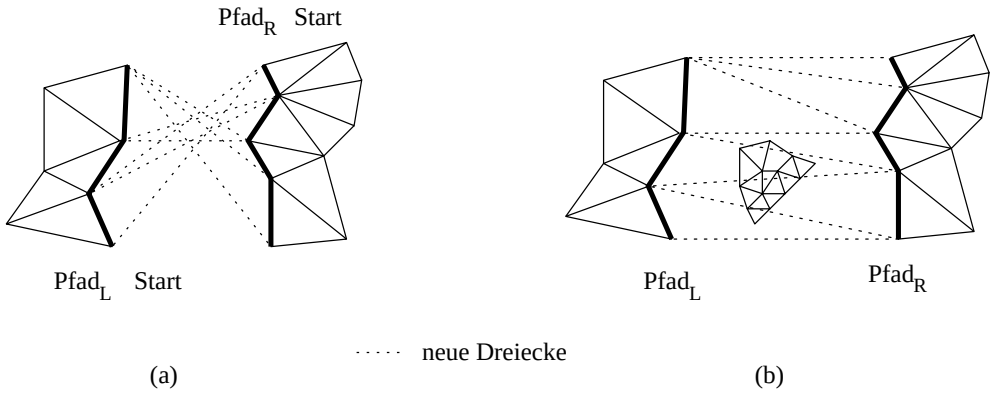


Abbildung 2.20: Ungünstige Fälle beim Verbinden von Polygonnetzen.

be der beiden Verbindungspfade erfolgt nach dem Schema der Pfadeingabe aus Kapitel 3.2.3.3, sie kann jedoch nur entlang der Ränder des Polygonnetzes erfolgen.

2.4 Programmtechnische Grundlagen

2.4.1 Objektorientierung

Im Laufe der letzten Jahre hat sich das Paradigma der objektorientierten Programmierung zur Realisierung großer Softwareprojekte etabliert. Die Wartung und Erweiterung von Programmen ist mit objektorientierten Techniken weniger aufwendig als z.B. bei rein prozeduralen Ansätzen. Die Möglichkeit, verschiedene Programmteile zu modularisieren, um so Programmteile einzeln entwickeln zu können, wird durch spezielle Sprachkonstrukte der objektorientierten Programmiersprachen unterstützt.
Beispiele für objektorientierte Programmiersprachen sind C++ , Smalltalk und Eiffel. Für den im Rahmen der Diplomarbeit entwickelten Editor wurde C++ verwendet. Ein guter Überblick der Möglichkeiten von C++ ist in [Str98] zu finden. Der Editor wurde in das Programm *IPE* integriert, welches mit Hilfe der objektorientierten Programmierung realisiert wurde (siehe auch Kapitel 3.3.5). Nachfolgend werden die Grundbegriffe der objektorientierten Programmierung vorgestellt.

2.4.1.1 Objekte

Objekte repräsentieren logisch zusammengehörige Sachverhalte oder Problemstellungen der verwendeten Modellwelt. Bei der Analyse der zu lösenden Aufgabe wird versucht, das Problem durch Abstraktion in Objekte zu zerlegen. Objekte besitzen Attribute und ein Verhalten. Attribute können wiederum Objekte sein, so daß beliebig komplexe Objekte entwickelt werden können. Das Verhalten der Objekte wird durch Objektmethoden beschrieben. Objekte können untereinander durch den Austausch von Nachrichten kommunizieren.
Als *Klasse* wird die Definition eines Typs von Objekten bezeichnet, in ihr werden die Attribute und Objektmethoden festgelegt. Eine *Instanz* eines Objektes ist ein im Programmablauf benutztes Objekt.

2.4.1.2 Kapselung

Oft ist die Kenntnis über die innere Struktur und Daten eines Objektes nicht zwingend erforderlich. So sind z.B. die Koordinaten der ausgewählten Farbe im Farbauswahlobjekt (Abbildung 3.8) für andere Objekte als das Farbauswahlobjekt unerheblich. Für Benutzer des Farbauswahlobjektes ist nur interessant, die aktuell ausgewählte Farbe zu lesen oder eine spezielle Farbe einzustellen.
Durch Kapselung können Zugriffe auf Daten des Objektes von außen unterbunden werden. Inkonsistente Zustände der Daten im Objekt können damit von außen durch direkten Zugriff nicht entstehen. Die Aussagekraft von Tests der Funktionen des Objektes wird verbessert.
Durch Bereitstellung einer Schnittstelle können andere Funktionen mit dem Objekt kommunizieren. Eine Schnittstelle umfaßt eine Menge von Objektmethoden und in besonderen Fällen zusätzlich eine Menge von Attributen. Durch Kapselung erreicht man die Trennung von Schnittstelle und Implementierung, was vorteilhaft bei der Entwicklung von Software ist. Beim Design der Schnittstellen ist auch die zukünftige Funktionalität der Software zu beachten, da spätere Änderungen an den Schnittstellendefinitionen weitere Änderungen in anderen Programmteilen nach sich ziehen.
Eine strenge Kapselung aller Daten eines Objektes ist nicht immer erwünscht. In C++ kann die Kapselung in mehreren Ebenen erfolgen. Dazu werden in C++ Attribute und Methoden als *private*, *protected* oder *public* deklariert. Durch *public* erfolgt keine Kapselung. Methoden sind von fremden Funktionen aufrufbar, Attribute können von außen geändert werden. Mit *protected* deklarierte Methoden und Attribute können nur vom Objekt selbst und von abgeleiteten Objekten benutzt werden. Durch die Verwendung von *private* kann nur noch das Objekt selbst zugreifen. Der Zugriff von außen auf *protected* oder *private*-Attribute eines Objektes kann nur indirekt mit Hilfe öffentlicher Methoden (*public*) erfolgen.

2.4.1.3 Vererbung

Die Konzepte von Spezialisierung und Generalisierung können durch Vererbung umgesetzt werden. Durch die Vererbung werden Attribute und Methoden einer Basisklasse in eine neue Klasse übernommen. Aus Sicht der neuen Klasse formuliert man, daß die neue Klasse von der Basisklasse *abgeleitet* wurde. Die Funktionalität der neuen Klasse kann erweitert werden und führt dann zu einer Spezialisierung der Basisklasse.
Ererbte Methoden können überschrieben werden, d.h. eine Methode der Basisklasse wird durch Verwendung des gleichen Namens in der abgeleiteten Klasse überschrieben und durch eine andere Implementierung ersetzt. Ein Hilfsmittel bei der Abstraktion ist die Verwendung von *virtuellen Methoden*, die in einer Basisklasse deklariert werden und erst in abgeleiteten Klassen implementiert werden.
Durch Vererbung können Klassen in Vererbungshierarchien organisiert werden. Ausgehend von einer Basisklasse mit allgemeiner Funktionalität können schrittweise Unterklassen mit speziellerer Funktionalität abgeleitet werden. Damit ist ein hoher Grad an Datenabstraktion erreichbar. Der große Vorteil ist, daß ähnliche Problemstellungen innerhalb einer Klassenhierarchie gelöst werden können. Mehrfache Implementierungen bestimmter Funktionen lassen sich so weitestgehend vermeiden. Von vielen Klassen benötigte Funktionen werden in einer Oberklasse implementiert. Spezielle Funktionen werden in den Unterklassen implementiert. Das oft fehleranfällige Kopieren von Kodefragmenten entfällt. Eine bessere Funktionalität oder eine Fehlerbeseitigung in einer Oberklasse wirkt sich auf alle abgeleiteten Klassen aus. Die Wartbarkeit und Wiederverwendbarkeit des Programms wird damit verbessert.
Besitzt eine abgeleitete Klasse mehrere Basisklassen, dann spricht man von Mehrfachvererbung. Ein streng hierarchischer Aufbau von Klassenhierarchien kann damit nicht mehr erfolgen. Nicht alle objektorientierten Programmiersprachen unterstützen Mehrfachvererbung.
In *IPE* wurde das Konzept der Vererbung verwendet. Für verschiedene Datensatztypen existieren spezielle Betrachter und Editoren. Da deren Funktionalität teilweise ähnlich ist, kann das Konzept der Vererbung vorteilhaft angewendet werden. Der bei der Di-

plomarbeit erstellte Editor für polygonale Datensätze wurde von einer Editorbasisklasse abgeleitet. In Abbildung 2.21 werden die Zusammenhänge zwischen Ober- und Unterklassen bei Betrachtern und Editoren in *IPE* dargestellt.

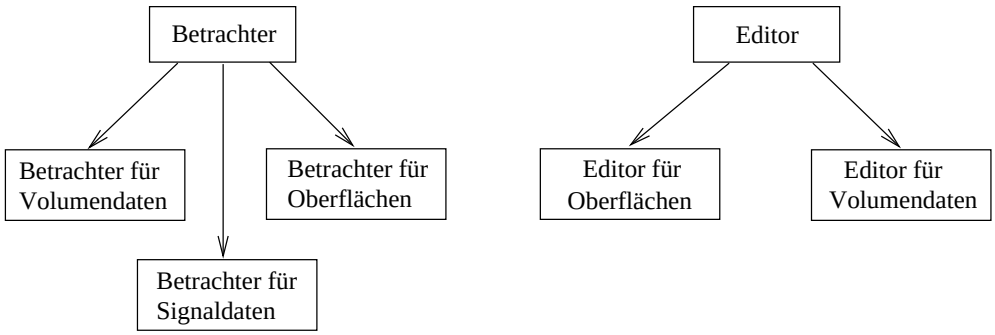


Abbildung 2.21: Klassenhierarchie für Betrachter und Editor.

2.4.1.4 Polymorphie

Unter Polymorphie versteht man, daß eine Methode in verschiedenen Zusammenhängen unterschiedlich wirken kann.
Es existieren verschiedene Arten von Polymorphismen. Das *Überladen* gestattet es verschiedenen Klassen, gleiche Methodennamen zu verwenden. Beim *parametrischen Polymorphismus* wird der gleiche Methodenname für verschiedene Datentypen und Parameter verwendet. *Überladen* und *parametrischer Polymorphismus* sind im wesentlichen syntaktische Möglichkeiten, die die Lesbarkeit von Programmen verbessern, indem für ähnliche Funktionen gleiche Namen benutzt werden. Mehrdeutigkeiten werden durch das System aufgelöst, indem es den Kontext bei jedem Auftreten des Namens überprüft.
In Compilersprachen wird mit *dynamischem Binden* (*late binding*) die Möglichkeit bezeichnet, die aufzurufende Methode eines Objektes erst zur Laufzeit des Programms zu bestimmen. Die Implementierung einer Methode in einer abgeleiteten Klasse kann von einer Basisklasse aus aufgerufen werden. Damit ist eine größere Flexibilität bei der Verwendung der Objekte möglich. Das geschieht in C++ durch Deklaration sogenannter *virtueller Methoden*, die in einer Basisklasse mit dem Schlüsselwort *virtual* gekennzeichnet wurden.

2.5 Visualisierung geometrischer Objekte des dreidimensionalen Raumes

Die Darstellung von 3D-Szenen ist ein sehr umfangreiches Gebiet. Hier können nur einige Aspekte dargestellt werden, die wesentliche Teile des Editors betreffen. Eine umfangreiche Darlegung der Thematik befindet sich in [FDFH96]. Zur Visualisierung wird auf die 3D-Graphikbibliothek Open GL[9] zurückgegriffen. Für Open GL ist auf vielen Rechnern im Max-Planck-Institut für neuropsychologische Forschung eine Hardwareunterstützung vorhanden. Unter Linux kommt der Open GL Ersatz MESA [MES99] zum Einsatz.

Open GL ist eine Software-Schnittstelle zur Graphikhardware. Sie besteht aus ca. 120 verschiedenen Kommandos, die für Objekte und Operationen benötigt werden, die zur Realisierung einer interaktiven 3D-Anwendung hilfreich sind [WND97]. Eine konkrete Open GL Implementierung sollte zweckmäßigerweise Hardwareunterstützungen für die einzelnen Funktionen verwenden (z.B. bei SGI-Rechnern realisiert), das ist aber nicht zwingend erforderlich. Viele Graphikkarten im PC-Bereich unterstützen nicht alle Open GL Funktionen und realisieren daher einen Teil der Berechnungen durch normalen Programmkode. Bei MESA werden alle Funktionen durch Software realisiert, deshalb ist es auf allen Plattformen und mit allen Graphikkarten lauffähig. Die erreichbare Darstellungsleistung ist entsprechend eingeschränkt.

Die Darstellung von Polygonen und Polygonnetzen mit Open GL ist relativ einfach. Für die Darstellung von Polygonen existieren spezielle Funktionen. Polygonnetze werden durch mehrere Polygone dargestellt, dazu ist für jedes Polygon ein Zeichenbefehl notwendig. Da die Polygone in einem Polygonnetz zusammenhängen und gemeinsame Kanten haben, müssen viele Vertizes mehrfach angegeben werden. Durch die Verwendung von Dreiecksstreifen (Trianglestrip) werden weniger Daten verarbeitet (Abbildung 2.22 a). Anstelle für jedes Dreieck drei Vertizes anzugeben, wird nur das erste Dreieck durch drei Vertizes spezifiziert. Nachfolgende Dreiecke ergeben sich aus der letzten Kante

[9] GL ist die Abkürzung für "Graphics Library" (Graphikbibliothek)

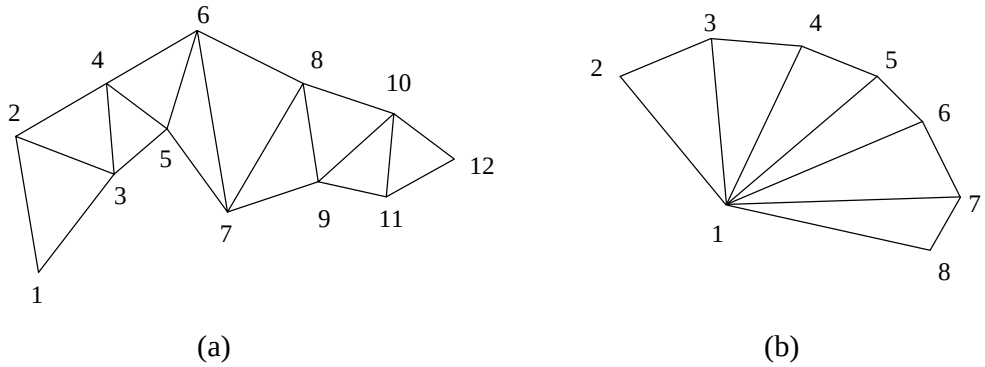


Abbildung 2.22: Durch Dreiecksstreifen (a) und Dreiecksfächer (b) kann die Darstellung eines Polygonnetzes optimiert werden. Die Zahlen geben die Reihenfolge der Vertizes an.

des vorher gezeichneten Dreiecks und einem neuen Vertex. Im Idealfall sinkt die Anzahl der anzugebenden Vertizes von $3n$ auf $n+2$ (n ist die Anzahl der Dreiecke). In der Praxis ergibt sich jedoch nur ein geringer Vorteil, da die Länge der einzelnen Dreiecksstreifen bei einfachen Konstruktionsverfahren nicht sehr groß wird. Es existieren spezielle Verfahren zur Konstruktion längerer Dreiecksstreifen. Eine Untersuchung verschiedener Verfahren befindet sich in [ESV96]. Nach jeder Anwendung einer topologieverändernden Operationen ist eine Neuberechnung der Dreiecksstreifen notwendig. Das erfordert Verfahren, die für eine interaktive Visualisierung geeignet sind. Da das Finden einer optimalen Triangulierung für die Konstruktion von Dreiecksstreifen NP-vollständig ist [ESV96], sind Konstruktionsverfahren mit sehr guten Ergebnissen und mit Rechenzeiten, die für eine interaktive Umgebung geeignet sind, kaum zu erwarten. Zur Konstruktion von Dreiecksstreifen wird ein einfaches und wenig zeitaufwendiges Verfahren benutzt. Ausgehend von einem beliebigen Startdreieck wird so lange nach Nachbardreiecken an der jeweils zuletzt angegebenen Kante gesucht, bis ein schon dargestelltes Dreieck gefunden wird. Die dargestellten Dreiecke werden markiert. Dann wird das nächste noch nicht markierte Dreieck gesucht, und von diesem aus ein neuer Dreiecksstreifen konstruiert. Die Konstruktion wird fortgesetzt, bis alle Dreiecke dargestellt sind. Die durchschnittliche Länge der erzeugten Dreiecksstreifen liegt bei ca. fünf Dreiecken, was eine Einsparung an Vertexangaben von über 50 Prozent bedeutet.

Ein ähnlicher Ansatz zur Optimierung sind Dreiecksfächer (Trianglefan), bei denen ebenfalls nur $n + 2$ Vertizes angeben werden müssen (Abbildung 2.22 b). Allerdings sind Dreiecksfächer weniger flexibel anwendbar als Dreiecksstreifen.

Für die Darstellung von Dreiecksstreifen und Dreiecksfächern sind unter Open GL spezielle Funktionen vorhanden.

Für Beleuchtungsberechnungen können in Open GL Vertexnormalen angegeben werden (Kapitel 2.2.1.2). Dadurch ist eine ansprechende räumliche Darstellung der Objekte erreichbar.

In vielen Programmen ist es üblich, die Rückseiten der im Bild befindlichen Flächen nicht darzustellen, was eine Verringerung der Anzahl der darzustellenden Polygone zur Folge hat. Die Entscheidung, ob nur die Rückseite des Polygons in der Szene zu sehen ist, kann mit Hilfe der Orientierung des Polygons getroffen werden. In Open GL existieren Einstellmöglichkeiten, ob nur Vorder-, nur Rückseiten oder Vorder- und Rückseiten der Polygone dargestellt werden sollen. Bei der Visualisierung von Hirnoberflächen ist die Darstellung der Rückseiten unumgänglich, da nur so ein realistischer Eindruck des zugrunde liegenden Datensatzes vermittelt werden kann.

Kapitel 3

Ein Editor für Polygonnetze

3.1 Aufgaben des Editors

Viele der aus Volumendaten erzeugten Polygonnetze lassen sich für weitere Untersuchungen nicht direkt verwenden und müssen daher nachbearbeitet werden. Die Komplexität z.B. der Hirnoberfläche ist so groß, daß oft interessante Details tief eingefaltet liegen und daher verdeckt sind. Mit vorhandenen Segmentationsverfahren lassen sich nicht alle gewünschten Aufgaben, wie z.B. das Extrahieren eines bestimmten Oberflächenteils, lösen. Der Editor soll gezielte Manipulationen an Polygonnetzen ermöglichen, wie z.B.:

- Verdeckte Strukturen im Polygonnetz sichtbar machen.
- Einzelne Teile des Polygonnetzes neu anordnen.
- Proportionen ändern.
- Unerwünschte Bereiche entfernen.
- Vertexeigenschaften ändern.

Aus diesen Anforderungen lassen sich mehrere Teilaufgaben ableiten, die zur Lösung der genannten Probleme notwendig sind:

- **Ausschneiden von Stücken aus einem Polygonnetz**
  Aus einem Polygonnetz wird ein vom Nutzer spezifiziertes Teilstück herausgeschnitten (Kapitel 2.3.1). Die Aufgabe teilt sich in Angabe des auszuschneidenden Teilstücks und den eigentlichen Ausschneidevorgang. Die Angabe des Teilstücks soll in einer für den Nutzer einfach zu handhabenden Weise erfolgen.

- **Löschen und Kopieren von Teilen eines Polygonnetzes**
  Einzelne oder mehrere Primitive sollen gelöscht oder kopiert werden. Die Selektion von Primitiven soll für den Benutzer mit einem zumutbaren Aufwand durchführbar sein.

- **Zusammenfügen von Polygonnetzen**
  Zwischen zwei Polygonnetzen soll eine Verbindung hergestellt werden. Nach der Operation sind beide Polygonnetze topologisch zusammenhängend. Da es sich um Flächen handelt, können Polygonnetze nur entlang von Rändern zusammengefügt werden, um die Konsistenzbedingung (eine Kante wird von maximal zwei Polygonen verwendet) zu erfüllen.

- **Drehen, Spiegeln, Skalieren, Verschieben von Teilen des Polygonnetzes**
  Eine vom Nutzer festgelegte Auswahl an Primitiven wird einer der oben genannten affinen Transformationen unterzogen. Die Auswirkungen der Änderungen werden dem Nutzer durch fortlaufende Visualisierung angezeigt.

- **Deformieren von Polygonnetzen**
  Das Polygonnetz wird durch eine spezielle Operation verformt (Kapitel 2.3.2).

- **Ändern von Farben**
  Für die Änderung der Vertexeigenschaft "Farbe“ wird die entsprechende Funktionalität benötigt. Es soll auf eine einfache Weise möglich sein, Farben auszuwählen, Farben vom Graphen zu übernehmen und Farbwerte im Graphen zu setzen.

3.2 Interaktivität

Der Editor ist ein interaktives Programm mit einer graphischen Nutzeroberfläche und wurde unter X Windows mit OSF Motif realisiert. Fast alle Eingaben werden graphisch mit Hilfe der Maus vorgenommen. Wegen der Darstellung der dreidimensionalen Daten auf einem zweidimensionalen Bildschirm und der mitunter komplizierten Form der betrachteten Oberflächen, ist es im allgemeinen notwendig, die Szene von verschiedenen Blickpunkten aus zu betrachten, um einen guten Überblick über die aktuelle Szene zu bekommen. Das kann am einfachsten durch Drehung der Szene erreicht werden. Bei einigen 3D-Operationen ist die Dekomposition in mehrere 2D-Operationen nötig, um zum gewünschten Ziel zu gelangen. Ein Beispiel ist die Verschiebung, bei der sich die gesamte Verschiebung in 3D auch als Komposition von 2D-Verschiebungen erreichen läßt. Die Auswirkung einzelner Operationen soll dem Nutzer möglichst sofort dargestellt werden, z.B. wird das Ergebnis einer Verschiebung fortlaufend angezeigt.

3.2.1 Eingabegeräte

Als Haupteingabegerät, sowohl für die Aktivierung von Editorfunktionen und Menüsteuerung, als auch für die Bestimmung von Funktionsparametern dient die Maus. Weitere Eingabegeräte sind Tastatur und Spaceball. Ein Spaceball ist ein Eingabegerät, welches speziell für 3D-Anwendungen entwickelt wurde. Ein Spaceball besitzt sechs Freiheitsgrade. Mit ihm kann ein Benutzer Eingabeparameter für eine Translation in 3D und für eine Rotation um eine beliebige Achse direkt erzeugen.

Bei der Implementierung einer einzelnen Editorfunktionen ist zu beachten, daß die Parameter in einer für den Nutzer nachvollziehbaren Weise an den Editor übertragen werden können. Bei der Bewegung eines Objektes soll sich das Objekt z.B. in Richtung der Mausbewegung bewegen. Alle Interaktionen können mit Hilfe einer Maus durchgeführt werden. Ein Vorteil der Maus liegt unter anderem darin, daß Manipulationen mit großer Genauigkeit durchführbar sind. Zudem ist die Maus das das einzige graphische Eingabegerät, das praktisch an jedem Rechner verfügbar ist. Die Verwendung eines Spaceballs bietet zwar mehr Freiheitsgrade bei der Wahl der Funktionsparameter, jedoch kann die Positionierung in der Raumtiefe nicht mit größerer Genauigkeit erfolgen. Das liegt hauptsächlich an der Art der Darstellung der Szene auf einem 2D-Bildschirm, welche die Wahrnehmung der tatsächlichen Tiefe des Objektes erschwert. Die Benutzung von 3D-Anzeigetechniken würde zwar Vorteile bringen, kann dieses Problem jedoch nicht vollständig lösen ([FDFH96], Kap. 8.2.6). Für die Navigation in einem Datensatz ist ein Spaceball in der Praxis jedoch gut geeignet.

3.2.2 Umwandlung von 2D-Eingabedaten in 3D-Daten

Die passenden Eingabeparameter für die einzelnen Editorfunktionen werden mit Hilfe der Mauskoordinaten bzw. Mausbewegung konstruiert. So wird z.B. für das Bewegen von Objekten ein Verschiebungsvektor bestimmt.

Die Umwandlung der Mauskoordinaten in 3D-Koordinaten erfolgt mit Hilfe von Open GL

Funktionen. Dabei werden die x-, y- und z-Koordinaten des Bildschirmpunktes in das Weltkoordinatensystem der dargestellten Objekte umgerechnet. Die x- und y-Koordinaten des Bildschirmpunktes ergeben sich direkt aus der Mausposition. Die z-Koordinate des Bildschirmpunktes wird durch Auswertung des Tiefenpuffers an der Mausposition bestimmt [WND97]. Für diese Methode ist entsprechend leistungsfähige Hardware notwendig. Um einen Verschiebungsvektor zu berechnen, kann für die z-Koordinaten ein fester Wert benutzt werden, da mit Hilfe der Maus i.a. nur in x- und y-Richtung des Bildschirms verschoben werden soll. Der Verschiebungsvektor ergibt sich dann aus der Differenz der umgewandelten Anfangs- und Endpunkte der Mausbewegung.

3.2.3 Spezielle Interaktionen

3.2.3.1 Rotationen mit Hilfe von Mausbewegungen

Die Manipulation der Orientierung eines Objektes im Raum erlaubt es, verschiedene Sichtweisen auf die Szene zu bekommen. Eine Rotation wird oft mit Hauptachsenrotationswinkeln (ϕ_x, ϕ_y, ϕ_z) beschrieben, d.h. man beschreibt die Rotation als Hintereinanderausführung von drei Rotationen, die jeweils um eine Hauptachse (x,y,z) erfolgen. Erlaubt man dem Benutzer die direkte Manipulation von $\phi_{[xyz]}$, so wird er nur in seltenen Fällen eine zufriedenstellende Umsetzung seiner Absichten feststellen. Die Zerlegung einer Rotation in Hauptachsenrotationswinkel ist nicht eindeutig, und löst dadurch zusätzliche Verwirrung beim Anwender aus. Andere Arten, eine Rotation zu repräsentieren, sind Quaternionen oder Matrizen.

Das Hauptproblem bei der Implementierung einer Rotationsoperation in einem Editor besteht in der Bereitstellung eines nutzerfreundlichen Rotationsschemas. Sind alle aktuell interessierenden Details von einer Position außerhalb der dargestellten Szene sichtbar, kann das Rotationszentrum in die Mitte der Szene gelegt werden. Stellt man sich die Szene eingebettet in eine Kugel vor, deren Mittelpunkt das Zentrum der Szene ist, kann die Orientierung der Szene durch Drehen dieser Kugel geändert werden. Für die Durchführung dieser Rotation existieren verschiedene Methoden. Einfache Lösungen speichern die ak-

tuelle Orientierung in Hauptachsenrotationswinkeln und setzen die Mausdifferenzen in horizontaler und vertikaler Richtung in die Hauptachsenrotationswinkel ϕ'_x und ϕ'_y um, die zu den Gesamtwinkeln ϕ_x und ϕ_y addiert werden. Eine andere Methode speichert die Orientierung mit Hilfe einer Matrix R. Die sich aus ϕ'_x und ϕ'_y ergebende Rotationsmatrix R' wird mit der Gesamtrotationsmatrix R multipliziert, so daß sich $R_{neu} = R'R_{alt}$ ergibt. Ein anderer Ansatz ist das "Arcball Rotation Control" aus [Hec94], welches mit Hilfe von Quaternionen realisiert ist und die aktuelle Orientierung mit Hilfe eines Quaternions repräsentiert. Die Bewegung der Maus wird auf die virtuelle Kugel projiziert. Aus dieser

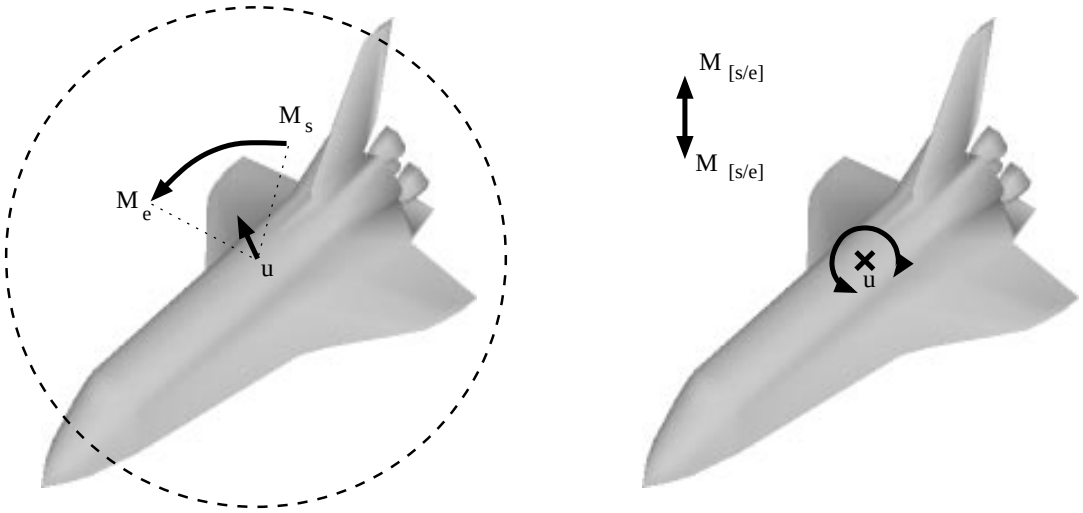


Abbildung 3.1: Vergleich von "Arcball Rotation" mit variabler Rotationsachse (links) und Rotation um die Achse, die senkrecht zur Bildschirmebene steht (rechts).

Projektion wird eine Rotationsachse und ein Rotationswinkel bestimmt. Die Rotation erfolgt in Richtung der Mausbewegung. Die aktuelle Orientierung der Szene ist dabei unabhängig von dem Pfad, entlang dessen die Maus bewegt wurde. Die Bedienung ist anwenderfreundlich und leichter verständlich als andere Methoden.

Wie gut man die Auswirkungen der Rotation eines Objektes verfolgen kann, ist von der Lage der Rotationsachse zur Bildschirmebene abhängig. Liegt die Rotationsachse senkrecht zur Bildschirmebene, läßt sich die Rotation am besten verfolgen. Aus diesem Grunde wird das Schema mit den in einer Kugel eingebetteten Objekten nur für die Rotation der gesamten Szene angewendet. Für Rotationen einzelner Objekte in der Szene wird die Rotation um die in den Bildschirm hineinzeigende Achse benutzt (Abbildung 3.1).

3.2.3.2 Orientierung in der Szene

Manchmal können nicht alle Details der Szene von Standpunkten außerhalb der Szene betrachtet werden. Oft werden einige Polygonnetzstrukturen von anderen verdeckt. Für diesen Fall kann man den Betrachterstandort nah an die interessierenden Details verlegen.

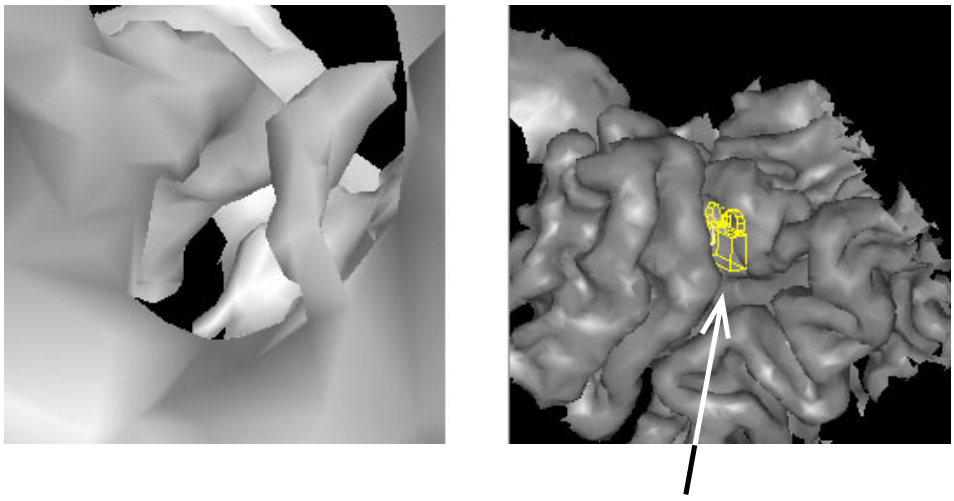


Abbildung 3.2: Positionierung innerhalb eines unübersichtlichen Polygonnetzes. Der Betrachterstandort ist als Kameramodell dargestellt.

Die Navigation in der Szene erfolgt ähnlich einer Kamerafahrt. Um die aktuelle Position des Betrachterstandortes in der Szene genau zu kennen, ist eine Ansicht der Szene vom Betrachterstandort aus und eine weitere für die Ansicht des Betrachterstandorts erforderlich (Abbildung 3.2). Die Navigation läßt sich sowohl mit einem Spaceball als auch mit der Maus durchführen.

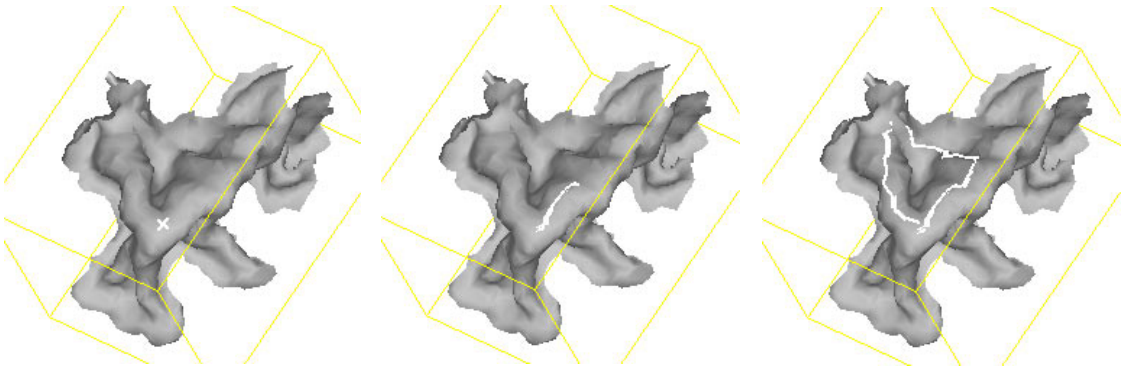


Abbildung 3.3: Interaktive Konstruktion eines Pfades auf einem Polygonnetz links: Startpunkt, rechts: Geschlossener Pfad.

3.2.3.3 Eingabe eines Pfades im Polygonnetz

Für das Ausschneiden von Stücken aus einem Polygonnetz muß zunächst ein Pfad konstruiert werden, der das auszuschneidende Gebiet angibt. Der Benutzer wählt sich zuerst einen Startpunkt aus. Dann klickt er auf eine Stelle des Polygonnetzes, die die Richtung angibt, in die der Pfad gezeichnet werden soll. Von dem aktuellen Ende des Pfades zur Mausposition wird ein Streckenzug konstruiert (Abbildung 3.3). Das Ausschneiden von Stücken kann nur erfolgreich verlaufen, wenn der Pfad von Rand zu Rand verläuft oder wenn der Pfad geschlossen ist (notwendige, aber keine hinreichende Bedingung). Informationen, wann ein Pfad geschlossen ist oder ein Rand erreicht wurde, können der Statuszeile des Editors entnommen werden.

3.3 Aufbau des Editors

3.3.1 Datenformate und Datenstrukturen

Polygonnetze bestehen aus Vertizes und Polygonen, welche in zwei Feldern abgelegt sind (Kapitel 2.2.1.4). Ein Polygonnetz kann als einzelnes großes Objekt aufgefaßt werden. Eine Unterteilung eines Polygonnetzes in mehrere, voneinander getrennte Objekte würde sich für viele Operationen vorteilhaft auswirken. Eine sinnvolle Möglichkeit, ein Polygonnetz in verschiedene Objekte aufzuteilen, ist die separate Betrachtung einzelner Zusammenhangskomponenten. Damit können einem Objekt grundlegende Eigenschaften,

wie Position und Orientierung in Form einer Transformationsmatrix zugeordnet werden, was eine einfache Handhabung von affinen Transformationen ermöglicht. Der Zugriff auf die Gesamtheit der Primitive eines speziellen Objektes ist im Gegensatz zur Verwaltung aller Primitive als einziges Objekt sehr einfach. Weiterhin läßt sich ein einfaches Undo-Konzept verwirklichen, daß auf den einzelnen Objekten basiert. Eine Zerlegung eines Polygonnetzes in mehrere logisch getrennte Objekte ist wegen der vorliegenden Daten und beabsichtigten Operationen jedoch nicht sinnvoll. Das liegt unter anderem darin, daß im Extremfall alle Objekte nur aus einem einzigen Polygon bestehen können. Bei der großen Anzahl an Primitiven scheidet diese Möglichkeit von vornherein aus. Bei topologieverändernden Operationen wäre mit einer Vielzahl von Objektverschmelzungen oder Trennungen zu rechnen, was sich negativ auf die Geschwindigkeit der einzelnen Operationen auswirken würde. Der verwendete Editor arbeitet deshalb direkt mit den Grundprimitiven des Polygonnetzes, ohne weitere Abstraktionsebenen zu benutzen. Eine Bearbeitung einzelner Bereiche des Polygonnetzes wird durch gezieltes Selektieren von Primitiven erreicht, wofür entsprechende Operationen implementiert wurden.

3.3.2 Operationen des Editors

Eine grobe Klassifizierung der einzelnen Editor-Operationen kann in topologieveränderende und topologieerhaltende Operationen vorgenommen werden. Die algorithmischen Grund-

	topologieveränderend	topologieerhaltend
Verschieben		×
Skalieren		×
Drehen		×
Spiegeln		×
Deformieren		×
Netze trennen	×	
Netze verbinden	×	

Tabelle 3.1: Klassifikation der Editor-Operationen

lagen zur Durchführung der einzelnen Aktionen werden in den Kapiteln 2.2 und 2.3 besprochen. Nachfolgend wird näher auf die Funktionsweise der Operationen im Editor

eingegangen.

3.3.3 Grundprinzip der Durchführung einer Editor-Operation

Die Ausführung einer einzelnen Editor-Operation läuft im wesentlichen in einer (*i*) Vorbereitungsphase und einer (*ii*) Ausführungsphase ab.
Zur Vorbereitungsphase gehört z.B. das Auswählen von Primitiven, die Eingabe eines Pfades oder die Positionierung der Hilfsebene sowie die Einstellung eines günstigen Blickpunktes auf die Szene.

Bei der Ausführungsphase kann man zwischen *kurzen Operationen* und *langen Operationen* unterscheiden. Die kurzen Operationen werden durch einen einmaligen Funktionsaufruf abgeschlossen (Spiegeln, Netze Trennen, Netze verbinden). Bei den langen Operationen wird ein Operationsparameter wiederholt verändert (Verschieben, Drehen, Skalieren, Verformen). Eine lange Operation ist erst beendet, wenn im Editorfenster eine andere Operation ausgewählt wird. Für eine Undo-Funktion sind erst nach Abschluß der Operation alle Parameter bekannt.

3.3.4 Undo Funktion

Um unbeabsichtigte Manipulationen an einem Polygonnetz rückgängig machen zu können, ist eine Undo-Funktion wichtig. Für die Implementierung einer Undo-Funktion stehen zwei Alternativen zur Verfügung:

(*i*) Die Erstellung einer Kopie des gesamten Graphen und

(*ii*) das Abspeichern der Editorfunktion mit ihren Parametern.

Bei (*ii*) wird eine Umkehroperation angewandt. Da einige Editor-Operationen auf eine Auswahl von Vertizes angewendet werden können, muß die Auswahl der betreffenden Vertizes mit den Parametern der Funktion abgespeichert werden. Die Anwendung einer Umkehrfunktion birgt die potentielle Gefahr einer numerischen Verfälschung des Graphen. Für manche Editorfunktionen sind Umkehrfunktionen komplex oder fehleranfällig. Bei (*ii*) wird für jede Editor-Operation eine spezielle Umkehrfunktion implementiert. Das

erschwert das Hinzufügen von neuen Operationen. Bei (*i*) erfolgt die Rücknahme einer Operation durch Austausch des Graphen gegen eine Kopie. Für (*i*) spricht als Hauptargument die Umgehung numerischer Probleme und die gesparte Zeit bei komplexen Umkehrfunktionen, für (*ii*) die Speicherplatzersparnis. Eine Kombination aus (*i*) und (*ii*) erscheint günstig und wurde implementiert. Dabei wird (*i*) für komplexere Funktionen und (*ii*) für einfache Funktionen mit weniger aufwendigen Umkehrfunktionen verwendet.

Bei Verwendung von (*i*) werden die Undo-Informationen *vor* Ausführung der Operation gespeichert. Falls (*ii*) benutzt wird, werden die Undo-Informationen *nach* Ausführung der Operation gespeichert. Es können mehrere Operationen rückgängig gemacht werden. Der Name der aktuell zurücknehmbaren Operation wird im Editorfenster angezeigt. Wird eine Operation zurückgenommen und sind noch weitere Operationen im Undo-Stack, rückt die nächste zurücknehmbare Operation auf den Platz der aktuell zurücknehmbaren Operation vor. Bei den langen Operationen *Verschieben*, *Drehen* und *Skalieren*, bei denen (*ii*) verwendet wird, tritt ein besonderer Fall auf. Die langen Operationen können unterbrochen werden, um die Szene zu drehen, und so die Auswirkung der Operation besser verfolgen zu können. Wird in dieser Situation die Undo Funktion ausgeführt, wird nicht wie sonst auf die vorige Operation des Undo-Stacks umgestellt, sondern die aktuelle Operation beibehalten.

3.3.5 Integration des Editors in die Visualisierungsoberfläche IPE

Zur Visualisierung von medizinischen Datensätzen wurde im Max-Planck-Institut für neuropsychologische Forschung das Programm *IPE* entwickelt. Den Aufbau von *IPE* zeigt Abbildung 3.4. Im mittleren Teil des Fensters befinden sich Rahmen, in denen Datensätze angezeigt werden können. Jeder Rahmen kann Datensätze darstellen und einen eigenen Editor für den in ihm enthaltenen Datensatz besitzen. Abbildung 3.6 zeigt *IPE* nach dem Laden eines Datensatzes und Starten des Editors. In Abbildung 3.5 ist eine grobe Struktur des *IPE* dargestellt. Ein zentrales Element ist der Datensatzmanager, der zum Lesen und Schreiben von Datensätzen von den Datenträgern und zum Zuordnen von Da-

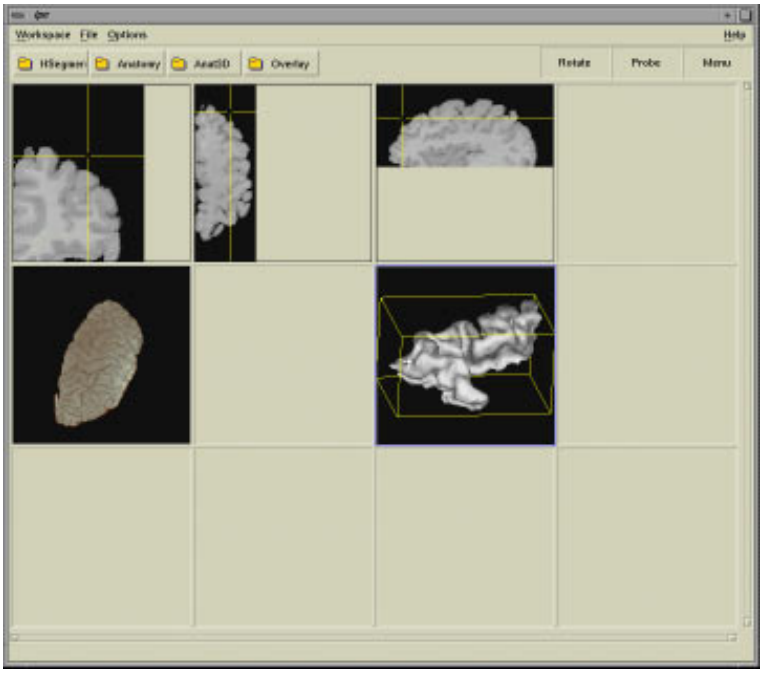


Abbildung 3.4: *IPE* nach dem Start. Oben, 2D-Schnittbilder; mitte links, Volumendarstellung; mitte rechts, Oberflächendarstellung (Polygonnetz).

tensätzen aus dem Datensatzpool zu einzelnen Rahmen dient. Einem Rahmen sind jeweils eine Menge von Datensätzen, ein Betrachter und ein Editor zugeordnet. Der Betrachter dient zur Visualisierung der Datensätze. Für jeden Datensatztyp existiert dabei ein spezieller Betrachter. Einem Rahmen können mehrere Datensätze zugeordnet werden. Der verwendete Editor ist abhängig vom Datensatztyp und arbeitet eng mit dem Betrachter zusammen. Für die Implementierung von Betrachter und Editor wird das Konzept der Vererbung (Kapitel 2.4.1.3) verwendet.

Editorfenster

Alle Funktionen sind über das Editorfenster erreichbar (Abbildung 3.7). In Anlehnung an die anderen Editoren in *IPE* ist das Editorfenster wie folgt aufgebaut: Rechts oben befinden sich Schaltflächen zur Auswahl einer Editorseite. Eine Editorseite beinhaltet eine Gruppe von logisch zusammengehörigen Steuerelementen für Editor-Operationen. Auf einer Editorseite befinden sich jeweils logisch zusammengehörige Steuerelemente. In Ab-

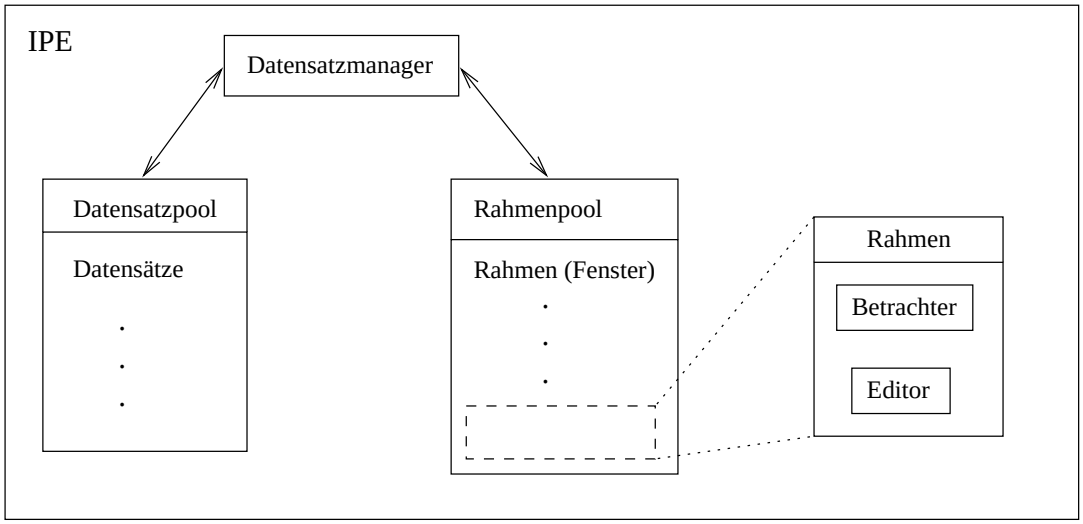


Abbildung 3.5: Grundaufbau von *IPE*.

bildung 3.7 ist die Editorseite für die geometrischen Operationen dargestellt. Es existieren Editmodi für die Vorbereitungsphase (z.B. Vertizes selektieren, Hilfsebene manipulieren und Hilfspfad eingeben) und Editmodi für die Ausführungsphase (z.B. Verschieben und Drehen). Editor-Operationen, die als Einzelaktion (Kapitel 3.3.3) aufgebaut sind, werden über die Schaltflächen für die Einzelaktionen erreicht.

In Abbildung 3.8 werden die entwickelten Auswahlelemente für die beiden Schemata zur Farbbearbeitung gezeigt, von denen eines Farbpaletten und das andere Echtfarben verwendet. Die Nutzung von Farbpaletten bietet die Möglichkeit, sich schwach unterscheidende Farbattribute durch stark voneinander abweichende Farben darzustellen. Für diese Zwecke wurden verschiedene Farbtabellen vordefiniert. Bei der Verwendung von Farbpaletten kann in der Farbauswahl ein Palettenindex ausgewählt werden. Mit dem Wertebereich kann eingestellt werden, wie die Indizes der Farbpalette auf die Farbwerte der Vertizes abgebildet werden sollen. Damit ist es möglich, Intervalle farblich besonders hervorzuheben. Werden Echtfarben verwendet, können die RGB-Werte des ausgewählten Farbfeldes genau eingestellt werden. Nach Anwahl einer Farbkomponente (RGB) am Rhombus wird der Wert über eine Mausbewegung modifiziert, der im unteren Teil der

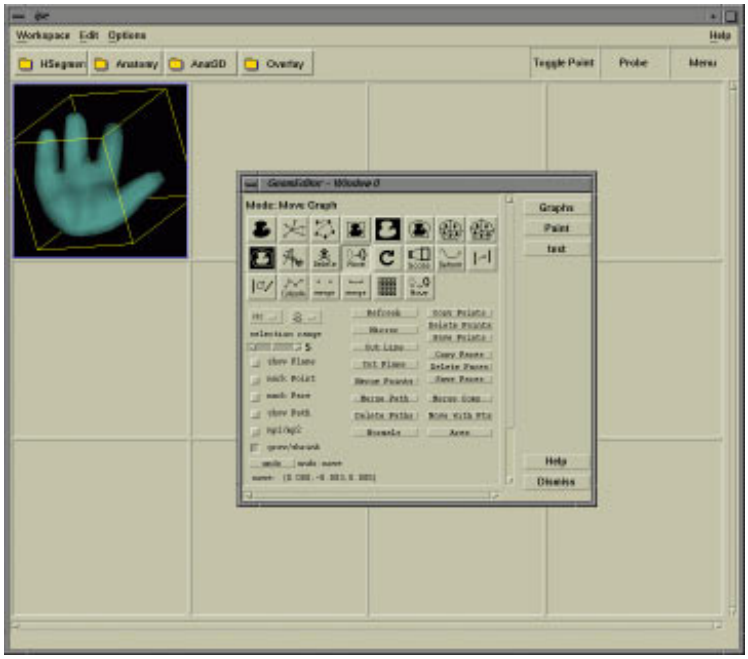


Abbildung 3.6: *IPE* mit einem Editor für geometrische Datensätze.

Farbauswahlkomponente angezeigt wird.

3.4 Vergleich mit anderen Editoren

Der Anwendungsbereich der meisten innerhalb anderer Softwarepakete vorhandenen Editoren ist weiter gefaßt als der des vorliegenden Editors. Oft besteht der Hauptverwendungszweck dieser Editoren in der Modellierung einer Szene für die spätere Darstellung in hoher Qualität oder zur Unterstützung der Konstruktion technischer Bauteile. Bei einigen Editoren ist es auch möglich, Animationen zu erstellen. Die meisten Editoren für dreidimensionale Szenen verwenden Grundobjekte wie Kugeln, Quader, Zylinder und Freiformflächen für die Modellierung (siehe auch Kapitel 2.1). Polygonnetze werden als eigene Grundobjekte angesehen, für deren Bearbeitung spezielle Funktionen zur Verfügung stehen. Ein Beispiel für diese Programme ist 3D Studio MAX [Imm97]. Bei diesem Programm gibt es umfangreiche Möglichkeiten, Polygonnetze, die dort als "Mesh-Objekte" bezeichnet werden, zu editieren. Neben der Möglichkeit, selektierte Primitive zu ver-

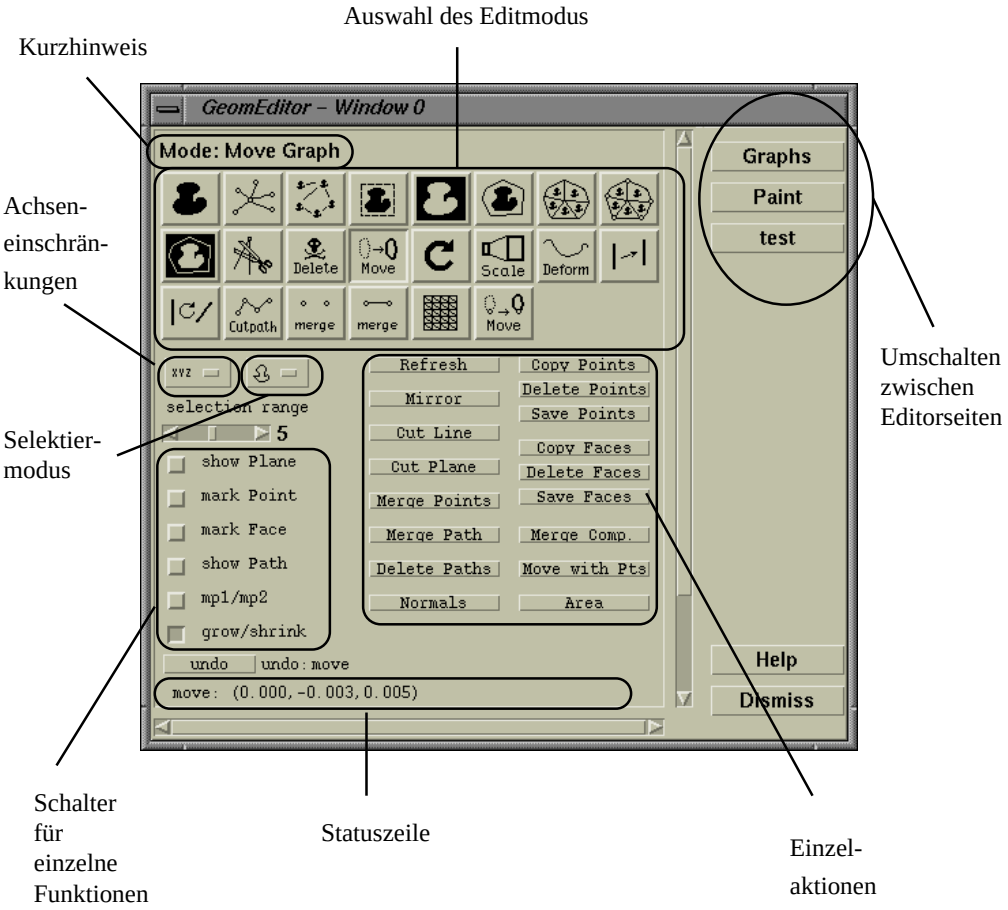
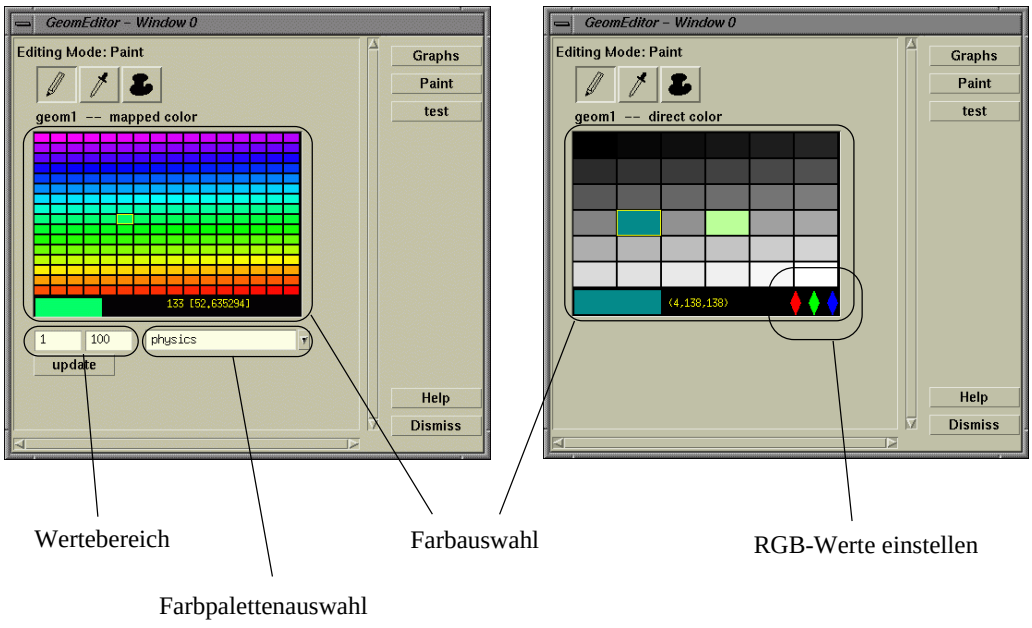


Abbildung 3.7: Editorfenster mit Seite für geometrische Operationen.

schieben, zu drehen oder zu skalieren, werden verschiedene Arten der Deformierung von Polygonnetzen angeboten. Ein Bearbeiten einzelner Primitive ist möglich. Die Bearbeitung von Polygonnetzen mit sehr vielen Primitiven ist jedoch sehr aufwendig. Funktionen, mit denen ein Hilfspfad komfortabel im Polygonnetz angegeben werden kann, um z.B. Primitive zu selektieren oder Stücke auszuschneiden, die für die Bearbeitung von Objekten mit vielen Primitiven hilfreich sind (Kapitel 3.2.3.3), existieren nicht.

Die Gründe, einen speziellen Editor zu entwickeln und in *IPE* zu integrieren, statt vorhandene Programme zu verwenden sind vielfältig. Zum einen sind verfügbare Editoren nicht für eine große Anzahl an Primitiven geeignet. Zum anderen sollte der Editor in das Soft-



(a) Auswahl mit Farbpaletten (b) Auswahl mit RGB-Werten

Abbildung 3.8: Editorfenster mit der Seite für Farbmanipulationen. (a) Farbpaletten (b) Echtfarben.

warepaket *BRIAN* (Brain Image Analysis) integriert werden. *BRIAN* besteht aus einer Kollektion von aufeinander abgestimmten Programmen zur Verarbeitung medizinischer Bilddaten. Neben Programmen zur Aufbereitung von MR-Rohdaten, Konvertierprogrammen, Bildfiltern und einer Vielzahl anderer Programme gibt es auch ein Programm zur Extraktion von Oberflächen aus Volumendaten. *IPE* ist Bestandteil des Softwarepakets *BRIAN*, das am Max-Planck-Institut für neuropsychologische Forschung entwickelt wird. Mit *IPE* können die erzeugten Datensätze visualisiert werden. Eine Erweiterung von *IPE* um Editierfähigkeiten für Oberflächen ergibt eine höhere Funktionalität des Bildverarbeitungssystems.

In *IPE* besteht die Möglichkeit mehrere Datensätze, auch unterschiedlichen Typs darzustellen. So können z.B. von einem Objekt gleichzeitig 2D-Schnittbilder und die dazu korrespondierende 3D-Oberfläche visualisiert werden. Eine spezielle Funktionalität von

IPE ist die gekoppelte Darstellung der Cursorposition von mehreren Bildern auch verschiedenen Datentyps. Mit der gekoppelten Darstellung kann die Cursorposition in allen Bildern gleichzeitig an den selben Koordinaten dargestellt werden. Damit ergibt sich ein besserer Überblick von dem betrachteten Objekt. Mit vorhandenen Editoren für Oberflächendaten könnte diese Funktionalität nicht erreicht werden.

Kapitel 4

Polygondatenreduktion

Ziel der Polygondatenreduktion ist es, ein mit einem Polygonnetz dargestelltes Objekt durch ein anderes ähnliches Polygonnetz darzustellen, welches wesentlich weniger Polygone enthält. Mit automatischen Verfahren erzeugte Polygonnetze besitzen oft eine sehr große Anzahl an Polygonen. Eine Verkleinerung der Polygonanzahl im Polygonnetz ist erforderlich, weil die Anzahl der Polygone so groß ist, daß eine interaktive Visualisierung dieser Polygonnetze nicht möglich ist. In den Ausgangsnetzen werden oft relativ ebene Flächen durch viele Polygone dargestellt. Läßt man kleine Abweichungen vom Originalnetz zu, können diese Abschnitte mit weniger Polygonen dargestellt werden. Für die Polygondatenreduktion gibt es verschiedene Ansätze, von denen einige im Kapitel 4.3 beschrieben sind.

4.1 Anforderungen

Eine Netzdezimierung kann unter Berücksichtigung einer Reihe von Anforderungen vorgenommen werden. Für viele Berechnungen an Polygonnetzen sind Dreiecke mit sehr spitzen Winkeln ungeeignet. Bei der Visualisierung von Netzen sind harte Übergänge ungünstig. Durch eine möglichst kleine Anzahl von Primitiven werden kurze Antwortzeiten erreicht. Das dezimierte Polygonnetz soll sich von dem Originalnetz möglichst wenig unterscheiden. Neben den geometrischen Abmessungen (Längen, Durchmesser etc.) sollen insbesondere die topologischen Eigenschaften des Originalnetzes erhalten bleiben. Saliente Strukturen (z.B. Nase, Ohren) sollen ebenfalls erhalten bleiben. Je nach Verwendungszweck können

bei einigen Forderungen weniger restriktive Kriterien angewendet werden. Wird z.B. ein Haus in einer Szene in großer Entfernung dargestellt, so können kleine Fenster des Hauses weggelassen werden, ohne das sich das störend auf die Qualität der Darstellung auswirkt. Bei einer Berechnung von einströmenden Luftmengen am Fenster, dürfen solche Merkmale bei der Dezimierung jedoch nicht verlorengehen. Bei der Bearbeitung medizinischer Bilddaten sind Änderungen an der Topologie des Netzes strikt zu vermeiden.

4.2 Level of Detail (LOD)

Ein Aspekt der Polygondatenreduktion ist die Erstellung einer "Level of Detail" Hierarchie (LOD) eines Objektes. Mit "Level of Detail" bezeichnet man Polygonnetzdarstellungen eines Objektes in unterschiedlichen Qualitätsstufen mit unterschiedlicher Anzahl an Primitiven (Dreieck, Vertex). Die einzelnen Darstellungen sind dabei einander ähnlich. Bei der Darstellung einer Szene können dann für weit entfernte Objekte einfachere und für nahe am Betrachter befindliche Objekte detailreiche Darstellungen des Objektes zur Visualisierung herangezogen werden. Weiterhin ist mit LODs eine Anpassung an die vorhandene Rechenleistung möglich.

Ein verbreitetes Konzept bei LOD - Hierarchien ist die Speicherung mehrerer Polygonnetze, die jeweils eine bestimmte Qualitätsstufe repräsentieren. Die Virtual-Reality-

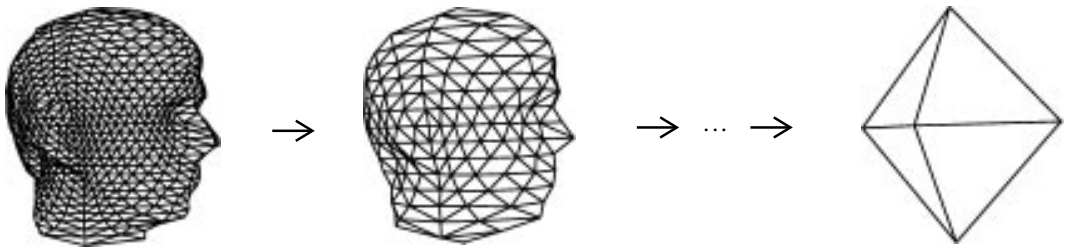


Abbildung 4.1: Beispiel für eine LOD-Hierarchie (aus [EDD+95]).

Beschreibungssprache VRML nutzt diese Form das LOD-Prinzips. Für die einzelnen Qualitätsstufen läßt sich der Bereich des Abstands von der Betrachterposition angeben, in dem das jeweilige Modell zur Darstellung benutzt werden soll. Eine Speicherung vie-

ler Qualitätsstufen ist wegen des enormen Speicherplatzbedarfs nicht sinnvoll. Bei der Verwendung weniger Qualitätsstufen wird der Übergang von einer Qualitätsstufe zur nächsten vom Betrachter deutlich wahrgenommen. Bei der Darstellung wird plötzlich zwischen verschiedenen Qualitätsstufen umgeschaltet, was auch als Popping-Effekt bezeichnet wird. In [Hop96] wird ein Verfahren beschrieben, mit welchem sich eine nahezu stufenlose LOD-Hierarchie erzeugen läßt, die keinen erhöhten Speicherplatzbedarf hat. Das Verfahren basiert auf der Umkehrung einer Polygondatenreduktionsmethode. Bei der Reduktion wird als einzelne Operation eine Kantenkontraktion durchgeführt. Dabei werden ein Vertex und zwei Dreiecke entfernt. Die Umkehrung dieses Operators wird als Vertex-Split bezeichnet (Abbildung 4.2). Mit dem Operator der Kantenkontraktion ist ein

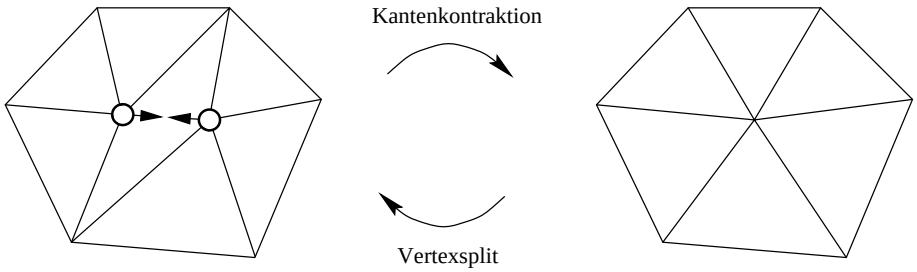


Abbildung 4.2: Kantenkontraktion und Vertexsplit.

effektives Verfahren zur Polygondatenreduktion durchführbar. Als Ergebnis der Reduktion erhält man ein Basisnetz, welches eine grobe Darstellung des ursprünglichen Objektes repräsentiert. Das Polygonnetz wird dann mit Hilfe des Basisnetzes und einer Folge von Vertex-Split Operationen gespeichert. Eine solche Darstellung eines Polygonnetzes wird als "progressives Polygonnetz“ bezeichnet und ist gut für Datenfernübertragung geeignet. Das Polygonnetz kann am Bildschirm dargestellt werden, sobald das Basisnetz übertragen wurde. Bei der weiteren Übertragung wird das Netz schrittweise (progressiv) mittels der Vertex-Split Operationen verfeinert.

4.3 Allgemeine Prinzipien der Polygondatenreduktion

In den letzten Jahren wurden zahlreiche Arbeiten veröffentlicht, die sich mit der Polygondatenreduktion beschäftigen (z.B. [SZL92], [Hop96], [EDD+95], [GR94], [KCS98], [CKS98], [CCMS97], [CVM+96]). Die wichtigsten Verfahren werden nachfolgend vorgestellt.

4.3.1 Clustering Verfahren

Der Raum, in dem sich das Polygonnetz befindet, wird in gleichgroße Volumensegmente zerlegt. Alle Vertizes innerhalb eines Segments werden zu einem Vertex zusammengefaßt. Die Qualität bzw. Abweichung des reduzierten Netzes ist abhängig von der verwendeten Größe der Segmente. Topologische Strukturen, die vollständig innerhalb eines Segments liegen, können nicht erhalten werden.

4.3.2 Re-Tiling Verfahren

Über das Polygonnetz wird eine bestimmte Anzahl neuer Vertizes verteilt und mit ihrer Hilfe wird das Netz neu konstruiert. Die Verteilung der Vertizes erfolgt abhängig von der lokalen Krümmung. Bereiche großer Krümmung erhalten mehr Vertizes. Probleme treten auf, wenn die Anzahl der verteilten Vertizes zu klein ist oder ihre Verteilung ungünstig vorgenommen wurde. Eine Messung der Abweichung des erzeugten Netzes vom Originalnetz ist sehr aufwendig.

4.3.3 Inkrementelle Verfahren

Mit Hilfe eines topologischen Operators werden am Polygonnetz kleine netzvereinfachende Änderungen vorgenommen. Der topologische Operator verändert dabei die Dreiecksstruktur eines Polygonnetzes. Der hierbei erzeugte Fehler wird berechnet und mit einer vorgegebenen Schranke verglichen. Es werden so lange Vereinfachungen vorgenommen, bis Fehlergrenzen überschritten werden. Am häufigsten werden die Knoten- und die Kantenentfernung eingesetzt (Abbildungen: 4.2 und 4.3). Eine Kantenkontraktion entfernt einen Vertex und zwei Dreiecke. Die beiden Vertizes der entfernten Kante werden zu ei-

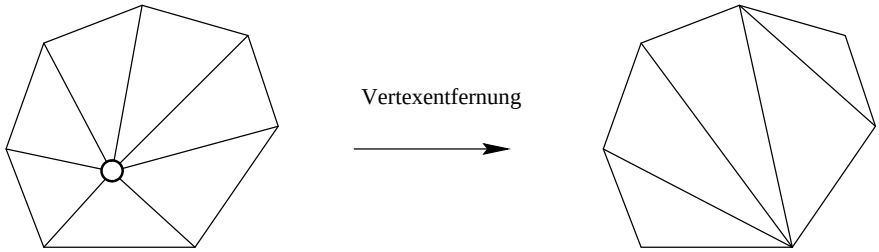


Abbildung 4.3: Topologischer Operator Vertex entfernen.

nem Vertex verschmolzen (Abbildung 4.2). Die Wahl der Koordinaten des verbleibenden Vertexes der entfernten Kante im Netz läßt sich variieren. Meist wird der Mittelpunkt der entfernten Kante benutzt. Es ist aber auch denkbar, anstelle neuer Koordinaten die Koordinaten eines alten Vertex der entfernten Kante zu benutzen [KCS98], was dann als Halbkantenkontraktion bezeichnet wird. Die Vertexentfernung läßt mehr Freiheitsgrade bei der Netzveränderung zu als die Kantenkontraktion. Nach Entfernung des Vertexes wird das entstehende Loch neu trianguliert. Diese Triangulierung läßt sich optimieren, um den lokalen Fehler zu minimieren ([CCMS97], [SL96]). Die meisten der veröffentlichten Verfahren verwenden inkrementelle Methoden (z.B. [SZL92], [KCS98], [CCMS97], [GR94], [Hop96], [CVM$^+$96], [SL96]). Bei den inkrementellen Verfahren kann man zwischen lokalem und globalem Fehler unterscheiden. Während des Ablaufs der Reduktion wird aus dem Originalnetz S_0 eine Folge von Polygonnetzen $S_0 \rightarrow S_1 \rightarrow \ \ldots \ \rightarrow S_n$ erzeugt. Zwei aufeinanderfolgende Polygonnetze S_i und S_{i+1} unterscheiden sich in einem Teilnetz T_i, welches durch einen topologischen Operator in ein Teilnetz T_{i+1} umgewandelt wurde. Der lokale Fehler, die Abweichung von T_i zu T_{i+1}, läßt sich z.B. durch Berechnung des Hausdorff-Abstands (siehe Definition 4.5.1) bestimmen. Eine Näherung für die Berechnung des Hausdorff-Abstands kann z.B. durch Verteilung von Testpunkten auf T_i und Berechnung des größten Abstands zu T_{i+1} erfolgen [SL96]. Der globale Fehler gibt die maximale Abweichung von S_i zu S_0 an.

Um den lokalen Fehler an einer bestimmten Stelle im Polygonnetz festzuhalten, gibt es mehrere Möglichkeiten. Man kann einem Dreieck einen Fehlerwert zuordnen, zu dem bei

jedem Reduktionsschritt, bei dem das Dreieck verändert wurde, der Fehler des aktuellen Reduktionsschrittes addiert wird. Mit dieser Methode wird die wahre Abweichung überschätzt, da lokale Abweichungen in beide Richtungen bezüglich des Originalnetzes gleichermaßen berücksichtigt werden. In [KCS98] wird einem Dreieck eine Liste von Vertizes des Originalnetzes zugeordnet, die bei einzelnen Reduktionsschritten entfernt wurden. Für die Berechnung des Hausdorff-Abstands wird der Abstand dieser Vertizes zu den Dreiecken von T_{i+1} herangezogen, was eine bessere Näherung ergibt. Bei [CVM+96] wird eine innere und eine äußere Hülle des Polygonnetzes bestimmt. Die Einhaltung einer maximalen Abweichung wird durch Test auf Berührung mit diesen Hüllen durchgeführt.

4.4 Bemerkungen zur Kantenkontraktion

Bei Verwendung der Kantenkontraktion als topologischen Operator (Abbildung 4.2) können Inkonsistenzen im Polygonnetz erzeugt werden. Aus Abbildung 4.4 geht hervor,

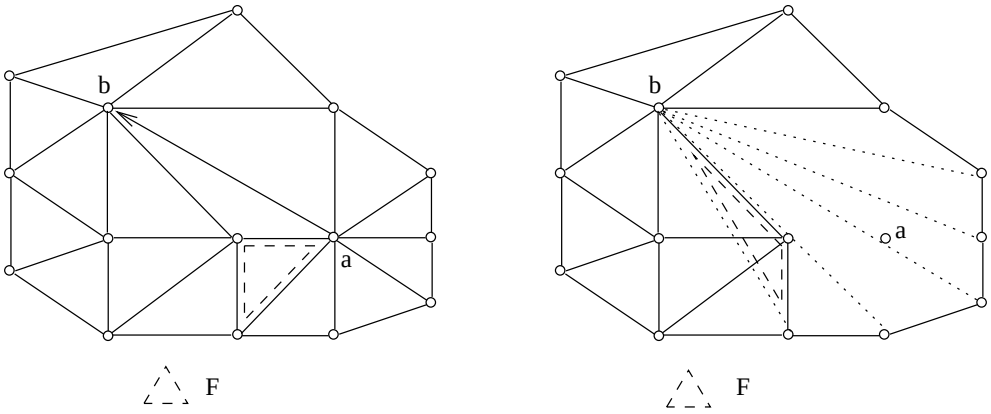


Abbildung 4.4: Beispiel für das Entstehen einer Inkonsistenz nach einer Halbkantenkontraktion. Das gestrichelt gezeichnete Dreieck F überlappt nach der Kontraktion andere Dreiecke.

daß Polygonnetzstrukturen mit überlappenden Dreiecken entstehen können. Vermieden werden kann dieser Vorgang durch Bestimmung der Änderung der Dreiecksnormalen bei der Kontraktion. Eine weitere Inkonsistenz kann auftreten, wenn es außer den Vertizes der zu entfernenden Dreiecke einen Vertex gibt, der beide Vertizes der zu entfernenden Kante als Nachbarn hat, was durch entsprechenden Test vermieden werden kann [GR94]

(Abbildung 4.5).

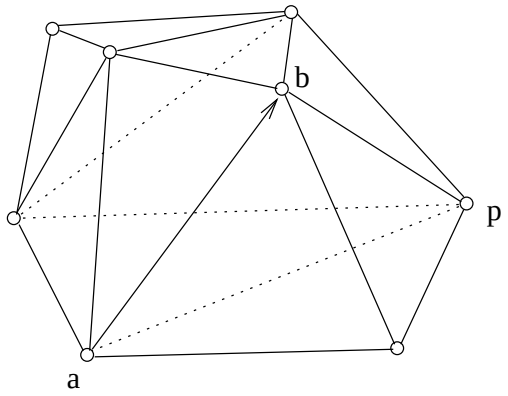


Abbildung 4.5: Beispiel für das Entstehen einer Inkonsistenz nach einer Halbkantenkontraktion. Bei Kontraktion von a nach b würden zwei Dreiecke aufeinander liegen.

4.5 Messung der Abweichung des vereinfachten Polygonnetzes

Ein dezimiertes Polygonnetz soll das Originalnetz möglichst gut annähern. Die Berechnung der geometrischen Abweichung zweier Polygonnetze S_1 und S_2 ist sehr komplex und rechenaufwendig, wenn alle Primitive beider Polygonnetze in die Berechnung einbezogen werden. In [CRS97] wird ein Verfahren beschrieben, mit dem Abweichungen zwischen einem Originalnetz und einem vereinfachten Netz bestimmt werden können. Da das Originalnetz und das vereinfachte Netz einander ähnlich sind, kann die Bestimmung der Abweichung vereinfacht werden. Basis für eine Aussage über die Abweichung von zwei Polygonnetzen, ist der Hausdorff-Abstand.

Definition 4.5.1 (Hausdorff-Abstand nach [PJS92]) *Sei X ein vollständiger metrischer Raum mit der Metrik d. Für jede kompakte Teilmenge A von X und $\epsilon > 0$ ist der ϵ-Kragen von A, definiert durch*

$$A_\epsilon = \{x \in X \mid d(x, y) \leq \epsilon \text{ für irgendein } y \in A\}$$

Für zwei beliebige kompakte Teilmengen A und B von X beträgt der Hausdorff-Abstand

$$h(A, B) = \inf\{\epsilon \mid A \subset B_\epsilon \text{ und } B \subset A_\epsilon\}$$

Der Hausdorff-Abstand $h(A, B)$ gibt den Abstand zwischen zwei Mengen an. Der Hausdorff-Abstand bildet eine Metrik über die Menge von allen geschlossenen, begrenzten Mengen. Er erfüllt die Eigenschaften der Äquivalenz, Symmetrie und Dreiecksungleichung. Betrachtet man zwei Polygonnetze als die Mengen A und B, beschreibt der Hausdorff-Abstand die geometrische Abweichung zwischen beiden Netzen. Die Bestimmung der maximalen geometrischen Abweichung zweier Polygonnetze ist sehr aufwendig, da eine Vielzahl von Polygonen in die Berechnungen einbezogen werden müssen. Es werden daher Vereinfachungen vorgenommen. Verzichtet man bei $h(A, B)$ auf die Bestimmung von ϵ in $B \subset A_\epsilon$, erhält man den einseitigen Hausdorff-Abstand von A und B. Für das Reduktionsverfahren wurde eine Näherung des Hausdorff-Abstands ähnlich wie in [KCS98] implementiert. Bei inkrementellen Verfahren ergibt sich eine Vereinfachung der Berechnungen, wenn lediglich das in einem Reduktionsschritt veränderte Teilnetz zur Bestimmung des Hausdorff-Abstands herangezogen wird (Abbildung 4.6). Es wird jeweils

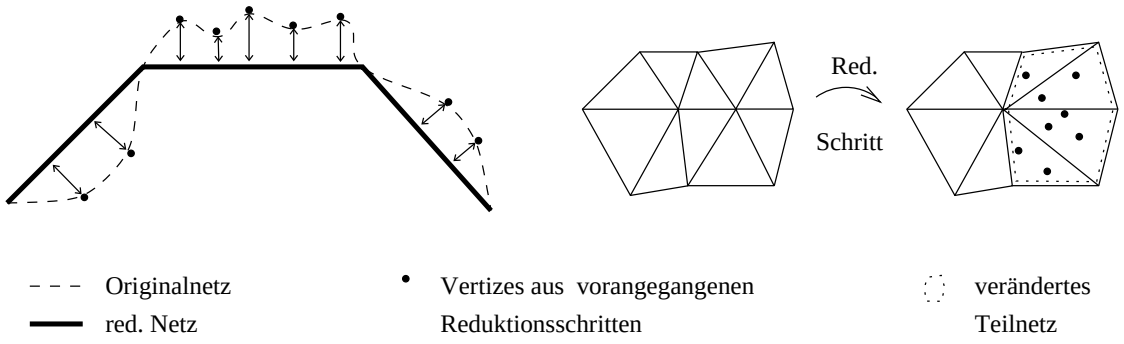


Abbildung 4.6: Vereinfachte Bestimmung des einseitigen Hausdorff-Abstands zwischen dem Originalnetz und dem reduzierten Netz.

der Abstand eines Vertexes, der bei vorangegangenen Reduktionsschritten übrig blieb, zu dem veränderten Teilnetz bestimmt. Für die Bestimmung des Abstands eines Vertexes wird das nächstgelegene Dreieck des veränderten Teilnetzes benutzt. Das Maximum dieser Abstände ergibt den für den Hausdorff-Abstand angesehenen Wert.

4.6 Grundschema der inkrementellen Verfahren

Für die meisten inkrementellen Verfahren wird folgendes Schema verwendet ([CKS98], [SL96]):

1. Berechne die Kosten für die Anwendung des topologischen Operators auf einen bestimmten Vertex oder eine Kante und lege das Ergebnis in einer Warteschlange ab.

2. Solange sich noch Einträge in der Warteschlange befinden, wird der Reduktionsschritt des Eintrags mit den niedrigsten Kosten im Polygonnetz durchgeführt, und die dabei veränderten Kosten in der Warteschlange aktualisiert.

Die Kostenfunktion hängt von den Gütekriterien und der verwendeten Metrik ab. Die bestimmten Werte können in einer Bewertungsfunktion quadratisch oder absolut eingehen. Neben den geometrischen Eigenschaften, wie lokale Krümmung und Abstand, werden auch die Winkel der entstehenden Dreiecke zur Berechnung der Kostenfunktion herangezogen. In [Hop96] werden zusätzlich die Farbattribute zur Kostenberechnung verwendet, d.h. Farbübergänge im Polygonnetz können erhalten bleiben.

4.7 Die Polygondatenreduktion als Optimierungsaufgabe

Die Polygondatenreduktion kann als Optimierungsproblem aufgefaßt werden. Bei vorgegebenem Fehler soll ein Polygonnetz mit möglichst geringer Anzahl an Primitiven erzeugt werden. Für die Optimierung sind Strategien zu entwickeln, die mit den vorhandenen, meßbaren Kriterien auskommen. Aufgrund der durch die hohe Anzahl an Primitiven entstehenden Komplexität ist nicht damit zu rechnen, daß für die vorgegebenen Parameter ein optimales Polygonnetz gefunden wird.

4.7.1 Kriterien

Für die Bewertung eines einzelnen Reduktionsschrittes werden mehrere Kriterien benutzt. Ein Maß ist die geometrische Abweichung beider Polygonnetze (E). Ein anderes Maß ist die Größe des Volumens, das sich zwischen Original und vereinfachtem Netz befindet. Ein

Maß für die Qualität der erzeugten Dreiecke ist das Verhältnis vom Radius des Inkreises zur längsten Dreiecksseite (R). Angestrebt wird ein möglichst kleiner Wert für R, damit so spitze Winkel in Dreiecken vermieden werden. (Abbildung 4.7).

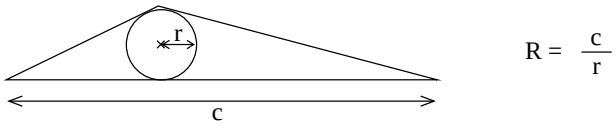


Abbildung 4.7: Kriterium für die "Qualität" eines Dreiecks. Verhältnis von der längsten Dreiecksseite zum Radius des Inkreises [KCS98].

Für die Bestimmung der lokalen Krümmung S ist die Idee aus [KCS98] geeignet. Es wird die Summe der Winkel gebildet, die zwischen den Dreiecksnormalen der Dreiecke des zu vereinfachenden Teilstücks und den anliegenden äußeren Dreiecken liegen (Abbildung 4.8). Diese Winkelsumme ist zwar von der Anzahl der Dreiecke abhängig, stellt aber in

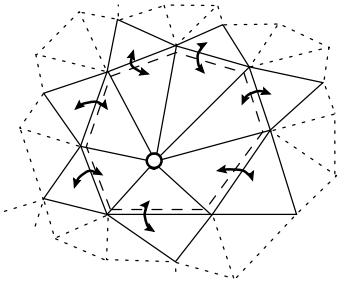


Abbildung 4.8: Bestimmung der lokalen Krümmung nach [KCS98].

der Praxis ein günstiges und leicht zu berechnendes Maß dar. Aus der Differenz vor und nach der Vereinfachung kann eindeutig bestimmt werden, ob die Krümmung zugenommen oder abgenommen hat.

Weil in einem vereinfachten Polygonnetz typischerweise größere schwach gekrümmte Flächenstücke entstehen, ist eine Verkleinerung der Winkelsumme anzustreben.

Da sehr viele Reduktionsschritte notwendig sind, darf die Berechnung der Kriterien nicht

übermäßig viel Zeit in Anspruch nehmen. Die Reduktion wird beendet, wenn das Kriterium E eine vorgegebenen Grenzwert e überschreitet. Dabei ist e der maximale Abstand, den das vereinfachte Netz zum Original haben darf und wird als Bruchteil ϵ der Diagonalen des das Gesamtnetz umschließenden Quaders gewählt. Durch diese Festlegung von e können für verschiedene Modelle vergleichbare e-Werte bei gleichem ϵ erzeugt werden.

Für das Kriterium der Dreiecksqualität R ist es sinnvoll, eine obere Schranke anzugeben, die bei einem Reduktionsschritt nicht überschritten werden darf. Weiterhin darf die Änderung einer Dreiecksnormalen einen bestimmten Wert nicht überschreiten, um ein konsistentes Polygonnetz[10] zu behalten. (z.B. keine Normalenänderungen größer als 25°).

4.7.2 Fairneßfunktion

Bei dem Reduktionsverfahren wird zwischen Kosten- und Fairneßfunktion unterschieden. Die Kostenfunktion gibt an, inwieweit das erlaubte Fehlermaß schon erschöpft ist, d.h. ob die geometrische Abweichung vom Original innerhalb der erlaubten Toleranz liegt. Die Fairneßfunktion bewertet einen einzelnen Reduktionsschritt hinsichtlich seiner Aussichten, spätere Reduktionsschritte zu begünstigen. Mit den berechneten Kriterien eines Reduktionsschrittes läßt sich eine Bewertungsfunktion aufstellen. Der Reduktionsschritt mit der größten Fairneß wird dann als nächster ausgeführt. Eine wesentliche Eigenschaft der Polygondatenreduktion ist, daß nachfolgende Reduktionsschritte stark abhängig von vorangegangenen sind. Eine Variierung der Ausführungsreihenfolge über mehrere Reduktionsschritte hinweg mit dem Ziel einer besseren Optimierung ist aus Gründen des erhöhten Rechenzeitbedarfs abzulehnen. Beschränkt man sich bei der Auswahl des nächsten Reduktionsschrittes auf die Bewertung der geometrischen Abweichung, erreicht man keine gute Optimierung. Deshalb werden weitere Kriterien zur Bewertung herangezogen.

Mit Wissen über allgemeine Eigenschaften von Polygonnetzen kann man die Bewertungs-

[10] siehe Kapitel 4.4

funktion verbessern. Polygonnetze besitzen im allgemeinen größere planare Bereiche, lokale Krümmungen im Polygonnetz sollen eher weniger ausgeprägt sein. Dreiecke mit sehr spitzen Winkeln und langen Kanten verschlechtern die Aussichten auf nachfolgende Reduktionsschritte. Achtet man bei einzelnen Reduktionsschritten auf die Qualität des entstehenden Polygonnetzes (spitze Winkel, lokale Krümmung), können bessere Optimierungsergebnisse gefunden werden. Führt man am Vertex p einen Reduktionsschritt durch, so läßt sich mit den drei Kriterien geometrische Abweichung $E(p)$, Dreiecksqualität $R(p)$ und lokale Krümmung $S(p)$ folgende Bewertungsfunktion F aufstellen [KCS98].

$$F(S) \;=\; \sum_{p \in S} \alpha\, E(p) + \beta\, R(p) + \gamma\, S(p)$$

Durch entsprechende Wahl der Parameter α, β und γ können die Eigenschaften der erzeugten Polygonnetze beeinflußt werden. Bei Dominanz von α bleiben Details gut erhalten, die Dominanz von β führt zu Dreiecken mit günstigen Winkeln, während die Dominanz von γ Polygonnetze mit einer glatten Oberfläche erzeugt. Die gewählten Parameter wirken bei verschiedenen Ausgangsnetzen sehr unterschiedlich. Weiterhin beeinflussen die oberen Schranken der Dreiecksqualität R und Änderung der lokalen Krümmung S das Ergebnis.

Untersuchung der Parameter der Fairneßfunktion

Als Untersuchungsobjekte dienen Polygonnetze mit unterschiedlichen Krümmungen, und Dreiecksanzahlen (Abbildung 4.9). Das Modell "glatte Fläche“ besteht aus 200 Dreiecken und ist vollkommen eben. Eine geometrische Abweichung kann nur bei Veränderung des Randes auftreten. Das Modell "Kugel“ besteht aus 1996 Dreiecken, deren Vertizes gleichmäßig auf der Kugeloberfläche verteilt sind. Das "Stanford-Bunny“ Modell ("Hase“) wurde mit Hilfe eines 3D-Scanners erzeugt und besitzt ca. 69000 Dreiecke. Das Modell "Trompete“ besitzt ca. 28000 Dreiecke. Das Modell "Hirnoberfläche I“ wurde aus Volumendaten erzeugt und besitzt ca. 48000 Dreiecke. Das Modell "Hirnoberfläche II“ wurde ebenfalls aus Volumendaten erzeugt, wurde aber schon mit einem anderem Polygondatenreduktionsverfahren bearbeitet ([GR94]). Es besteht aus ca. 11000 Dreiecken. Bei den Hirnoberflächen existieren neben größeren, relativ schwach gekrümmten

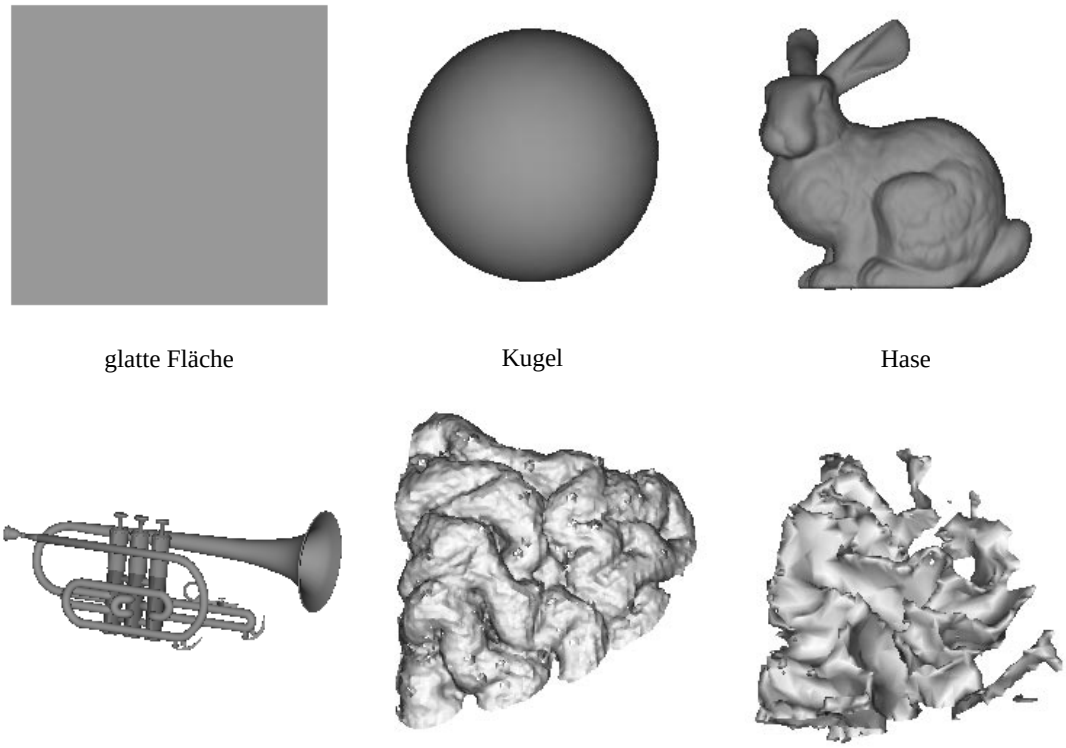


Abbildung 4.9: Verschiedene Testobjekte für die Polygondatenreduktion, Erklärungen im Text.

Flächenabschnitten auch kleinere Unregelmäßigkeiten.

Zunächst wird der Einfluß des vorgegebenen Grenzwertes ϵ untersucht, ohne die Parameter α, β und γ zu berücksichtigen. Das heißt, der Reduktionsvorgang wird ohne Einfluß der Fairneßfunktion durchgeführt. Die Reihenfolge der Reduktionsschritte ist nur abhängig von den Indizes der Primitive in den Ausgangsdatensätzen (Tabelle 4.1).
Bei der "glatten Fläche" wird das Ideal von zwei Dreiecken nicht erreicht, da der Algorithmus ein lokales Minimum erreicht, das keine weiteren Reduktionen zuläßt. Bei den Modellen "Hase", "Hirnoberfläche I" und "Hirnoberfläche II" ist zu erkennen, daß bei einem relativ großen ϵ von 0.05 eine weitere Erhöhung von ϵ keine Erniedrigung der Dreiecksanzahl bewirkt.

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	200	1996	69451	28815	48115	10636
$\epsilon = 0.001$	11	1000	16527	9690	18546	6750
$\epsilon = 0.002$	11	852	12605	8414	15647	6485
$\epsilon = 0.005$	11	656	9546	7304	12404	5940
$\epsilon = 0.01$	11	512	8674	6610	10997	5430
$\epsilon = 0.05$	16	352	8181	6310	10091	5078
$\epsilon = 0.1$	13	294	8185	6096	10098	5080

Tabelle 4.1: Reduktionsraten ohne Einfluß der Fairneßfunktion bei verschiedenem ϵ.

Nun erfolgt eine Berücksichtigung des Kriteriums E. Damit werden die Reduktionsschritte zuerst durchgeführt, die eine geringe geometrische Abweichung hervorrufen. Die Anzahl der Dreiecke sollte gegenüber dem vorigen Test abnehmen (Tabelle 4.2).

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	200	1996	69451	28815	48115	10636
$\epsilon = 0.001$	11	1034	16728	9994	18537	6676
$\epsilon = 0.002$	11	932	12222	8544	14812	6476
$\epsilon = 0.005$	11	660	8967	7254	11742	5929
$\epsilon = 0.01$	11	496	7752	6612	9970	5443
$\epsilon = 0.05$	17	338	7428	6312	9168	4966
$\epsilon = 0.1$	15	300	7402	6310	9168	4952

Tabelle 4.2: Reduktionsraten unter Berücksichtigung der geometrischen Abweichung E .

Bei den künstlich erzeugten Modellen "glatte Fläche", "Kugel" und "Trompete" ist keine Verbesserung der Reduktionsraten gegenüber der zufälligen Reduktion erkennbar. Da bei der vorigen Untersuchung die "Qualität" des entstehenden Polygonnetzes nicht berücksichtigt wurde, kann noch Optimierungspotential ausgeschöpft werden. Bei fest vorgegebener maximaler Abweichung $\epsilon = 0.005$ wird der Einfluß des Kriteriums R untersucht (Tabelle 4.3).
Die Einführung der Dreiecksqualität mit der Gewichtung β ergibt eine Verbesserung der Reduktionsraten gegenüber der alleinigen Berücksichtigung der geometrischen Abweichung. Bei den Testmodellen ergeben sich, vom Sonderfall "glatte Fläche" abgesehen,

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	200	1996	69451	28815	48115	10636
$\alpha = 0, \beta = 1$	2	490	3437	6564	9825	5666
$\beta = 0.1, \alpha = 1$	2	542	3841	6544	9673	5755
$\beta = 0.3, \alpha = 1$	2	540	3250	6442	9682	5761
$\beta = 1, \alpha = 1$	2	508	3230	6430	9598	5710
$\beta = 2, \alpha = 1$	2	490	3313	6496	9605	5672
$\beta = 5, \alpha = 1$	2	496	3294	6472	9691	5675
$\beta = 10, \alpha = 1$	2	496	3408	6538	9780	5672
$\beta = 20, \alpha = 1$	2	468	3387	6508	9863	5699

Tabelle 4.3: Reduktionsraten unter Berücksichtigung der Dreiecksqualität R in Verbindung mit der geometrischen Abweichung in Anzahl $\triangle$. In der ersten Zeile ist $\alpha = 0$ und $\beta = 1$, dort wird der alleinige Einfluß von R sichtbar. ($\epsilon = 0.005$)

Verbesserungen bis zu 60 Prozent. Bei dem Modell glatte Fläche wird das ideale Ergebnis von zwei Dreiecken erreicht.

Als nächstes wird der Einfluß des Parameters γ untersucht. Die erreichbare Reduktion müßte größer sein als bei der zufälligen und der lediglich die geometrische Abweichung benutzenden Funktion. Die vorgegebene maximale Abweichung ist ebenfalls $\epsilon = 0.005$ (Tabelle 4.4).

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	200	1996	69451	28815	48115	10636
$\alpha = 0, \gamma = 1$	25	520	7305	7048	11196	5827
$\gamma = 0.1, \alpha = 1$	22	592	7654	6980	10759	5765
$\gamma = 0.3, \alpha = 1$	22	582	7527	6970	10675	5857
$\gamma = 1, \alpha = 1$	22	610	7007	7006	10831	5823
$\gamma = 2, \alpha = 1$	22	606	6871	7000	10786	5854
$\gamma = 5, \alpha = 1$	22	544	6710	6928	11041	5840
$\gamma = 10, \alpha = 1$	22	528	6906	7106	11123	5836
$\gamma = 20, \alpha = 1$	22	520	6921	7054	11263	5805

Tabelle 4.4: Reduktionsraten unter Berücksichtigung der lokalen Krümmung S in Verbindung mit der geometrischen Abweichung in Anzahl $\triangle$. In der Zeile $\alpha = 0$ ist $\gamma = 1$, dort wird der alleinige Einfluß von S betrachtet. ($\epsilon = 0.005$)

Die erreichten Reduktionsraten sind niedriger als bei alleiniger Verwendung der geometrischen Abweichung, jedoch nicht so gut wie bei der Verwendung der Dreiecksqualität.

Nun werden die Krümmung und die Dreiecksqualität kombiniert, die maximale geometrische Abweichung ist konstant bei $\epsilon = 0.005$. Nachfolgend werden die Ergebnisse für die einzelnen Testmodelle aufgeführt.

Kugel

$\beta =$	0.1	0.3	0.6	1	2	3	4	5	10	20	30
$\gamma = 0.01$	540	532	510	536	498	490	478	494	504	470	502
$\gamma = 0.05$	544	556	514	512	492	498	486	492	496	486	510
$\gamma = 0.1$	554	578	544	498	504	498	482	494	514	490	506
$\gamma = 0.3$	576	556	526	528	510	498	496	510	506	506	494
$\gamma = 0.6$	592	550	516	508	500	486	492	502	486	512	498
$\gamma = 1$	558	558	542	486	516	512	500	508	480	502	510
$\gamma = 2$	556	528	510	508	506	502	516	492	472	482	506
$\gamma = 5$	576	534	544	494	518	518	514	496	514	480	504
$\gamma = 10$	558	548	532	506	520	494	510	526	502	502	498
$\gamma = 20$	542	514	552	528	496	490	506	490	514	500	494

Tabelle 4.5: Dreiecksanzahlen bei der reduzierten Kugel, minimale Werte sind hervorgehoben.

Hase

$\beta =$	0.1	0.3	0.6	1	2	3	4	5	10	20	30
$\gamma = 0.01$	3664	3371	3235	3154	3294	3309	3255	3264	3383	3436	3374
$\gamma = 0.05$	3962	3308	3223	3163	3239	3300	3363	3337	3398	3346	3440
$\gamma = 0.1$	4186	3402	3190	3126	3157	3335	3240	3345	3264	3461	3418
$\gamma = 0.3$	4086	3457	3255	3176	3195	3258	3247	3358	3227	3285	3394
$\gamma = 0.6$	4116	3465	3341	3269	3222	3363	3325	3294	3351	3402	3385
$\gamma = 1$	4200	3667	3491	3279	3386	3245	3342	3397	3279	3356	3393
$\gamma = 2$	4386	3783	3517	3495	3406	3397	3459	3335	3451	3386	3457
$\gamma = 5$	5102	4050	3819	3560	3429	3549	3456	3529	3504	3386	3361
$\gamma = 10$	5569	4711	4076	4026	3680	3561	3520	3513	3560	3441	3466
$\gamma = 20$	6135	5389	4749	4260	3845	3768	3705	3628	3582	3470	3444
$\gamma = 30$	6600	5738	5092	4769	4238	3922	3726	3711	3573	3480	3614

Tabelle 4.6: Dreiecksanzahlen bei dem reduzierten Modell "Hase", minimale Werte sind hervorgehoben.

Trompete

$\beta =$	0.1	0.3	0.6	1	2	3	4	5	10	20	30
$\gamma = 0.01$	6412	6476	6400	6388	6436	6468	6516	6518	6452	6488	6452
$\gamma = 0.05$	6494	6384	6338	6446	6402	6470	6494	6522	6488	6510	6504
$\gamma = 0.1$	6504	6454	6554	6412	6490	6510	6488	6514	6520	6436	6456
$\gamma = 0.3$	6722	6574	6456	6522	6506	6428	6448	6484	6536	6526	6430
$\gamma = 0.6$	6856	6748	6556	6476	6500	6518	6486	6466	6514	6506	6552
$\gamma = 1$	6820	6742	6700	6650	6474	6544	6528	6496	6528	6438	6530
$\gamma = 2$	6822	6818	6888	6788	6710	6590	6516	6588	6546	6456	6446
$\gamma = 5$	7084	6970	6764	6862	6814	6810	6676	6578	6450	6580	6604
$\gamma = 10$	6980	6942	6996	6940	6880	6852	6746	6928	6564	6598	6732
$\gamma = 20$	7048	6954	7060	6912	6842	6906	6912	6890	6732	6648	6552
$\gamma = 30$	7046	7050	6972	7028	7064	6936	6912	6828	6818	6754	6618

Tabelle 4.7: Dreiecksanzahlen bei dem reduzierten Modell "Trompete", minimale Werte sind hervorgehoben.

Hirnoberfläche I

$\beta =$	0.1	0.3	0.6	1	2	3	4	5	10	20	30
$\gamma = 0.01$	9518	9558	9601	9488	9624	9612	9655	9788	9786	9853	9888
$\gamma = 0.05$	9749	9635	9452	9575	9575	9681	9615	9616	9716	9788	9834
$\gamma = 0.1$	9689	9513	9667	9490	9597	9676	9652	9583	9623	9763	9857
$\gamma = 0.3$	10028	9647	9657	9529	9667	9512	9563	9487	9823	9675	9726
$\gamma = 0.6$	10189	9707	9697	9635	9524	9521	9707	9554	9637	9856	9787
$\gamma = 1$	10320	10050	9919	9632	9737	9602	9692	9733	9610	9640	9924
$\gamma = 2$	10633	10295	10096	9989	9799	9885	9641	9597	9560	9619	9732
$\gamma = 5$	10902	10698	10482	10305	9981	10049	10065	9815	9847	9594	9721
$\gamma = 10$	11025	10941	10794	10588	10277	10198	10212	10139	9934	9651	9742
$\gamma = 20$	11029	11017	10859	10794	10653	10444	10225	10321	10084	10019	9831
$\gamma = 30$	11089	11023	10979	10841	10604	10620	10457	10307	10260	10037	9848

Tabelle 4.8: Dreiecksanzahlen bei dem reduzierten Modell "Hirnoberfläche I".

Hirnoberfläche II

$\beta =$	0.1	0.3	0.6	1	2	3	4	5	10	20	30
$\gamma = 0.01$	5734	5774	5740	5713	5691	5687	5696	5704	5704	5709	5703
$\gamma = 0.05$	5731	5778	5713	5762	5682	5687	5686	5690	5717	5712	5706
$\gamma = 0.1$	5755	5720	5719	5737	5675	5705	5707	5712	5702	5712	5702
$\gamma = 0.3$	5758	5744	5742	5732	5743	5695	5693	5709	5725	5722	5691
$\gamma = 0.6$	5780	5794	5799	5776	5752	5730	5714	5718	5701	5731	5711
$\gamma = 1$	5843	5781	5797	5784	5803	5751	5770	5742	5689	5714	5732
$\gamma = 2$	5815	5802	5804	5783	5794	5822	5811	5776	5711	5695	5734
$\gamma = 5$	5814	5823	5789	5777	5753	5799	5817	5828	5787	5744	5713
$\gamma = 10$	5785	5836	5814	5824	5800	5793	5789	5837	5767	5739	5781
$\gamma = 20$	5843	5787	5814	5835	5806	5766	5805	5813	5756	5810	5795
$\gamma = 30$	5804	5832	5812	5796	5810	5785	5795	5788	5733	5789	5781

Tabelle 4.9: Dreiecksanzahlen bei dem reduzierten Modell "Hirnoberfläche II", minimale Werte sind hervorgehoben.

Die Werte in den Tabellen zeigen deutlich, daß der Einfluß der Parameter β und γ bezüglich der erreichbaren Reduktionsraten nur grob geschätzt werden kann. Prinzipiell sind bei der gemeinsamen Verwendung von β und γ aber kleinere Dreiecksanzahlen erreichbar. Günstig erscheinen Werte für β von 1 und γ von 0.1. Eine Übergewichtung von γ führt zu ungünstigeren Reduktionsraten. Die erreichbare Verbesserung der Reduktionsraten liegt im Bereich von unter fünf Prozent gegenüber der alleinigen Verwendung der Dreiecksqualität (R). Kleine Änderungen am Verfahren zur Reduktion, wie z.B. die Reihenfolge der Bearbeitung von Reduktionsschritten mit gleicher Bewertung verursachen auch Veränderungen der Reduktionsrate in dieser Größenordnung.

Die Reduktionsergebnisse hängen stark von den Ausgangsdaten ab. Die Grundlage für die Polygondatenreduktion sind schwach gekrümmte Bereiche im Polygonnetz, die mit weniger Polygonen beschrieben werden können. Mit den Modellen "glatte Fläche" und "Kugel" können die Auswirkungen der Reduktion auf glatte und gekrümmte Flächenabschnitte einzeln betrachtet werden. Bei schwach gekrümmten Flächenabschnitten treten nur geringe geometrische Abweichungen auf. Das Kriterium für die Dreiecksqualität R des Reduktionsverfahrens sorgt in diesen Bereichen dafür, daß keine Dreiecke mit spitzen Winkeln entstehen. Dreiecksstrukturen, bei deren weiterer Vereinfachung überlappende Dreiecke entstehen würden und damit einen Abbruch der Reduktion in diesem Bereich bewirken, treten deshalb weniger häufig auf. Die Datenreduktion in diesen Bereichen ist höher als bei gekrümmten Bereichen. Bei gekrümmten Flächenabschnitten spielt die geometrische Abweichung E eine größere Rolle. Durch Bevorzugung von Reduktionsschritten mit geringer geometrischer Abweichung wird die Reduktion positiv beeinflußt. Eine Glättung der Oberfläche tritt durch Verwendung des Kriteriums S auf.

Die Datenmenge bei dem Modell "Hase" konnte stark verringert werden. Das kann mit der einfachen Topologie und den großen Bereichen mit schwacher Krümmung begründet werden. Zudem wird die Oberflächenstruktur des Objektes innerhalb der durch die geo-

metrische Abweichung beschriebenen Toleranz geglättet. Dabei reichen weniger Polygone zur Darstellung aus. Die Reduktionsrate liegt bei über 95 Prozent (Tabelle 4.6) bzw. über 99 Prozent (Tabelle 4.10). Bei Modellen mit vielfältiger Topologie, wie dem Objekt "Hirnoberfläche I" sind nicht so hohe Reduktionsraten möglich. Zum einen sind die Bereiche mit schwach gekrümmten Flächenabschnitten kleiner, zum anderen gibt es eine Vielzahl an topologischen Besonderheiten, wie Löcher und kleine Erhebungen, für deren Beschreibung mehr Polygone benötigt werden. Die Reduktionsraten liegen bei 80 Prozent (Tabelle 4.8) bzw. 85 Prozent (Tabelle 4.10).

4.8 Visuelle Beurteilung der Reduktionsergebnisse

Abbildung 4.10 zeigt die Unterschiede zwischen einem nicht vereinfachten und vereinfachten Polygonnetzen. Während bei einem geringen zulässigen Fehler ϵ wenige Verfälschungen des Originals auftreten, ist bei größerem ϵ eventuell mit unerwünscht großen Veränderungen auszukommen. Gut zu sehen ist, daß kleinere Details, wie die "Rauhigkeit" der Oberfläche, durch das Reduktionsverfahren verschwinden. Die Topologie bleibt erhalten.

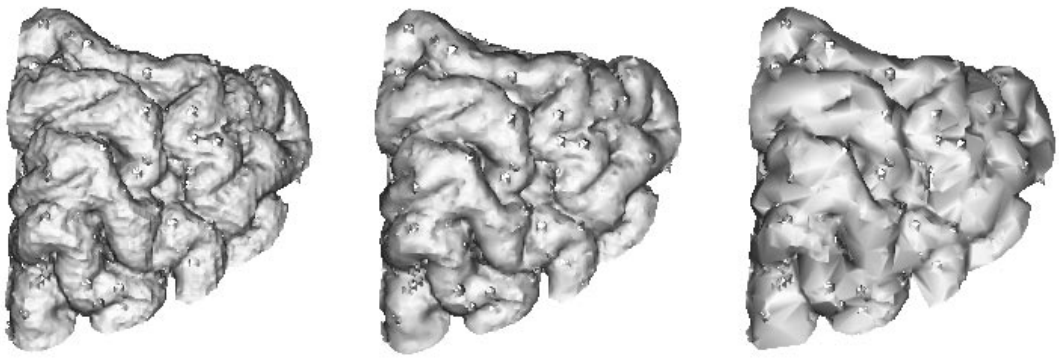


Abbildung 4.10: Vergleich des Erscheinungsbildes vom Original (links 48115 $\triangle$) und vereinfachten Polygonnetzen (mitte 16934 $\triangle$ $\epsilon = 0.001$, rechts 7152 $\triangle$ $\epsilon = 0.05$)

4.9 Veränderung des Flächeninhaltes durch die Reduktion

Erwartet wird eine Abnahme des Flächeninhaltes, da durch die Reduktion Details des Polygonnetzes weniger ausgeprägt sind. Mit den Messungen der Tabellen 4.10 und 4.11 wird nach einer Bestätigung dieser Annahme gesucht.

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	200	1996	69451	28815	48115	10636
$\epsilon = 0.001$	2	936	12501	9354	16934	6639
$\epsilon = 0.002$	2	768	7325	7798	13084	6375
$\epsilon = 0.005$	2	520	3203	6538	9726	5754
$\epsilon = 0.01$	2	330	1823	5806	7940	5210
$\epsilon = 0.05$	2	122	660	5452	7152	4855
$\epsilon = 0.1$	2	74	552	5326	7180	4855

Tabelle 4.10: Reduktionsraten bei variablem ϵ und $\alpha = 1$, $\beta = 1$ und $\gamma = 0.4$ in Anzahl der $\triangle$.

	glatte Fläche	Kugel	Hase	Trompete	Hirn I	Hirn II
Original $\triangle$	100	100	100	100	100	100
$\epsilon = 0.001$	100	99,6	99,6	98,6	99,5	99,4
$\epsilon = 0.002$	100	99,4	99,3	98,2	99,4	99,6
$\epsilon = 0.005$	100	98,9	98,7	98,1	99,4	99,4
$\epsilon = 0.01$	100	97,9	98,1	97,8	99,3	99,4
$\epsilon = 0.05$	100	93,6	96,2	98,3	99,3	99,8
$\epsilon = 0.1$	100	90,5	93,7	98,8	98,9	99,8

Tabelle 4.11: Vergleich der Flächeninhalte von reduzierten Polygonnetzen im Vergleich zum Original auf der Basis der Polygonnetze aus Tabelle 4.10 in Prozent.

Die Erwartungen stimmen mit den gemessenen Ergebnissen überein. Bei dem Modell "Kugel" tritt die größte Abweichung auf. Das ist darauf zurückzuführen, daß bei Entfernung von Dreiecken der Kugeldarstellung die Krümmung zwangsläufig größer wird, und das reduzierte Netz innerhalb der Originalkugel schrumpft. Das Modell "Hase" hat an mehreren Stellen auch die Eigenschaften von kugelförmigen Oberflächen und daher eine größere Abweichung. Insgesamt ist der Flächenunterschied zwischen Original und reduziertem Netz jedoch relativ klein und liegt unterhalb von zehn Prozent. Bei den

Hirnoberflächen ist die Abweichung sehr gering und liegt unter zwei Prozent.

Kapitel 5

Zusammenfassung

Neurometrische Untersuchungen von Hirnoberflächen spielen im Max-Planck-Institut für neuropsychologische Forschung eine große Rolle. Mit dem im Rahmen der Diplomarbeit entwickelten Editor lassen sich solche Untersuchungen durchführen, was das Bearbeiten und Vermessen von Polygonnetzen bedeutet.

Die Darstellung von 3D-Daten auf einem 2D-Anzeigegerät (Bildschirm) kann nur mit Hilfe von Schnitten oder Projektionen erfolgen. Das hat zur Folge, daß oft nicht alle Details eines Objektes zu sehen sind, so daß es sinnvoll ist (*i*) die Szene von verschiedenen Standpunkten aus zu betrachten, (*ii*) die freie Auswahl eines verdeckungsfreien Betrachtungspunkts zu ermöglichen oder (*iii*) den interessierenden Ausschnitt aus geringer Entfernung zu betrachten. Die 3D-Operationen Verschieben und Drehen haben dreidimensionale Vektoren als Eingabeparameter. Mit einer Maus können in einer interaktiven Umgebung nur zweidimensionale Eingabevektoren erzeugt werden, so daß die Operationen des Verschiebens und Drehens als Komposition von 2D-Operationen dargestellt werden müssen.

Mit dem Editor ist eine Bearbeitung und Vermessung hochkomplexer Polygonnetze möglich. Bei sehr großen Polygonnetzen (Anzahl der Polygone größer als 100000) ist eine Interaktion in nutzerfreundlichen Zeiträumen nicht mehr möglich. Durch Weiterentwicklung der Rechentechnik wird sich diese Grenze weiter nach oben verschieben. Allerdings wird

auch die Weiterentwicklung von bildgebenden Verfahren zu höheren Bildauflösungen und damit zu größeren Datenmengen führen. Auswege aus dieser Situation sind das *Resolution Modeling* und die Polygondatenreduktion.

Durch die Polygondatenreduktion kann die Datenmenge der aus Volumendaten extrahierten Polygonnetze verringert werden. Für dieses Problem existieren eine Vielzahl verschiedener Algorithmen und Verfahren. Die Entscheidung für ein inkrementelles Verfahren, wurde durch die Notwendigkeit der Erhaltung der Topologie eines Polygonnetzes beeinflußt. Die Polygondatenreduktion kann man als komplexes Optimierungsproblem auffassen. Schwierigkeiten bereitet dabei die Tatsache, daß ein Reduktionsschritt abhängig von vorangegangenen Reduktionsschritten ist. Wegen der Vielzahl an Ableitungen von einem Originalnetz zu einem reduzierten Netz kann die Suche nach optimalen Lösungen nur eingeschränkt erfolgen. Durch Berücksichtigung von Heuristiken, wie die bei einem Reduktionsschritt entstehenden Dreieckswinkel oder der lokalen Krümmung kann das Reduktionsverfahren positiv beeinflußt werden. Zur Steuerung des Reduktionsverfahrens wird daher eine Fairneßfunktion verwendet, die die eben genannten Parameter berücksichtigt. Die Reduktionsergebnisse hängen stark von den Ausgangsdaten ab. Objekte mit einfacher Topologie und großen schwach gekrümmten Bereichen lassen sich besonders gut vereinfachen. Bei durch 3D-Scanner oder Flächenextraktion aus Volumendaten erzeugten Datensätzen sind die Reduktionsraten hoch und liegen bei den untersuchten Objekten im Bereich von 65 bis 99 Prozent. Obwohl die Reduktionsraten zum Teil sehr groß sind, repräsentieren die reduzierten Polygonnetze die Originale recht gut.

Anhang A

Beispiele

Eine Aufgabe des Editors ist die Herauslösung von kleineren Segmenten aus einem größeren Polygonnetz (Abbildung A.1). Die Möglichkeit Trennlinien auf komplexen Oberflächen

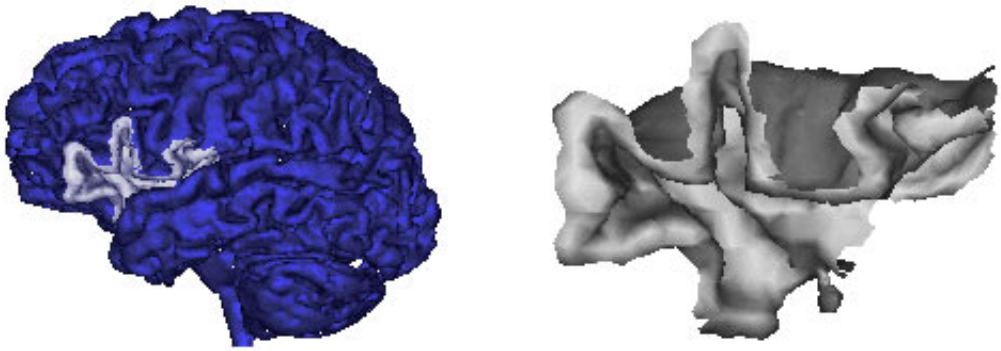


Abbildung A.1: Das rechte Bild wurde aus dem großen linken Bild extrahiert

anzugeben, um Stücke des Polygonnetzes herauszuschneiden funktioniert bei sehr komplexen Datensätzen nur für kleinere Bereiche. Wie schon in Kapitel 3.2.3.3 erwähnt genügt es nicht nur einen geschlossenen Pfad auf einem Polygonnetz anzugeben,um ein richtiges Ausschneiden eines Teilstücks zu bewirken. Bei den im Institut erstellten Datensätzen existieren eine Vielzahl von "Querverbindungen" innerhalb eines Polygonnetzes. Wählt

man den Hilfsfad so, daß noch Querverbindungen zwischen dem auszuschneidenden Stück und dem Rest des Polygonnetzes bestehen, kann keine topologische Trennung erfolgen. Kennt man die Querverbindungen, kann man diese einzeln trennen und erhält dann ein separates Polygonnetz.

Eine andere Möglichkeit kleinere Teilstücke aus einem größerem Polygonnetz zu erhalten, besteht darin, die nicht zur interessanten Region gehörenden Teile zu löschen.

Abbildung A.2 zeigt die Anwendung des Trennvorgangs mit einer Hilfsebene. Gut zu sehen ist, wie das Polygonnetz mit einem glatten Schnitt geteilt wurde.

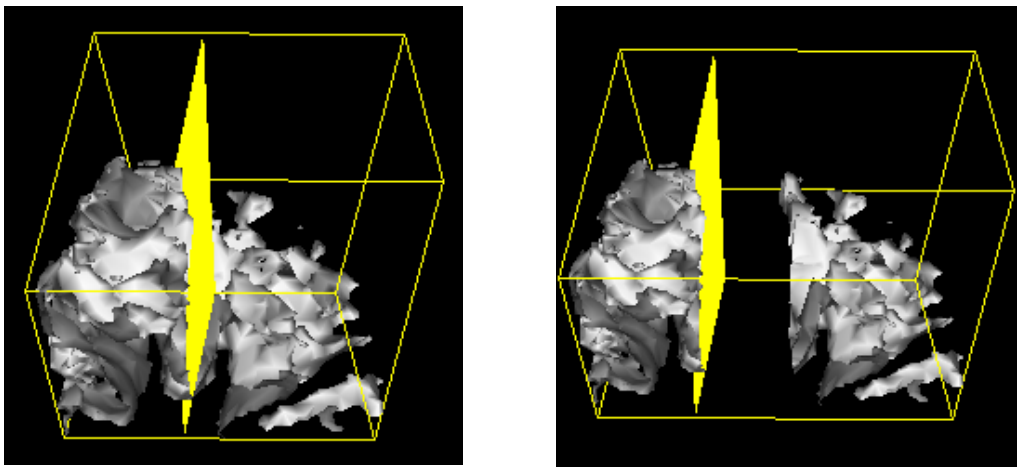


Abbildung A.2: Trennung mit einer Trennebene.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[CCMS97] A. Ciampalini, P. Cignoni, C. Montani, R. Scopigno. *Multiresolution Decimation based on Global Error.* Technical Report, Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Via S. Maria 36 - 56126 Pisa, ITALY, 1997.

[CKS98] Swen Campagna, Leif Kobbelt, Hans-Peter Seidel. *Efficient Decimation of Complex Triangle Meshes.* Technical Report 3/98, Computer Graphics Group, University Erlangen-Nürnberg, Germany, 1998.

[CMRS98] P. Cignoni, C. Montani, C. Rocchini, R. Scopigno. "Zeta: A Resolution Modeling System". *Graphical Models and Image Processing 60*, S. 305–329, 1998.

[CRS97] P. Cignoni, C. Rocchini, R. Scopigno. "Metro: measuring error on simplified surfaces". Technical Note, Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Via S. Maria 36 - 56126 Pisa, ITALY, 1997.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, William Wright. "Simplification envelopes". *SIGGRAPH 96 Conference Proceedings*, S. 119–128, 1996.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, Werner Stuetzle. *Multiresolution Analysis of Arbitrary Meshes.* Technical Report 95-01-02, University of Washington, Seattle, WA and Micro-

soft Research, Redmond, WA and Alias Research, Toronto, Ontario, Canada, 1995.

[ESV96] Francise Evans, Steven Skiena, Amitabh Varshney. *Optimizing Triangle Strips for fast Rendering* Technical report, Department of Computer Science, State University of New York at Stony Brook, Stony Brook NY 11794-4400, 1996.

[FDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley, Reading, 2. Auflage, 1996.

[GKN90] Walter Gellert, Herbert Kästner, Siegfried Neuber. *Lexikon der Mathematik.* VEB Bibliographisches Institut Leipzig, Leipzig, 5. Auflage, 1990.

[GR94] Andre Guéziec, Hummel Robert. "The wrapper algorithm: Surface extraction and simplification". In *Proceedings of the IEEE Workshop on Biomedical Image Analysis*, S. 204–213, 1994. IEEE Computer Society Press, Los Alamitos.

[Hec94] Paul S. Heckbert. *Graphics Gems IV.* Academic Press, London, 1994.

[Hop96] Hugues Hoppe. "Progressive meshes". In *SIGGRAPH 96 Conference Proceedings*, S. 99–108, 1996.

[Imm97] Christian Immler. *Das grosse Buch 3D Studio Max.* Data Becker GmbH & Co. KG, Düsseldorf, 1. Auflage, 1997.

[KCS98] Leif Kobbelt, Swen Campagna, Hans-Peter Seidel. "A general Framework for Mesh Decimation". *Graphics Interface '98*, S. 43–50, 1998.

[MES99] "Mesa OpenGL-like API". *http://www.opengl.org/Documentation/Implementation/Mesa.html*, 1999.

[PJS92] Heinz-Otto Peitgen, Hartmut Jürgens, Dietmar Saupe. *Bausteine des Chaos — Fraktale.* Springer Verlag und Klett-Cotta-Verlag, Heidelberg und Stuttgart, 1. Auflage, 1992.

[Sie96] Hans-Jürgen Siegert. *Robotik: Programmierung intelligenter Roboter.* Springer Verlag, Heidelberg, 1. Auflage, 1996.

[SL96] Marc Soucy, Denis Laurendeau. "Multiresolution surface modeling based on hierarchical triangulation". *Computer Vision and Image Understanding*, 63(1):1–14, 1996.

[Str96] Karin Strohmer. *Effiziente Berechnung und Darstellung von Freiformkurven und Freiformflächen in der Computergraphik.* Diplomarbeit, Technische Universität München, 1996.

[Str98] Bjarne Stroustrup. *Die C++ Programmiersprache.* Addison-Wesley, Reading, 3. Auflage, 1998.

[SZL92] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen. "Decimation of Triangle Meshes". In *SIGGRAPH 92 Conference Proceedings*, S. 65–70, 1992.

[WND97] Mason Woo, Jackie Neider, Tom Davis. *OpenGL Programming Guide.* Addison-Wesley, Reading, 2. Auflage, 1997.

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, _____ _____

Thomas Nowotka