Synthetisierung und Visualisierung von Erdfragmenten und Mineralien

Diplomarbeit

von

Eldar Sultanow

711289

UNIVERSITÄT POTSDAM

Institut für Informatik

Betreuende Dozenten	Prof. Dr. Ullrich Kortenkamp, Prof. Dr. Christoph Kreitz
Studiengang	Informatik
Professur	Mathematik, Theoretische Informatik
Abgabetermin	19. Dezember 2005

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Thema:

Synthetisierung und Visualisierung von Erdfragmenten und Mineralien

selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Danksagung

Ich möchte an dieser Stelle all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen der vorliegenden Diplomarbeit beigetragen haben.

Beiden Professoren, Herrn Prof. Kortenkamp und Herrn Prof. Kreitz danke ich für die wissenschaftliche Betreuung, die Ermöglichung und Förderung dieser Diplomarbeit.

Für die immer freundliche wissenschaftliche Unterstützung und die damit einhergehende unermüdliche Hilfsbereitschaft bedanke ich mich bei Frau Dr. Richter. Ihre zahlreichen wissenschaftlichen Ratschläge haben stets zur Verbesserung der Diplomarbeit beigetragen.

Es zeigten viele neben den hier genannten Personen ein großes Interesse an der vorliegenden Arbeit. Auch Ihnen sei an dieser Stelle recht herzlich gedankt.

Kurzfassung

In der vorliegenden Arbeit werden Mineralien mit dem Computer nachgebildet. Dazu wird eine Java Klassenbibliothek entwickelt, mit der man dreidimensionale Modelle für Mineralien erzeugen und visualisieren kann. Das Erzeugen der 3D Modelle ist das Hauptproblem und wesentlicher Bestandteil dieser Arbeit. Es existiert derzeit keine Klassenbibliothek für das Erstellen von 3D Modellen für Mineralien, die vergleichbar mit den in dieser Arbeit vorgestellten Modellen sind. Für die Visualisierung der 3D Modelle wurden unter anderem bereits existierende Klassenbibliotheken angepasst und integriert.

Die im Rahmen dieser Arbeit entwickelte Klassenbibliothek ist ein Java Werkzeug („Java Minerals 3D“ oder kurz JAMIN), welches unter Verwendung von verschiedenen Algorithmen und geometrischen Verfahren Kristalle und Edelsteine nachbildet. Neben den Kristallen und Edelsteinen lassen sich mit JAMIN auch größere Erdfragmente wie Felsen oder Felsblöcke nachbilden. Die Nachbildungen können in einer dreidimensionalen Ansicht rotieren, was die Betrachtung von jeder beliebigen Seite ermöglicht.

Die Abbildung 1-1 zeigt ein beispielhaftes 3D Modell des Felsenbogens *Delicate Arch* aus dem *Arches National Park* (Utah). In dieser Arbeit werden Felsen, Steine und Kristalle entsprechend wie sie in der Natur vorkommen oder auf Fotos dargestellt sind nachgebildet. Wie die Nachbildung unter Verwendung welcher Methoden erfolgt, wird im Laufe der Ausführungen näher erläutert.

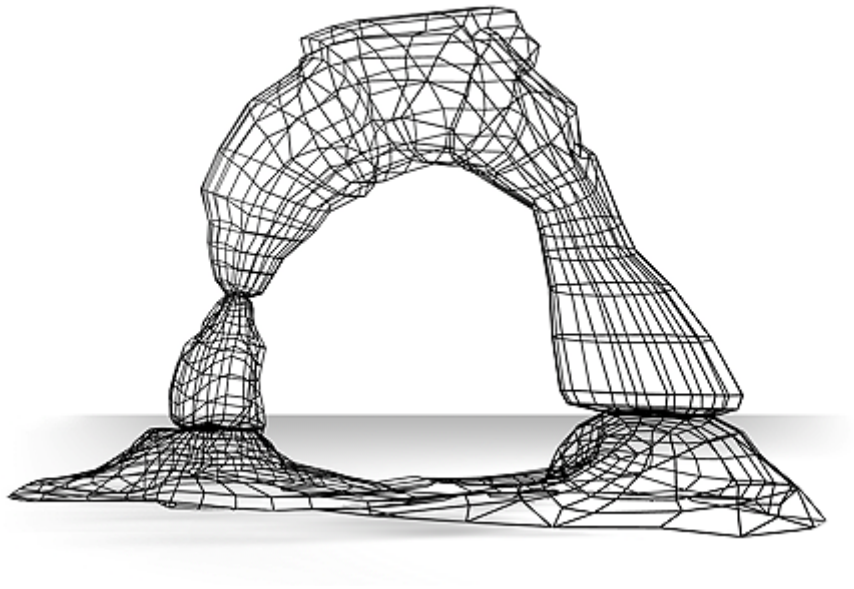


Abbildung 1-1: 3D Modell des Felsenbogens *Delicate Arch*

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

Seite

Tabellenverzeichnis

Seite

Anlagenverzeichnis

Seite

1 Einleitung

Viele Webseiten, insbesondere jene zu wissenschaftlichen Themen, integrieren zunehmend eine interaktive Visualisierung ihrer Inhalte. Ein Beispiel dafür ist die Mathematikwebseite von Dr. Eric W. Weisstein [Weis05], die im Bereich der Geometrie und Analysis mit Hilfe von *JavaView* verschiedene Polyeder und Funktionsgraphen visualisiert. *JavaView* ist eine Java Klassenbibliothek für Numerik und interaktive Visualisierung dreidimensionaler Geometrien. Studenten, Lehrer und Wissenschaftler können *JavaView* dafür verwenden, E-Learning Plattformen oder wissenschaftliche Veröffentlichungen im Internet um Experimente zu erweitern [PKPR02].

Java ist besonders relevant für Anwendungen, die in Webseiten eingebettet sind, da Java für den Einsatz im Internet entwickelt wurde [MBFM99]. Webbrowser ermöglichen dem Anwender ohne Installationsaufwand, nur durch Aufruf einer Webseite so genannte *Applets*[1] auszuführen. Die Ausführung von *Applets* erfolgt unter Vorgabe strenger Sicherheitsvorkehrungen: *Applets* haben keinerlei lesenden oder schreibenden Zugriff auf den lokalen Rechner des Anwenders [MBFM99].

Bisher veröffentlichen Webseiten, sowohl von Freizeit-Mineralogen, Steinsammlern, Juwelieren als auch von Wissenschaftlern, die Eigenschaften, Fundorte und statischen Darstellungen (Skizzen, Fotos) der Kristalle und Edelsteine. Für räumlich komplexere Formen von Mineralien ist es zweckmäßig, diese interaktiv zu visualisieren. Das im Rahmen dieser Arbeit entwickelte Werkzeug soll die bisherigen Möglichkeiten der Präsentation von Mineralien im Internet erweitern: Es sollen auf Webseiten 3D Modelle für Mineralien präsentiert werden, die sich mit Hilfe der Maus in Rotation versetzten lassen.

Das Ziel der vorliegenden Arbeit war die Entwicklung einer Java Klassenbibliothek, um eine animierte graphische Darstellung von Kristallen, Edelsteinen und anderen Mineralien im Internet zu ermöglichen. Im Gegensatz zu beispielsweise 3D Computerspielen besteht die Herausforderung bei der Entwicklung von Anwendungen, die in Webseiten eingebettet sind darin, dass keine direkten Graphikhardwarezugriffe (etwa über Renderingsysteme wie *OpenGL* oder *DirectX*) möglich sind. Diese Einschränkung bedeutet, dass sämtliche 3D Graphik-Funktionen softwarebasiert umgesetzt werden müssen. Dabei kommt es vor allem auf die Auswahl geeigneter Algorithmen der geometrischen Datenverarbeitung und auf die Auswahl vorhandener Klassenbibliotheken für die Visualisierung an.

[1] Ein Applet ist ein Java-Programm, das sich in Webseiten einbetten lässt und im Browserkontext abläuft. Applets können somit über das Internet geladen werden.

Das Werkzeug JAMIN berücksichtigt bei der graphischen Darstellung besonders

- *die verschiedenen Kristallformen*
- *die Transparenzeigenschaften* (die Materialien, aus denen Kristalle und Edelsteine bestehen)
- *die Farbgebung und Textur*

der Mineralien. Die 3D Modelle können interaktiv beliebig „gedreht“ und „deformiert“ werden. Bisher wurden die Berücksichtigung von Schatten und das Anwenden von Schliffen noch nicht realisiert.

Die Arbeit ist in drei Hauptteile gegliedert: die Erstellung der Kristallformen, die Erstellung der Steinformen und die Entwicklung des Werkzeugs JAMIN. In Kapitel 3 werden Möglichkeiten für die Berechnung der Kristallformen und in Kapitel 4 Methoden für die Modellierung von Steinformen beschrieben. Das Werkzeug JAMIN sowie die dafür verwendeten Klassenbibliotheken *JavaView* und *Idx3d* werden in Kapitel 5 vorgestellt.

Mit JAMIN soll es möglich sein, Polygonmodelle für Kristallformen zu erzeugen. Die Kristallformen entsprechen konvexen Polyedern. So soll JAMIN unter Angabe eines Polyedernamens das Polygonmodell des entsprechenden Polyedes erstellen. Dies wurde nicht für alle, aber für einige bekannte Polyeder umgesetzt, deren Formen der vieler Kristalle entspricht. In Kapitel 3 werden Polygonmodelle für konvexe Polyeder, auch das für den Rhombendodekaeder vorgestellt.

JAMIN soll auch das Polygonmodell eines Polyeders berechnen können, wenn man die Ebenen angibt, welche die Begrenzungsflächen des Polyeders einbetten. Dazu werden so genannte *Miller'sche Indizes* (eine häufige Notation für Kristallformen) verwendet. In Kapitel 3 werden die Berechnung des Polygonmodells für einen Kristall unter Vorgabe der *Miller'schen Indizes* und ein entsprechender Algorithmus beschrieben. Dieser Algorithmus wurde für das Werkzeug JAMIN implementiert.

Neben den Kristallformen soll JAMIN auch Steinformen erstellen können. Hierfür erzeugt das Werkzeug JAMIN Polygonmodelle für Steine, Steinscheiben oder Felsen. In Kapitel 4 werden Methoden für die Erstellung solcher Modelle beschrieben. Es wird dort gezeigt, wie unter Verwendung der konvexen Hülle von Zufallspunkten die Polygonmodelle für Steinschnitte und -scheiben erstellt werden. Neben der konvexen sind Subdivisionsflächen, Stream Bubbles und Interpolierte Implizite Flächen Methoden, die sich jeweils für die Erstellung von Steinformen verwenden lassen. So wird in Kapitel 4 auch gezeigt, wie mit Hilfe der Subdivisionsflächen Polygonmodelle für größere Steine und mit Hilfe der Stream Bubbles Polygonmodelle für Felsen erstellt werden.

Das Werkzeug JAMIN kann Polygonmodelle für implizite Flächen in einem vorgegeben Intervall erstellen. Implizite Flächen und eine spezielle Teilklasse, die *Blobby Surfaces* werden in Kapitel 4 beschrieben. Dort wird eine Approximation für die Gaußfunktion vorgestellt, welche auch im Rahmen dieser Arbeit entwickelt wurde.

Die in Kapitel 4 beschriebenen Methoden wurden für das Werkzeug JAMIN implementiert. In Kapitel 5 wird das Werkzeug JAMIN näher vorgestellt. JAMIN berechnet aus einer vorgegebenen Menge von *Miller'schen Indizes* die entsprechende Kristallform. Es gibt Webseiten, welche zu verschiedenen Kristallen die *Miller'schen Indizes* oder die polygonale Darstellung als *Eckenliste* (kartesische Koordinaten) angeben. Diese Webseiten sind für die Arbeit mit dem Werkzeug JAMIN hilfreich, weil sich die dort angegebenen kristallgeometrischen Daten als Eingabe verwenden lassen. In Kapitel 5 werden einige solcher Webseiten genannt.

Es gibt Programme, die Polygonmodelle für Kristallformen erzeugen, graphisch darstellen und in eine Datei abspeichern. Auch für diese Programme können die auf den Webseiten veröffentlichten Daten als Eingabe verwendet werden. In Kapitel 5 werden einige dieser Programme vorgestellt. JAMIN soll das Format der Polygonmodelle, welche diese Programme in den Dateien abspeichern, lesen können. In Kapitel 5 sind dazu einige Austauschformate für Polygonmodelle beschrieben.

Weiter werden in Kapitel 5 Webseiten genannt, welche Bilder veröffentlichen, die sich für eine Texturierung der Polygonmodelle durch JAMIN eignen. Das sind häufig Fotos von Kristallen oder Texturen von Steinoberflächen. Aus den dort genannten Webseiten wurden viele Bilder entnommen und als Textur für die Polygonmodelle in JAMIN verwendet.

JAMIN greift auf zwei vorhandene Systeme *JavaView* und *Idx3d*, zurück. Beide sind Java Klassenbibliotheken und wurden in das Werkzeug JAMIN integriert. Im letzten Teil des Kapitels 5 werden die Klassenbibliotheken JavaView, Idx3d und JAMIN beschrieben. Insbesondere wird gezeigt, wie die Klassebibliothek JavaView verwendet wird und was für ihre Integration zu berücksichtigen ist. Abschließend wird in Kapitel 5 zu der Beschreibung des Werkzeugs JAMIN ein Verwendungsbeispiel gegeben.

2 Grundlagen

In diesem Kapitel sollen einige grundlegende Begriffe aus der Mineralogie, der geometrischen Datenverarbeitung sowie der Mathematik eingeführt werden, welche für die Implementierung des Werkzeugs JAMIN verwendet wurden.

2.1 Grundbegriffe aus der Mineralogie

Gegenstand der Mineralogie ist unter anderem das Studium vom Aufbau der Mineralien sowie ihrer Entstehungsbedingungen. Die Mineralien sind aus mehreren chemischen Elementen aufgebaut oder bestehen lediglich aus einem einzigen Element, wie zum Beispiel Kupfer, Gold und Schwefel [Natu04]. In Abhängigkeit von seiner chemischen Zusammensetzung weist jedes Mineral eine typische Form auf. Häufig kommen Polyeder vor, das heißt räumliche Strukturen, die durch ebene Flächen begrenzt werden. Man spricht in diesem Fall von *Kristallen*. In [Wagn17] ist folgende Definition zu finden:

„Ein Kristall ist ein chemisch gleichartig zusammengesetzter von ebenen Flächen begrenzter Körper, der durch natürliches Wachstum der Substanz gebildet worden ist und dessen physikalische Eigenschaften mit der äußeren Form in gesetzmäßigen Beziehungen stehen."

Mineralien, die keine Kristallstruktur aufweisen werden als *amorph* bezeichnet (sie sind oft kugel- oder nierenförmig). Beispiele für amorphe Mineralien sind Bernstein, Pech und Opal [Wagn17].

Kristalle lassen sich nach der speziellen Lage der einzelnen Kristallflächen zueinander, die sich aus der Symmetrie des Kristalls ergibt, klassifizieren. Diese Klassifizierung ist für weitere Betrachtungen insbesondere für die Visualisierung bedeutsam. Die verschiedenen Kristallformen lassen sich in sieben Kristallsysteme unterteilen. Jedes dieser sieben Kristallsysteme zeichnet sich durch eine bestimmte Kombination von folgenden Symmetrieelementen aus [Natu04]:

- *Symmetrieebene*: Durch eine Symmetrieebene wird der Kristall in zwei spiegelbildlich gleiche Hälften geteilt.

- *Symmetrieachse*: Es gibt eine Gerade mit der Eigenschaft, dass zu jedem Punkt P auf der Oberfläche ein Punkt P' existiert, so dass die Strecke $\overline{PP'}$ senkrecht zu der Geraden verläuft und von ihr genau in der Mitte geteilt wird.

- *Symmetriezentrum*: Ein solches Zentrum ist vorhanden, wenn es zu jeder einzelnen Kristallfläche eine parallele, spiegelverkehrte Gegenfläche gibt.

Abbildung 2-1 zeigt Beispiele zu jedem der sieben Kristallsysteme. Die Abbildung illustriert die äußere Form der Kristalle entsprechend ihrer Zuordnung zu dem Kristallsystem. In dieser Abbildung sind die sieben Kristallsysteme namentlich genannt. Diese Namen wurden für die Implementierung des Werkzeugs JAMIN verwendet. Einige diese Formen wurden so auch für das Werkzeug JAMIN implementiert.

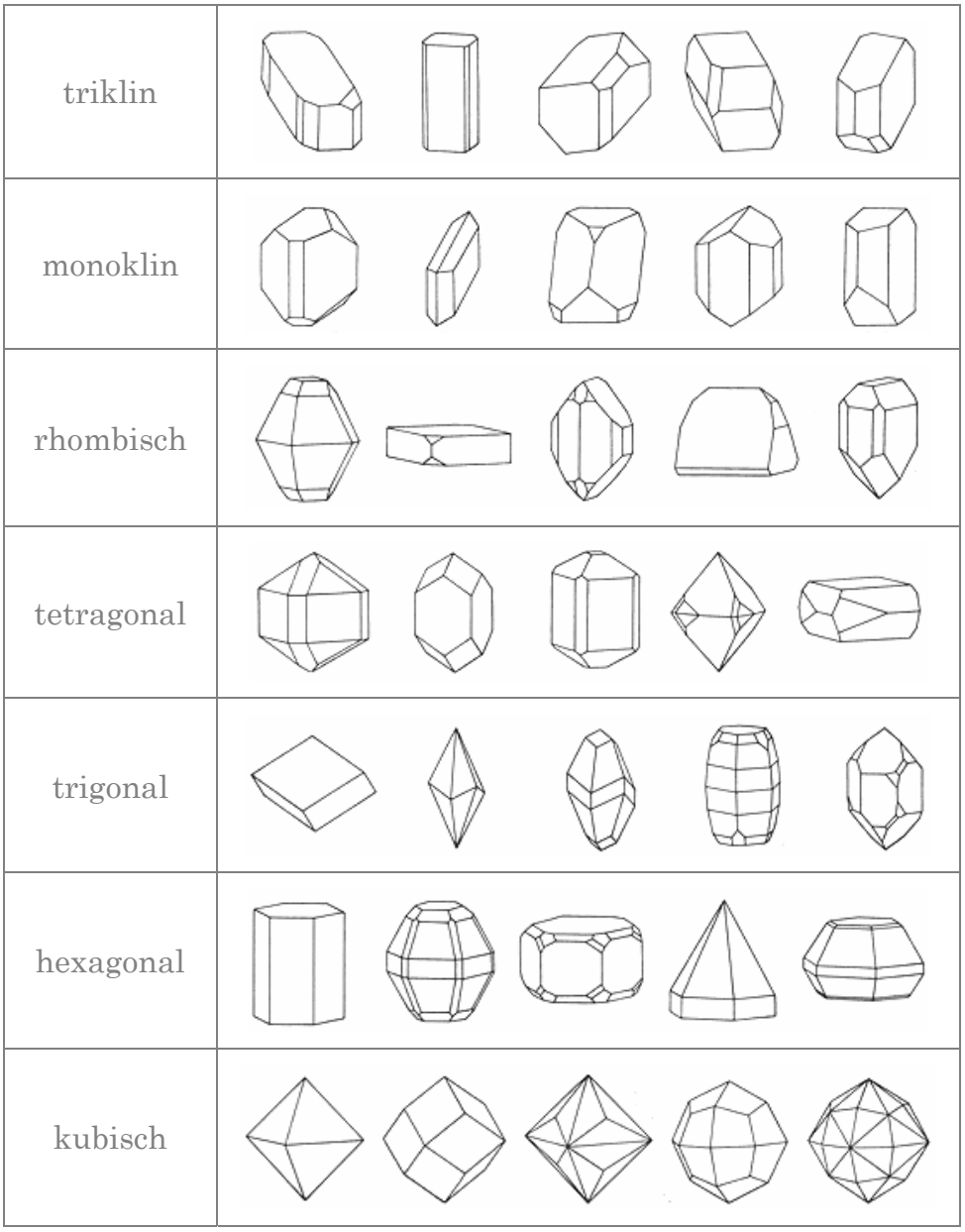


Abbildung 2-1: Die sieben Kristallsysteme [2]

[2] [Natu04], S. 12

2.2 Einige Grundlagen der geometrischen Datenverarbeitung

Zur visuellen Darstellung von Oberflächen, Begrenzungsflächen und Verbindungslinien zwischen gegebenen Punkten greift JAMIN auf bekannte Verfahren der geometrischen Datenverarbeitung zurück. Einige Methoden, die sich bei der Darstellung von Steinformen verwenden lassen sind in der folgenden Tabelle aufgeführt und werden in Kapitel 4 näher beschrieben. Diese Methoden wurden für die Implementierung des Werkzeugs JAMIN verwendet.

Methode	Kurze Beschreibung
Random Convex Hulls	Es wird die konvexe Hülle einer Menge von zufällig erzeugten Raumpunkten berechnet.
Subdivisionsflächen	Ausgehend von einem polygonalen Netz wird durch vorgegebene Schemata ein verfeinertes polygonales Netz berechnet.
Stream Hulls, Stream Bubbles	Spezielle NURBS Flächen, begrenzt durch eine Kontrollpunkt Box, bilden verformbare geschlossene Oberflächen.
Interpolierte Implizite Flächen	Durch Interpolieren von impliziten Flächen werden sphärische Oberflächen erzeugt.

Tabelle 2-1: Methoden, die sich für die Erstellung von Steinformen verwenden lassen

An dieser Stelle sollen die Grundbegriffe *Subdivisionsfläche*, *NURBS* und *implizite Fläche* erläutert werden, da sie zur Terminologie für den verstehenden Umgang mit der vorliegenden Arbeit gehören.

Eine *Subdivisionsfläche* ist ein verfeinertes polygonales Netz. Ausgehend von einer vorgegebenen Punktmenge wird durch Hinzufügen neuer Punkte (die mit den im Netz vorhandenen Punkten verbunden werden) ein verfeinertes Netz erzeugt. Die Position der neuen Punkte wird als Funktion von der Position der alten Punkte berechnet. Auf diese Weise lässt sich eine visuelle Glätte des verfeinerten Netzes erreichen.

NURBS steht für *Non Uniform Rational Basis Splines* und bezeichnet Freiformkurven, -flächen und -volumenkörper, die durch Interpolation berechnet werden. Sie werden über ein Netz von Kontrollpunkten und assoziierten Gewichtungsfunktionen definiert. Diese Gewichtungsfunktionen sind dabei parametrisierte Polynome. NURBS beschreiben die 3D Objekte nicht explizit über Raumkoordinaten, sondern nähern sie stückweise in einer benutzerdefinierbaren Genauigkeit an. NURBS haben sich zum Industriestandard für das Modellieren von 3D Objekten entwickelt. So werden sie im digitalen Grafik-, Produkt- und Maschinendesign wie zum Beispiel für die Modellierung von Bauteilen verwendet.

Implizite Flächen sind Nullstellengebilde im dreidimensionalen Raum. Sie werden entsprechend einer besonderen mathematischen Form definiert:

$$\{p \in \mathbb{R}^3 \mid f(p) = 0\}$$

Setzt man einen Punkt p in die implizite Funktion f ein und das Ergebnis ist Null, so liegt dieser Punkt p auf der impliziten Oberfläche [Sult04].

2.3 Darstellung und Datenstruktur

Es gibt grundsätzlich zwei Darstellungsformen, in denen Geometrien[3] vorliegen können: Sie können entweder in einer analytischen oder in einer diskreten Darstellung vorliegen. Die Geometrie in einer diskreten Darstellung heißt 3D Modell.

Die analytische Darstellung ist eine Beschreibung der Geometrie durch parametrisierte mathematische Funktionen. Als Beispiel sei eine Kugel genannt, die durch ihren Mittelpunkt *M* und ihren Radius *r* und die Funktion $X : \mathbb{R}^2 \to \mathbb{R}^3$ mit:

$$X(u,v) = \overrightarrow{OM} + r \cdot \begin{pmatrix} \cos u \cos v \\ \sin u \cos v \\ \sin v \end{pmatrix}, \ (u,v) \in [0, 2\pi] \times [-\pi/2, \pi/2]$$

beschrieben wird.

Die diskrete Darstellung besteht zum Beispiel aus

- einer Aufzählung aller Kanten oder
- einer Menge von ebenen Begrenzungsflächen. Hier spricht man von einer polygonalen Darstellung der Geometrie. Die Geometrie wird in diesem Fall auch Polygonmodell genannt.

Ein 3D Modell muss zwei Typen von Informationen über die Geometrie enthalten: die geometrische Information (zum Beispiel in Form von Punktkoordinaten) sowie die topologische Information, welche beschreibt, wie die geometrischen Entitäten in Beziehung zueinander stehen (zum Beispiel, welche zwei Punkte miteinander verbunden sind). Für die interne Darstellung dieser Informationen gibt es verschiedene so genannte topologische Datenstrukturen. Entsprechend der unterschiedlichen Datenstrukturen gibt es auch verschiedene Austauschformate, in denen das einzelne 3D Modell oder sogar mehrere 3D Modelle jeweils in einer Datei gespeichert werden. Einige Austauschformate werden in Kapitel 5.2 näher vorgestellt. Die Struktur dieser Austauschformate beinhaltet die entsprechende topologische Datenstruktur. Nachfolgend seien Beispiele für solche topologischen Datenstrukturen genannt.

- *Eckenliste*: Es werden *n* durchnummerierte Eckpunkte zu einer Liste zusammengefasst und sämtliche Polygone als Liste der Eckenindizes dargestellt. Die Reihenfolge der Eckenindizes ist für alle Polygone konsistent, das heißt für alle Polygone werden entweder die Eckenindizes im Uhrzeigersinn oder im Gegenuhrzeigersinn angegeben.
- *Kantenliste*: Pro Kante werden die Indizes der Endpunkte angegeben sowie der Index der Polygone, zu der die Kante gehört. Die Polygone sind als Liste der Kantenindizes dargestellt.
- *Halbkantenliste*: Halbkanten sind gerichtet. Eine Kante setzt sich aus zwei Halbkanten zusammen. Eine der zwei zu einer Kante gehörenden Halbkanten ist zu der anderen

[3] Mit Geometrie meint man hier Gebilde wie zum Beispiel Polyeder oder Oberflächen, die durch mathematische Funktionen beschrieben sind.

Halbkante entgegengerichtet. Zu einer Halbkante werden der Index des Anfangs- und des Endpunktes sowie der Index der auf der linken Seite der Halbkante befindlichen Fläche angegeben. Die Polygone sind als Liste der Halbkantenindizes dargestellt.

2.4 Die Systeme JavaView und Idx3d

Bei der Entwicklung des Werkzeugs JAMIN wurde sich auf die Klassenbibliotheken *JavaView* und *Idx3d* gestützt. *JavaView* und *Idx3d* sind Java Klassenbibliotheken für die Verarbeitung und die graphische Darstellung von dreidimensionalen Geometrien. *JavaView* stellt eine Vielzahl von Funktionen für die Manipulation der 3D Modelle, wie beispielsweise die Verfeinerung der Oberfläche durch Hinzufügen von Eckpunkten und Polygonen, zur Verfügung. Die in *JavaView* verwendete topologische Datenstruktur und das *JavaView* Austauschformat werden für das Werkzeug JAMIN verwendet. *Idx3d* ermöglicht die Anpassung der graphischen Darstellung, zum Beispiel die Veränderung der Transparenz oder des Reflektionskoeffizienten und wird für die Visualisierung der Geometrien eingesetzt. Beide Klassenbibliotheken, *JavaView* und *Idx3d*, werden im Kapitel 5.4 noch genauer beschrieben. Dort werden auch einige Klassen, die für die Implementierung des Werkzeugs JAMIN verwendet wurden näher vorgestellt.

3 Erstellung der Kristallformen

In diesem Kapitel werden Möglichkeiten für die Berechnung der Kristallformen beschrieben. Kristallformen entsprechen konvexen Polyedern. So wird für die Erstellung der Kristallform eine polygonale Darstellung von konvexen Polyedern erzeugt. Um eine graphische Darstellung der Polyeder (Kristalle) zu erzeugen, müssen alle darzustellenden Eckpunkte und Begrenzungsflächen berechnet werden. Ausgangspunkt ist der Name des Polyeders (Kristalls), der wie in Abbildung 2-1 einem Kristallsystem zugeordnet ist. Von bekannten Polyedern (Würfel, Oktaeder, …) sind die Eckpunkte und Begrenzungsflächen in der Literatur wie zum Beispiel in [Baie00] angegeben. Ausgangspunkt für die Berechnung der Eckpunkte und der Begrenzungsflächen kann auch die Angabe der so genannten *Netzebenen* durch *Miller'sche Indizes* sein. Es gibt Webseiten, welche zu verschiedenen Kristallen die *Miller'schen Indizes* angeben. Einige von ihnen werden in Kapitel 5 genannt.

Für die Implementierung des Werkzeugs JAMIN ist die interne Darstellung der geometrischen und topologischen Information von Polyedern als Eckenliste zweckmäßig, da viele 3D Softwarebibliotheken die topologische Datenstruktur Eckenliste verwenden. Dieser Sachverhalt trifft insbesondere für die im Rahmen der Arbeit verwendeten Klassenbibliotheken *JavaView* und *Idx3d* zu.

3.1 Polyeder als Kristallformen

In [Baie00] wird ein konvexes Polyeder *P* in einer polygonalen Darstellung als Eckenliste angegeben. Ein Polyeder ist demnach gegeben als *n*-Tupel[4]

$$V = ((x_0, y_0, z_0), \ldots, (x_{n-1}, y_{n-1}, z_{n-1}))$$

ihrer kartesischen Eckpunktskoordinatentripel $v_k = (x_k, y_k, z_k)$. Es gilt:

$$P = \mathcal{K}(v) = \sum_{k=0}^{n-1} a_k v_k \text{ mit } a_k \geq 0 \text{ und } \sum_{k=0}^{n-1} a_k = 1$$

[4] Die Bezeichnung der Tupel *V* und *F* ist auf die englische Übersetzung für Eckpunkt *(Vertex)* und Fläche *(Face)* zurückzuführen.

P ist die konvexe Hülle dieser Eckpunkte. Die Begrenzungsflächen F_i der Hülle des Polyeders können durch die m-Tupel der Indizes der Eckpunkte $v_{i_0},\ldots,v_{i_{m-1}}$ in der Form $F_i=(i_0,\ldots i_{m-1})$ angegeben werden. Dabei beschreibt das Tupel $F=(F_0,\ldots,F_{p-1})$ die Oberfläche des Polyeders. Die Ebenen $E_0,E_1,\ldots,E_{p-1}$, welche die Oberflächenpolygone einbetten, sind die *Netzebenen* des Kristalls.

Nachfolgend werden die Polygonmodelle einiger Kristalle vorgestellt, die für das Werkzeug JAMIN verwendet wurden. Die Eckenliste für ein Rhombendodekaeder wie in Abbildung 3-1 (links) kann wie folgt angegeben werden:

$$V=((0,0,-2),(1,-1,-1),(-1,1,1),(0,-2,0),(-1,-1,-1),(0,2,0),(-2,0,0),\\(0,0,2),(-1,-1,1),(1,1,1),(1,1,-1),(1,-1,1),(2,0,0),(-1,1,-1))$$

$$F=((2,6,8,7),(13,5,10,0),(12,11,3,1),(3,8,6,4),(2,7,9,5),(0,1,3,4),(11,7,8,3),\\(10,12,1,0),(5,9,12,10),(0,4,6,13),(7,11,12,9),(6,2,5,13))$$

Die Kristalle des Andradits, wie alle übrigen Granatmineralien, kommen in Form von Rhombendodekaedern vor [Natu04]. Die Eckenliste eines regulären nicht normierten Ikosaeders, dargestellt in Abbildung 3-1 (rechts), mit der Festlegung

$$(a_1,a_2,a_3,a_4,a_5,a_6)=\left(\frac{1}{4}\left(\sqrt{5}-1\right),\frac{1}{4}\sqrt{2}\sqrt{5+\sqrt{5}},\frac{1}{8}\sqrt{6}\sqrt{5-\sqrt{5}},\frac{1}{4}\left(\sqrt{5}+1\right),\frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}},\frac{1}{8}\sqrt{94-6\sqrt{5}}\right)$$

ist gegeben durch

$$V=((a_1,-a_2,a_3),(1,0,a_3),(a_1,a_2,a_3),(-a_4,a_5,a_3),(-a_4,-a_5,a_3),(a_4,-a_5,-a_3),\\(a_4,a_5,-a_3),(-a_1,a_2,-a_3),(-1,0,-a_3),(-a_1,-a_2,-a_3),(0,0,a_6),(0,0,-a_6))$$

$$F=((10,0,1),(10,1,2),(10,2,3),(10,3,4),(10,4,0),(11,9,8),(11,8,7),(11,7,6),(11,6,5),(11,5,9)\\(0,9,5),(5,1,0),(1,5,6),(6,2,1),(2,6,7),(7,3,2),(3,7,8),(8,4,3),(4,8,9),(9,0,4))$$

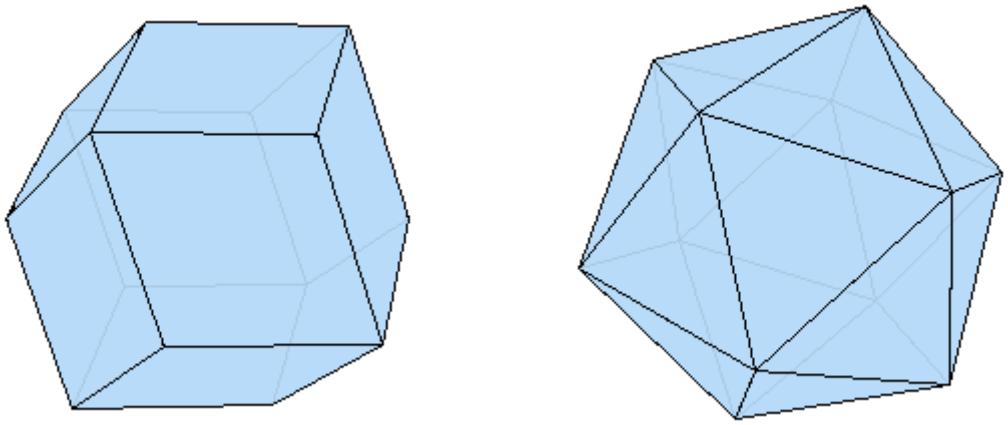


Abbildung 3-1: Rhombendodekaeder (links) und Ikosaeder (rechts)

Im Grundlagenkapitel wurde bereits erwähnt, dass bei der Eckenliste die Reihenfolge der Eckenindizes für alle Polygone konsistent sein muss. Aus diesem Grund ist bei der Erstellung von Polygonmodellen darauf zu achten, dass entweder für alle Polygone die Indizes der Eckpunkte im Uhrzeigersinn oder im Gegenuhrzeigersinn angegeben werden. Viele 3D Softwarebibliotheken, so auch *Idx3d*, setzen dies aus Effizienzgründen für die Visualisierung der Polygonmodelle voraus. Die graphische Darstellung des Rhombendodekaeders und des Ikosaeders in Abbildung 3-1 wurden jeweils durch JAMIN (im Speziellen mit Hilfe der Klassenbibliothek *JavaView*) erzeugt.

3.2 Beschreibung der Kristallformen durch Netzebenen

Eine andere Beschreibung der Kristallformen erfolgt durch die Angabe der Netzebenen, was zu einer analytischen Darstellung führt. Ausgangspunkt hierbei ist die vorgegebene Menge von Netzebenen $\{E_1,\ldots,E_n\}$ eines Kristalls. Der einer Kristallform entsprechende Polyeder *P* enthält in seinem Inneren den Koordinatenursprung $O=(0,0,0)$. Falls ein Symmetriezentrum existiert, entspricht der Koordinatenursprung dem Symmetriezentrum des Polyeders. Die Netzebenen werden durch die so genannten *Miller'schen Indizes* gekennzeichnet. Für Kristallformen sind die Miller'schen Indizes eine sehr häufige Notation, die in der Literatur über Mineralogie verwendet wird [Hild02]. William H. Miller (1801-1880) entwickelte dieses Modell zur Beschreibung von Kristallformen und damit eine kompakte sowie einheitliche Darstellung, die eine Ordnung in die Vielzahl der zuvor existierenden unterschiedlichen Beschreibungsformen hineinbrachte [Hild02]. Grundsätzlich beschreibt Miller eine Kristallform durch eine Menge von Tripel ganzer Zahlen (h,k,l). Eine Ausnahme bilden die Kristalle des hexagonalen Kristallsystems, die er durch eine Menge von Quadrupeln ganzer Zahlen (h,k,i,l) beschreibt. Im Folgenden wird am Beispiel des kubischen Gitters aufgezeigt, wie die Miller'schen Indizes ermittelt werden können. Die Gittereinheitszelle bei einem kubischen Gitter ist der Würfel. Er wird durch sechs Flächen begrenzt, die jeweils in einer der sechs Netzebenen des Würfels liegen. In Abbildung 3-2 sind die Miller'schen Indizes zu diesen Netzebenen angegeben. Vom Inneren des Würfels gehen drei Achsen aus, deren gemeinsamer Schnittpunkt, das Achsenkreuz, zu allen Würfelflächen denselben Abstand hat. Dieser Abstand entspricht dann genau einer Längeneinheit. Einer Netzebene *E* wird zunächst das Indextripel

$$(h',k',l'),\ h',k',l' \in \mathbb{Z} \cup \{\infty\}$$

auf folgende Art zugeordnet: Schneidet *E* die durch e_1 beschriebene Achse auf der negativen Seite des Ursprungs, so ist der Index h' negativ. Schneidet sie die Achse auf der positiven Seite des Ursprungs, so ist h' positiv. Wenn *E* die durch e_1 beschriebene Achse im Unendlichen schneidet, so wird der Index h' mit ∞ bezeichnet. Dieselbe Zuordnung gilt analog für e_2 mit dem Index k' und für e_3 mit dem Index l'.

Sei *d* das kleinste gemeinsame Vielfache der Indizes die kleiner ∞ sind. Dann erhält man durch Multiplikation von *d* mit den Kehrwerten $1/h'$, $1/k'$ und $1/l'$ neue Zahlen $h,k,l \in \mathbb{Z}$. Dabei wird festgelegt, dass $1/\infty = 0/1$ ist. Die Zahlen $h,k,l \in \mathbb{Z}$ sind die gesuchten Miller'schen Indizes:

$$h = d/h',\ k = d/k',\ l = d/l'$$

Die Miller'schen Indizes werden in runden Klammern ohne Kommas angegeben. Negative Indizes werden nicht mit einem Minuszeichen vor dem Index sondern mit einem Überstrich über dem Index dargestellt, so schreibt man zum Beispiel $(\bar{1}01)$ anstelle von (-101).

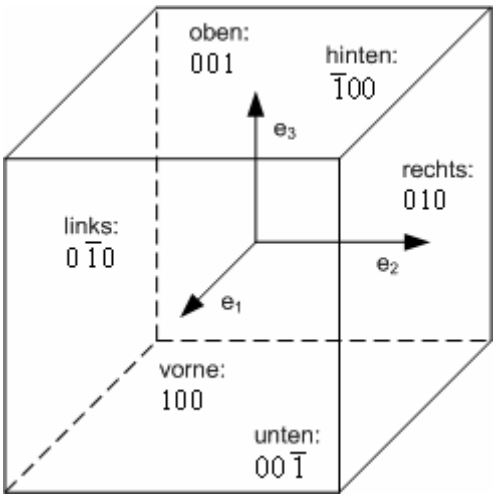


Abbildung 3-2: Miller'sche Indizes der Würfelflächen

Unten ist ein zweites Beispiel für die Miller'sche Indizierung beim kubischen Gitter aufgeführt: Es werden die Miller'schen Indizes des in Abbildung 3-3 dargestellten Quaders angegeben. Die vordere Ebene schneidet die durch e_1 beschriebene Achse um zwei Längeneinheiten vom Ursprung entfernt auf der positiven Seite des Ursprungs, somit ist der Index $h' = 2$. Diese Ebene schneidet die durch e_2 und e_3 beschriebenen Achsen im Unendlichen. Für die vordere Ebene ist demzufolge $k' = l' = \infty$ und das Tripel $(h', k', l') = (2, \infty, \infty)$. Das kleinste gemeinsame Vielfache d der Indizes h', k', l' die kleiner ∞ sind ist 2. Man erhält durch Multiplikation von d mit den Kehrwerten $1/h'$, $1/k'$, $1/l'$ schließlich die Miller'schen Indizes der vorderen Ebene (100). Für die restlichen Ebenen gilt die Tabelle 3-1.

Ebene	(h', k', l')	$(1/h', 1/k', 1/l')$	(hkl)
rechts	$(\infty, 1, \infty)$	$(0, 1, 0)$	(020)
hinten	$(-2, \infty, \infty)$	$(-1/2, 0, 0)$	$(\bar{1}00)$
links	$(\infty, -1, \infty)$	$(0, -1, 0)$	$(0\bar{2}0)$
oben	$(\infty, \infty, 1)$	$(0, 0, 1)$	(002)
unten	$(\infty, \infty, -1)$	$(0, 0, -1)$	$(00\bar{2})$

Tabelle 3-1: Miller'sche Indizierung beim Quader

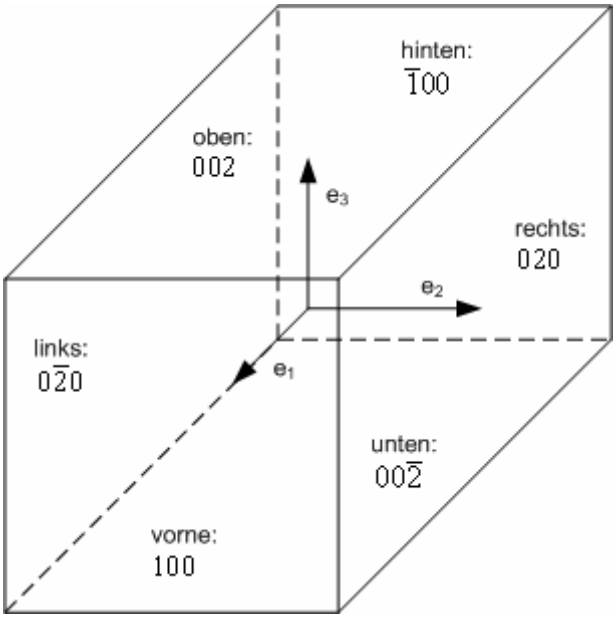


Abbildung 3-3: Miller'sche Indizes der Quaderflächen

Die Miller'schen Indizes ermöglichen eine eindeutige Zuordnung von der vorgegebenen Netzebenenmenge $\{E_1,\ldots,E_n\}$ zu einer Menge von Tripel ganzer Zahlen. Mit der Ausnahme im hexagonalen Gitter können die Miller'schen Indizes als Koordinaten einer Basis für den dreidimensionalen reellen Raum aufgefasst werden [Hild02]. Die Vektoren e_1, e_2, e_3 sind Einheitsvektoren, die Basis ist definiert als die Menge $\{e_1,e_2,e_3\}$.

Zunächst eine kurze Erläuterung zu der Menge *symmetrisch äquivalenter Netzebenen*. Für feste $h,k,l,c\in\mathbb{Z}$ werden die Punkte einer Netzebene *E* beschrieben durch die Gleichung

$$(h,k,l)\cdot\left(x\vec{e}_1+y\vec{e}_2+z\vec{e}_3\right)=c$$

Jedes Tripel (*hkl*) spezifiziert eine bestimmte Ebene in der Gittereinheitszelle. Sei $\Pi=\{\pi_1,\pi_2,\ldots\}$ die Menge aller Permutationen der Elemente *h*, *k*, *l* und sei *P* die Menge der entsprechenden Permutationsmatrizen:

$$P=\left\{\left(p_{ij}^{k}\right)_{i,j=1,2,3}\,\middle|\,k=1,2,\ldots,|\Pi|\right\},\quad p_{ij}^{k}=\begin{cases}1 & j=\pi_k(i)\\ 0 & sonst\end{cases}$$

Weiter sei *S* die Menge der Matrizen

$$S=\left\{\left(s_{ij}^{abc}\right)_{i,j=1,2,3}\,\middle|\,a,b,c\in\{0,1\}\right\},\quad s_{ij}^{abc}=\begin{pmatrix}(-1)^a & 0 & 0\\ 0 & (-1)^b & 0\\ 0 & 0 & (-1)^c\end{pmatrix}$$

Es ist $\{hkl\}$ mit

$$\{hkl\} = \left\{(h,k,l) \middle|\ (h_0,k_0,l_0)\cdot s \cdot p,\ \ p \in P, s \in S\right\}$$

die Menge der symmetrisch äquivalenten Netzebenen zu (hkl). Einer Kristallbegrenzungsfläche wird der Index der Netzebene zugeordnet, die diese Fläche einbettet. Die Kristallbegrenzungsflächen, welche in die Ebenen der Menge $\{hkl\}$ eingebettet sind, ergeben die Menge der symmetrisch äquivalenten Flächen. In Abbildung 3-3 sind die symmetrisch äquivalenten Flächen des Quarzes gleichfarbig gekennzeichnet.

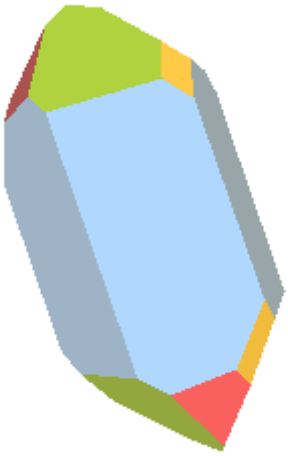


Abbildung 3-4: Symmetrisch äquivalente Flächen des Quarzes

In einfachen Fällen ist die Beschreibung von Kristallformen durch die Menge $\{hkl\}$ ausreichend. Es gibt aber auch Kristallformen, die durch mehrere solcher Mengen beschrieben sind, beziehungsweise beschrieben werden müssen. Die in diesem Fall entstehende Kristallform stelle man sich als den *Durchschnitt* der Kristallformen vor, die jeweils durch eine einzelne Menge $\{hkl\}$ definiert sind. Dieser „Durchschnitt" liefert je nach gewählter Ausdehnung der einzelnen Kristallform eine neue Kristallform. Abbildung 3-4 illustriert die neu entstandene Kristallform als Resultat einer *Durchschnittsbildung* von Würfel und Oktaeder. Der Würfel hat dabei die Kantenlänge 1 und der Oktaeder die Kantenlänge $\sqrt{2}$. Diese in Abbildung 3-4 dargestellte Schnittfigur wird auch als Kuboktaeder bezeichnet. Der Würfel wird durch die Netzebenenmenge

$$\{100\} = \left\{(100),(010),(001),\ \ (\bar{1}00),(0\bar{1}0),(00\bar{1})\right\}$$

beschrieben, der Oktaeder seinerseits durch die Menge

$$\{111\} = \left\{(111),(\bar{1}\,\bar{1}\,\bar{1}),\ \ (\bar{1}11),(1\,\bar{1}1),(11\,\bar{1}),\ \ (\bar{1}\,\bar{1}1),(\bar{1}1\,\bar{1}),(1\,\bar{1}\,\bar{1})\right\}$$

In der Abbildung 3-4 (links) ist der Würfel dargestellt, der bei „Durchschnittbildung" mit dem Oktaeder eine neue Kristallform liefert, die in der Abbildung 3-4 (rechts) dargestellt ist. Diese neue Kristallform wird beschrieben durch die Vereinigungsmenge $\{100\} \cup \{111\}$ der Netzebenenmenge des Würfels und der Netzebenenmenge des Oktaeders. Die entstehenden Kristallformen hängen von den Größenverhältnissen zueinander ab. Viele Programme, die Kristallformen

wie zum Beispiel den Kuboktaeder auf oben beschriebene Weise berechnen, berücksichtigen die Größenverhältnisse, indem sie als Eingabe

- eine Menge, deren Elemente Paare sind, die jeweils aus Miller'schen Indizes (hkl) und einem Abstand d der durch (hkl) beschriebene Netzebene zum Symmetriezentrum bestehen oder
- eine Menge, deren Elemente Paare sind, die jeweils aus einer Ebenenmenge $\{hkl\}$ und einem Abstand d aller durch $\{hkl\}$ beschriebenen Ebenen zum Symmetriezentrum bestehen

zulassen.

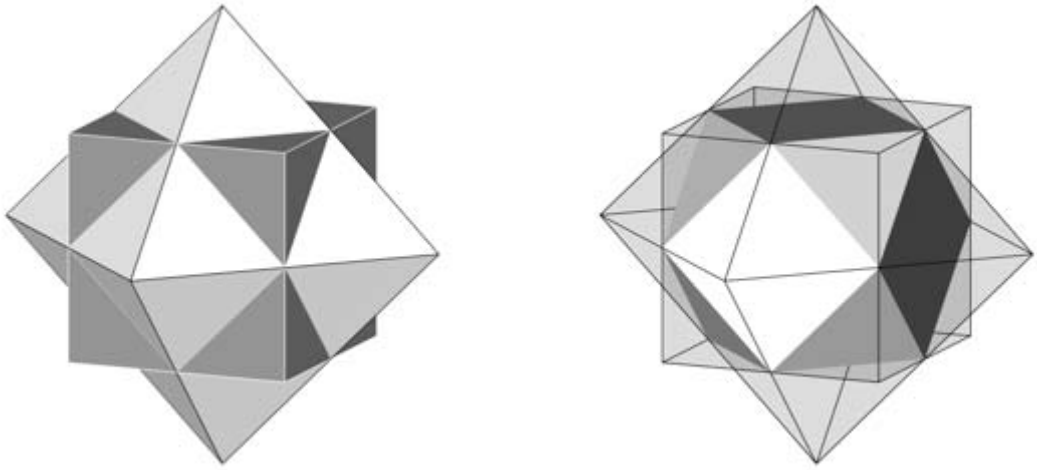


Abbildung 3-5: „Durchschnittbildung" von Würfel und Oktaeder

3.2.1 Miller'sche Indizierung im hexagonalen Gitter

Bei der Indizierung eines hexagonalen Gitters wird ein zusätzlicher Vektor hinzugefügt, wobei nun die Vektoren e_1, e_2, e_3 in einer Ebene liegen und nicht mehr linear unabhängig sind. Es gilt:

$$e_3 = -(e_1 + e_2)$$

Wie in Abbildung 3-4 dargestellt, steht der Vektor e_4 senkrecht auf der von e_1 und e_2 aufgespannten Ebene. Die Indizierung eines hexagonalen Gitters wird mit der Vierer-Indizierung (h,k,i,l) erleichtert, da sich für die Indizes h, k, i, l ganze Zahlen verwenden lassen.

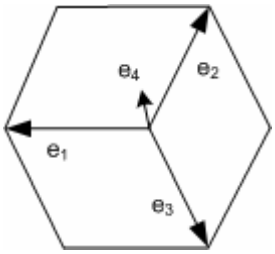


Abbildung 3-6: Miller'sche Indizierung im hexagonalen Gitter

3.3 Algorithmus zur Berechnung der Kristallformen

Das im Rahmen dieser Arbeit entwickelte Werkzeug JAMIN soll ausgehend von der Polyederdarstellung das Polygonmodell der entsprechenden Kristallform berechnen können. Im Folgenden wird ein entsprechender Algorithmus vorgestellt, der auch für das Werkzeug JAMIN implementiert wurde.

Bei der Berechnung von Kristallformen unterscheidet man grundsätzlich zwei Fälle, die auftreten können:

- Die zu visualisierende Kristallform ist durch eine endliche Menge einzelner Tripel - der Millerschen' Indizes - beschrieben: $\{(h_0k_0l_0),(h_1k_1l_1),\ldots,(h_nk_nl_n)\}$. In diesem Fall müssen die den Kristall räumlich begrenzenden Ecken und Polygone berechnet werden. Hierfür werden alle Eckpunkte ermittelt, die sich jeweils als Schnitt von drei, durch die Indizes $(h_ak_al_a)$, $(h_bk_bl_b)$, $(h_ck_cl_c)$ definierten Ebenen für verschiedene $a,b,c \in \{0,1,\ldots,n\}$ ergeben.

- Die Kristallform ist nicht durch die einzelnen Tripel beschrieben, sondern durch eine oder mehrere Mengen von symmetrisch äquivalenten Netzebenen $\{hkl\}$. In diesem Fall müssen zunächst die Miller'schen Indizes aller zu (hkl) symmetrisch äquivalenten Netzebenen berechnet werden. Anschließend ermittelt man wie im obigen Fall beschrieben die Eckpunkte als Schnitt dieser Ebenen.

Als Beispiel für den zuletzt genannten Fall ergeben die zu (110) symmetrisch äquivalenten Netzebenen genau die Ebenenmenge, durch welche ein Rhombendodekaeder beschrieben wird:

$$\{110\} = \{(110),(1\bar{1}0),(\bar{1}10),(\bar{1}\bar{1}0),\ (101),(10\bar{1}),(\bar{1}01),(\bar{1}0\bar{1}),\ (011),(01\bar{1}),(0\bar{1}1),(0\bar{1}\bar{1})\}$$

Drei Permutationen mit jeweils vier Vorzeichenkombinationen ergeben die 12 Rhombendodekaederflächen, beziehungsweise die 12 Netzebenen, welche die Begrenzungsflächen des Rhombendodekaeders einbetten. Von diesen Ebenen müssen die Ebenengleichungen ermittelt werden. Aufbauend auf den Ausführungen in Kapitel 3.2 soll im Folgenden die geometrische Bedeutung der Miller'schen Indizes gezeigt werden. Eine Netzebene mit den Miller'schen Indizes (hkl) hat die Normalform:

$$hx + ky + lz = d \text{ beziehungsweise } hx/d + ky/d + lz/d = 1$$

Dabei ist (h,k,l) ein Normalenvektor der Ebene, wie auch jeder Vektor $(h/d\,,k/d\,,l/d)$ mit beliebigem $d \neq 0$. Setzt man

$$d = h' \cdot h = k' \cdot k = l' \cdot l = kgV\left(h',k',l'\right),$$

so folgt für die Ebenengleichung

$$hx/kgV\left(h',k',l'\right) + ky/kgV\left(h',k',l'\right) + lz/kgV\left(h',k',l'\right) = x/h' + y/k' + z/l' = 1$$

Dies ist die Achsenabschnittsform der Netzebene. Damit verläuft die Netzebene (hkl) durch die drei Spurpunkte $H\left(h',0,0\right)$, $K\left(0,k',0\right)$, $L\left(0,0,l'\right)$.

Dieser geometrische Zusammenhang erlaubt es einen Algorithmus anzugeben, der mit Hilfe von Miller'schen Indizes die Ecken und Polygone berechnet, welche den Kristall räumlich begrenzen. Der im Folgenden beschriebene Algorithmus wurde für das Werkzeug JAMIN implementiert. Er lässt sich in drei wesentliche Schritte teilen.

3.3.1 Die Berechnung der Schnittpunkte von den Netzebenen

Im ersten Schritt, bei der Berechnung der Schnittpunkte von den Netzebenen, geht man von einer vorgegebenen Menge von Paaren aus:

$$\left\{\left(\left(h_1k_1l_1\right),d_1\right),\left(\left(h_2k_2l_2\right),d_2\right),\ldots\right\}$$

Drei sich schneidende Netzebenen ergeben einen möglichen Eckpunkt (*Kandidaten*) der Kristallform. Ausgehend von drei der oben genannten Paare lässt sich das untenstehende lineare Gleichungssystem aufstellen. Jede der drei Gleichungen dieses linearen Gleichungssystems ist die Normalform der Ebenengleichung einer Netzebene.

$$\begin{aligned} h_a x + k_a y + l_a z &= d_a \\ h_b x + k_b y + l_b z &= d_b \\ h_c x + k_c y + l_c z &= d_c \end{aligned}$$

Die Zahlen $a,b,c \in \mathbb{N}$, $a \neq b \neq c$ sind die Indizes der Netzebenen. Das Lösen des Gleichungssystems liefert den Schnittpunkt der drei Netzebenen. Es stellt sich nun die Frage, wie viele Schnittpunkte es insgesamt gibt, wenn sich *m* Netzebenen im Raum schneiden. Die Anzahl dieser Schnittpunkte ist die obere Grenze für die Eckpunktzahl der Kristallform. Betrachtet wird dazu die Partialsummenfolge einer arithmetischen Reihe. Sei (a_n) die arithmetische Reihe, die für alle natürlichen Zahlen *n* beschrieben ist durch:

$$a_n = 1 + 2 + 3 + \ldots + n = \frac{n^2 + n}{2}, \quad \left(a_n\right) = \left(1,3,6,10,15,\ldots\right)$$

Dann wird die Partialsumme über (a_n) gebildet als

$$b_n = \sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \frac{i^2 + i}{2} = \frac{1}{2}\sum_{i=1}^{n} i^2 + \frac{1}{2}\sum_{i=1}^{n} i = \frac{2n^3 + 3n^2 + n}{12} + \frac{n^2 + n}{4} = \frac{n^3 + 3n^2 + 2n}{6}$$

Die ersten Glieder ergeben sich zu

$$(b_n) = (1, 4, 10, 20, 35, \ldots)$$

Gegeben seien *m* Ebenen im dreidimensionalen Raum und keine zwei Ebenen seien parallel zueinander. Dann beschreibt b_{m-2} die Anzahl der Schnittpunkte von je drei der *m* Ebenen. Die Anzahl der Schnittpunkte von je drei aus *m* Ebenen ist gegeben durch den Binomialkoeffizienten

$$\binom{m}{3}$$

Formt man die Partialsumme b_n geeignet um, so erhält man:

$$b_n = \frac{n^3 + 3n^2 + 2n}{6} = \frac{(n+2)(n+1)n}{6} = \frac{(n+2)(n+1)n \cdot (n-1)!}{6 \cdot (n-1)!} = \frac{(n+2)!}{3!(n-1)!} = \frac{(n+2)!}{3!(n+2-3)!} = \binom{n+2}{3}$$

Die Partialsumme b_{m-2} entspricht also der Anzahl der Schnittpunkte von je drei aus *m* Ebenen. Diese Anzahl b_{m-2} ist eine obere Schranke für die Anzahl der möglichen Ecken der Kristallform, weil auch Ecken, die durch Schnitte von mehr als drei Ebenen entstehen, schon als Schnitt von drei Ebenen eindeutig bestimmt sind.

3.3.2 Eliminieren der Schnittpunkte, die nicht Ecken der Kristallform sind

Im vorigen Kapitel wurde gezeigt, wie man *Kandidaten* für Eckpunkte der Kristallform ermittelt. Diese *Kandidaten* ergeben sich durch den Schnitt von jeweils drei Netzebenen. Da nicht alle Schnittpunkte den Ecken der Kristallform entsprechen, muss festgestellt werden, welche der Schnittpunkte keine Ecken der Kristallform sind:

Eine Netzebene im dreidimensionalen Raum definiert zwei Halbräume. Für die folgende Beschreibung wird der Halbraum, welcher den Koordinatenursprung enthält als *Innenseite* und der andere Halbraum als *Außenseite* der entsprechenden Netzebene bezeichnet. Für sämtliche *Kandidaten* ermittelt man, ob er jeweils Element der *Innenseite* aller vorgegebenen Netzebenen ist. Trifft dies für einen *Kandidaten* nicht zu, so ist dieser nicht Eckpunkt der Kristallform.

3.3.3 Berechnung der Begrenzungspolygone

Wenn die Eckpunkte der Kristallform ermittelt wurden, werden im letzten Schritt die Begrenzungspolygone berechnet. Hierfür werden für jede Netzebene alle Eckpunkte ermittelt, die in der Netzebene liegen. Diese Eckpunkte sind Ecken eines Begrenzungspolygons der Kristallform.

Die Ecken eines Begrenzungspolygons liegen als Tupel $(P_0, P_1, P_2, \ldots)$ vor. Die Reihenfolge dieser Eckpunkte $P_0, P_1, P_2, \ldots$ muss dahingehend angepasst werden, dass zwei aufeinander folgende

Punkte eine Polygonkante definieren. Dazu wird ein Winkelsortierverfahren verwendet. Das im Folgenden vorgestellte Winkelsortierverfahren wird auf ähnliche Weise bei dem so genannten *Graham-Scan-Verfahren* zur Berechnung der konvexen Hülle einer endlichen Punktmenge in der Ebene angewandt.

Dieses Winkelsortierverfahren wurde hier modifiziert und lässt sich für konvexe Polygone im dreidimensionalen Raum verwenden. Das Verfahren ist in Abbildung 3-5 beispielhaft für die Eckpunkte eines regelmäßigen Fünfecks illustriert. Es wird zunächst einen beliebigen Eckpunkt des Polygons als Ausgangspunkt festgelegt. Dieser Ausgangspunkt wird in der Abbildung 3-5 als P_{Start} bezeichnet. Anschließend werden alle Differenzvektoren $\vec{d}_0, \vec{d}_1, \vec{d}_2, \ldots$ zwischen P_{Start} und den übrigen Eckpunkten des Polygons ermittelt.

Wie in Abbildung 3-5 dargestellt, wird einem Punkt P_i der durch die Vektoren $\vec{d}_0$ und $\vec{d}_i$ eingeschlossene Winkel zugeordnet. Zuletzt werden die Punkte nach dem Winkel, der ihnen zugeordnet ist, sortiert. Schließlich ist $(P_{Start} P'_0, P'_1, \ldots)$ die gesuchte Reihenfolge der Eckpunkte des Polygons.

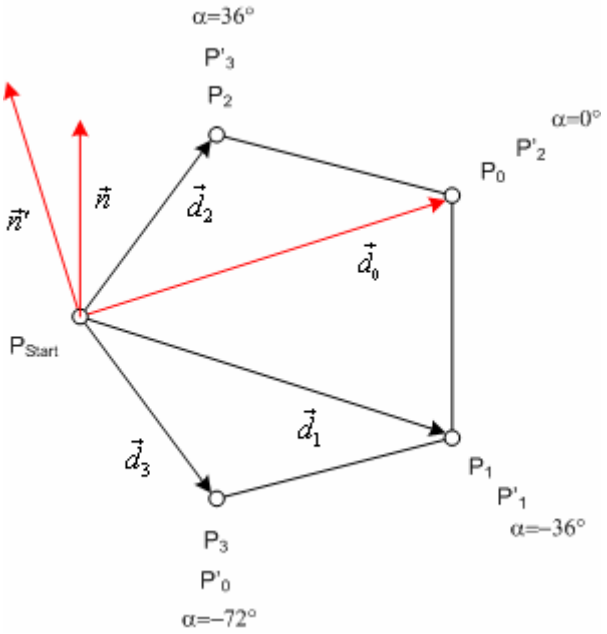


Abbildung 3-7: Das Winkelsortierverfahren am Beispiel eines regelmäßigen Fünfecks

Die Winkel lassen sich mit Hilfe des Skalarproduktes der Vektoren, die den entsprechenden Winkel einschließen, berechnen. Seien zwei Vektoren $\vec{n}$ und $\vec{n}'$ folgendermaßen definiert:

$$\vec{n} = \vec{d}_i \times \vec{d}_j, \quad i \neq j, \quad \vec{n}' = \vec{n} \times \vec{d}_0$$

Der Vektor $\vec{n}$ ist das Kreuzprodukt zweier beliebiger Differenzvektoren. Er ist Normalenvektor der Ebene, in die das Polygon eingebettet ist. Der Vektor $\vec{n}'$ liegt in dieser Ebene und ist orthogonal zu dem Normalenvektor $\vec{n}$ und dem Differenzvektor $\vec{d}_0$. Die Vektoren $\vec{n}'$ und $\vec{d}_i$ schließen den Winkel α'_i ein. Die Summe der zwei Winkel α'_i und α_i ist konstant 90°. Es gilt:

$i = 0$	$\vec{n}' \cdot \vec{d}_i = 0 \leftrightarrow \alpha_i' = 90° \leftrightarrow \alpha_i = 0°$
$i \neq 0$	$\vec{n}' \cdot \vec{d}_i > 0 \leftrightarrow \alpha_i' < 90° \leftrightarrow \alpha_i > 0°$ $\vec{n}' \cdot \vec{d}_i < 0 \leftrightarrow \alpha_i' > 90° \leftrightarrow \alpha_i < 0°$

3.4 Weitere Möglichkeiten der Berechnung von Kristallformen

In [Baie99] sind insgesamt 14 Verfahren zur Berechnung von Kristallformen beschrieben. Fünf dieser Verfahren werden im Folgenden kurz genannt:

- *Elementar-analytisch-geometrische Konstruktion*: Die Eckpunktkoordinaten des Polyeders, der die Kristallform definiert, werden mit Hilfe von Zirkel und Lineal bestimmt. Anschließend werden die Indizes der zu einem Begrenzungspolygon gehörenden Ecken ermittelt.

- *Konstruktion durch Falten*: In [Baie99] werden für Polyeder faltbare Netze, so genannte Faltnetze, angegeben. Die Ecken und Kanten (Falten) eines Faltnetzes entsprechen denen eines Polyeders. Durch das Falten des Faltnetzes erhält man den Polyeder. Der Faltvorgang wird durch Drehung der Eckpunkte um eine Falte als Drehachse realisiert.

- *Konstruktion durch Mittelbildung*: Die Schwerpunkte der Begrenzungspolygone eines konvexen Polyeders *P* sind die Eckpunkte des dualen Mittenpolyeders von *P*.

- *Konstruktion durch Projektion*: Wird zu einem Polyeder *P* die Kugel, der dieser Polyeder *P* einbeschrieben ist, ermittelt, so erhält man durch Projizieren der Schwerpunkte der Begrenzungspolygone auf die Kugeloberfläche die Ecken des zu *P* dualen Polyeders.

- *Konstruktion durch Schnitt von Großkreisen*: Die Eckpunkte eines Polyeders werden als Schnittpunkte von Großkreisen auf einer Kugel bestimmt. Ein Großkreis ist die Schnittfigur von einer Kugel und einer durch den Mittelpunkt dieser Kugel verlaufenden Ebene.

4 Erstellung der Steinformen

Im dritten Kapitel wurden dreidimensionale Modelle von einzelnen Kristallen und Methoden zur Berechnung dieser Modelle vorgestellt. Im Folgenden werden Modelle für Steine und Konglomerate vorgestellt. Diese Modelle sind ausschließlich geschlossene Oberflächen im dreidimensionalen Raum. Das geschlossene Oberflächenmodell für einen Stein ist gegeben durch eine analytische Darstellung, zu der eine polygonale Darstellung als geschlossenes polygonales Netz erzeugt wird. Um diese Modelle für Steine graphisch darzustellen, müssen die entsprechenden polygonalen Netze berechnet werden. Für die Modellierung von Steinen verwendet das Werkzeug JAMIN *zufällige* konvexe Hüllen, Subdivisionsflächen, Stream Bubbles und Interpolierte Implizite Flächen.

4.1 Zufällige konvexe Hüllen (Random Convex Hulls)

Die konvexe Hülle von zufällig gewählten Punkten im dreidimensionalen Raum wird hier als *zufällige konvexe Hülle* bezeichnet. Zufällige konvexe Hüllen liefern geschlossene Flächen. Man ermittelt zunächst eine Menge von *n* dreidimensionalen Zufallspunkten, die innerhalb eines vorgegebenen Bereichs (zum Beispiel innerhalb der Einheitskugel) liegen

$$\{P_0, P_1, \ldots, P_{n-1}\}, \quad P_i \in \mathbb{R}^3, \quad i = 0,1,\ldots,n-1$$

und berechnet anschließend die konvexe Hülle dieser Menge. In Abbildung 4-1 (links) ist die konvexe Hülle von zufällig gewählten Punkten, die auf einer Kugeloberfläche liegen, dargestellt. Dazu wurden zunächst 300 Zufallspunkte $P_0, P_1, \ldots, P_{299}$ erzeugt, die innerhalb der Einheitskugel liegen. Die Ortsvektoren $\overrightarrow{OP_0}, \overrightarrow{OP_1}, \ldots$ der Zufallspunkte wurden normiert, damit sie Ortsvektoren $\overrightarrow{OP_0'}, \overrightarrow{OP_1'}, \ldots$ von neuen Punkten $P_0', P_1', \ldots, P_{299}'$ sind, die auf der Oberfläche der Einheitskugel liegen. Statt einer Kugel kann ebenso von einem Ellipsoiden ausgegangen werden.

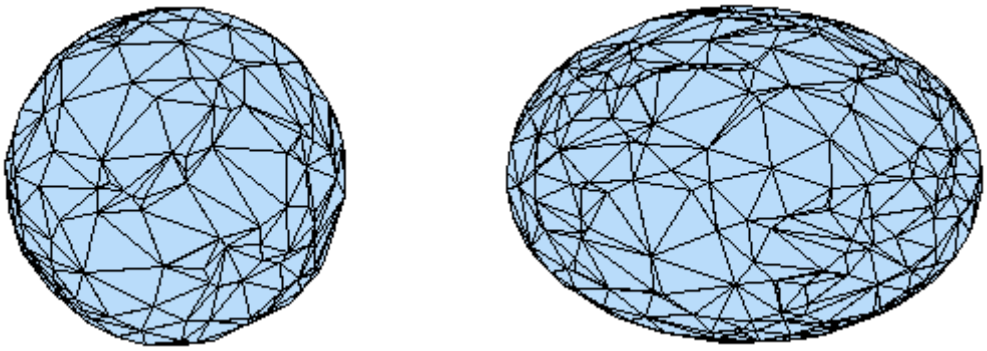


Abbildung 4-1: Zufällige konvexe Hülle einer Kugel (300 Punkte) und eines Ellipsoids (500 Punkte)

In der Natur gibt es Steine, deren äußere Farbe und Struktur sich von der des Steininneren unterscheidet. Derartige Steine werden häufig aufgeschnitten, die Schnittfläche poliert und anschließend zum Beispiel als Sammlerstück oder Schmuck verkauft. Solche Steinschnitte oder *(Polished) Rock Slices* lassen sich mit Hilfe von zufälligen konvexen Hüllen nachbilden. Nachfolgend wird anhand eines Beispiels gezeigt, wie in JAMIN unter Verwendung der zufälligen konvexen Hülle die Polygonmodelle für Steinschnitte und -scheiben erstellt werden. In diesem Beispiel wird von dreihundert Zufallspunkten ausgegangen, die auf der Oberfläche eines Halbellipsoids im dreidimensionalen Raum liegen. Das Polygonmodell dieses Halbellipsoids, dargestellt in Abbildung 4-2 (links), entsteht durch „Aufschneiden“ des Polygonmodells des Ellipsoids, das gegeben ist durch

$$x^2/1 + y^2/0{,}8 + z^2/1 = 1$$

Ausgangspunkt sind dreihundert Zufallspunkte, die zunächst auf der Oberfläche des Ellipsoids liegen, das durch die oben angegebene Gleichung beschrieben ist. Mittelpunkt dieses Ellipsoids ist der Koordinatenursprung. Nun werden die Punkte, deren *z*-Koordinate kleiner als Null ist, an der xy-Ebene gespiegelt. So erhält man 300 Zufallspunkte eines Halbellipsoids um die *z*-Achse, welches oberhalb der xy-Ebene liegt. Anschließend wird die konvexe Hülle dieser 300 Zufallspunkte berechnet. Die *z*-Koordinate jener Punkte der konvexen Hülle, deren *z*-Koordinate kleiner eines vorgegebenen Schwellenwertes (in diesem Beispiel 0,1) ist, wird gleich Null gesetzt, um den Rand der Schnittfläche zu glätten.

Auf diese Weise konstruiert man das in Abbildung 4-2 (links) dargestellte Polygonmodell des Steins. Aus diesem Polygonmodell wird anschließend die Schnittfläche extrahiert. Damit besteht das gesamte 3D Modell aus zwei Teilen:

- der Steinhülle,
- der Schnittfläche des Steins.

Die Schnittfläche ist in Abbildung 4-2 (links) hervorgehoben. Die Extraktion dieser Schnittfläche dient ausschließlich der Texturierung. Es ist möglich, dass die Steinhülle eine andere Textur als die Schnittfläche bekommt. Alle drei in Abbildung 4-2 graphisch dargestellten 3D Modelle wurden mit dem im Rahmen der Arbeit entwickelten Werkzeug JAMIN erstellt und illustriert.

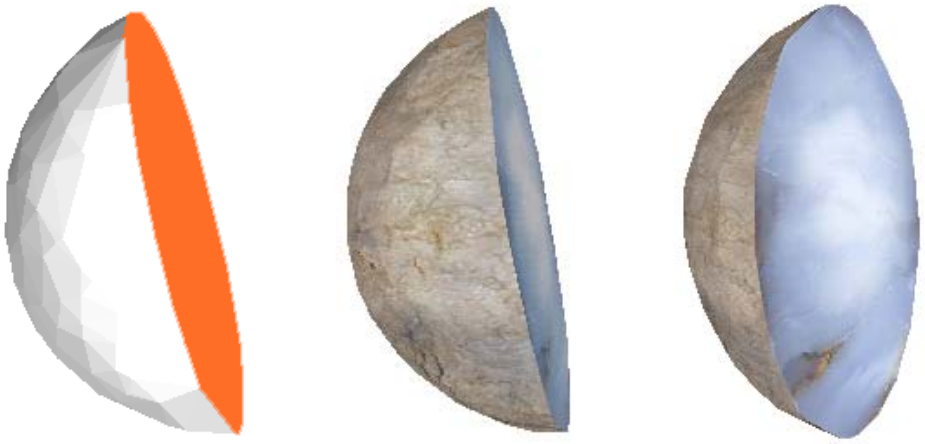


Abbildung 4-2: Halbellipsoid als 3D Modell für ein Steinschnitt

Schneidet man das Ellipsoid parallel an mindestens zwei unterschiedlichen Stellen, so erhält man die Steinscheiben. Das 3D Modell einer Steinscheibe lässt sich also durch den Schnitt des Polygonmodells der konvexen Hülle von Zufallspunkten (die sich in oder auf einem elliptischen Volumen befinden) mit zwei parallelen Ebenen erzeugen. Bei der Wahl der Schnittebenen muss beachtet werden, dass sie das Polygonmodell tatsächlich schneiden. Die Abbildung 4-3 zeigt exemplarische 3D Modelle für Steinscheiben unterschiedlicher Dicke.

Ausgangspunkt der Erstellung eines 3D Modells für eine Steinscheibe sind Zufallspunkte, die auf der Oberfläche eines Ellipsoids um den Koordinatenursprung liegen. Das Ellipsoid ist gegeben durch die Gleichung

$$x^2/a^2 + y^2/b^2 + z^2/c^2 = 1, \quad a,b,c \in \mathbb{R}$$

Das Polygonmodell dieses Ellipsoids wird als konvexe Hülle der Zufallspunkte berechnet. Für das „Aufschneiden“ des Polygonmodells geht man von zwei Höhenwerten h_1 und h_2 aus. Sie geben die zwei parallelen Schnittebenen $E_1 : z = h_1$ und $E_2 : z = h_2$ an. Nun wird die z-Koordinate aller Punkte (des Polygonmodells), deren z-Koordinate kleiner als h_1 ist gleich h_1 gesetzt und die z-Koordinate jener Punkte, deren z-Koordinate größer als h_2 ist gleich h_2 gesetzt. Da die so entstandenen Punkte möglicherweise außerhalb der Schnittfläche liegen, muss auf sie noch eine Transformation angewendet werden. Sei P' der durch Transformation aus $P = (x_P, y_P, z_P)$ erzeugte Punkt, dann berechnet sich P' nach:

$$P' = \left(k_P x_P, k_P y_P, z_P\right), \text{ mit } k_P = \sqrt{\frac{1-\left(z_P/c\right)^2}{\left(x_P/a\right)^2 + \left(y_P/b\right)^2}}$$

Jeder der Punkte P' erfüllt die Gleichung, durch welche das Ellipsoid beschrieben ist und befindet sich somit auf seiner Oberfläche:

$$\left(k\left(P\right)x_P\right)^2/a^2 + \left(k\left(P\right)y_P\right)^2/b^2 + z_P^2/c^2 = 1$$

$$\frac{1-(z_P/c)^2}{(x_P/a)^2+(y_P/b)^2}\cdot\frac{x_P}{a^2}+\frac{1-(z_P/c)^2}{(x_P/a)^2+(y_P/b)^2}\cdot\frac{y_P}{b^2}+\frac{z_P^2}{c^2}=\frac{1-(z_P/c)^2}{(x_P/a)^2+(y_P/b)^2}\cdot\left(\frac{x_P}{a^2}+\frac{y_P}{b^2}\right)+\frac{z_P^2}{c^2}=1$$

In der Abbildung 4-3 (rechts) hat das 3D Modell für die Steinscheibe entsprechende Texturen. Die Steinschnitte und -scheiben sind typische Beispiele für Steinmodelle die, wie in Abbildung 4-2 und 4-3 dargestellt, verschiedene Texturen haben. Deswegen muss eine Extraktion der Schnittfläche vorgenommen werden. Bei den 3D Modellen für Steinscheiben sind das in diesem Fall:

- zwei Schnittflächen und
- eine Mantelfläche.

Jede der zwei Schnittflächen bekommt eine Textur für dieselbe Steinpolitur. Das in Abbildung 4-3 (rechts) dargestellte 3D Modell für die Steinscheibe hat eine grüne Achattextur für die zwei Schnittflächen und eine Felstextur für die Mantelfläche. Die in Abbildung 4-3 dargestellten 3D Modelle wurden mit JAMIN erstellt. Diese 3D Modelle, so auch jene, die in Abbildung 4-2 dargestellt sind, lassen sich mit JAMIN jeweils durch ein Applet auf Webseiten präsentieren.

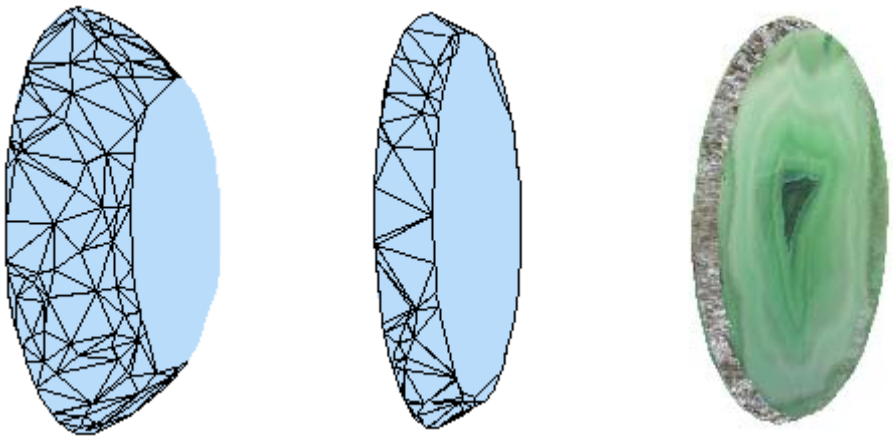


Abbildung 4-3: 3D Modelle für Steinscheiben unterschiedlicher Dicke

4.2 Subdivisionsflächen

In diesem Kapitel soll ein Verwendungsbeispiel für die Erstellung von Steinformen und die Idee der Subdivisionsflächen vorgestellt werden.

Subdivisionsflächen sind iterative Verfahren, mit deren Hilfe aus einem vorgegebenen Polygonmodell, dem so genannten *Kontrollnetz*, visuell glattere Oberflächen erzeugt werden. In jedem Schritt werden die Polygone des Kontrollnetzes in drei oder vier kleinere Polygone zerlegt [Hecht05]. Diese Zerlegung liefert ein neues Polygonmodell (Abbildung 4-4). In manchen Fällen enthält das neue Polygonmodell, das ausgehend von dem vorgegebenen Modell erzeugt wurde, auch dieselben Punkte des vorgegebenen Modells. Die Zerlegung wird so oft wiederholt, bis die gewünschte Glätte erreicht ist. Man unterscheidet die approximativen von den interpolierenden Verfahren (Tabelle 4-1). Bei den interpolierenden Verfahren enthält das im Verfeinerungsschritt

neu erzeugte Polygonmodell auch immer dieselben Punkte des Modells, von dem die Zerlegung in diesem Verfeinerungsschritt ausgegangen ist. In diesem Fall verläuft die glatte Oberfläche durch die Punkte des Kontrollnetzes. Die approximativen Verfahren berechnen alle Punkte neu.

Das folgende Beispiel zeigt, wie JAMIN mit Hilfe der Subdivisionsflächen Polygonmodelle für größere Steine erstellt. An dieser Stelle sei erwähnt, dass die im Rahmen der vorliegenden Arbeit verwendete Klassenbibliothek *JavaView* auch ein geeignetes Hilfsmittel für das Erzeugen von Subdivisionsflächen ist.

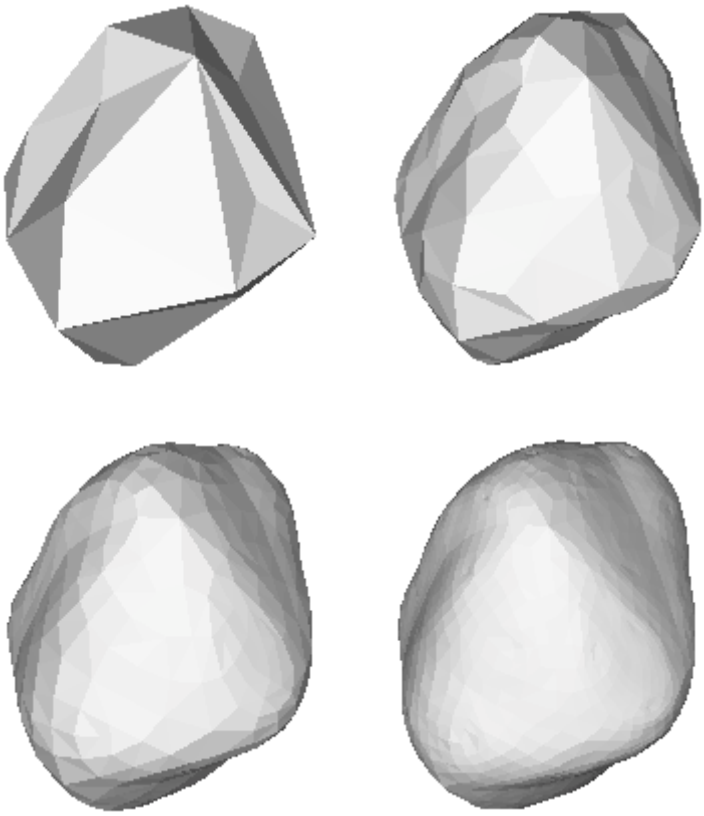


Abbildung 4-4: Erstellung einer Steinform mit Hilfe von Subdivisionsflächen

Die Abbildung 4-4 zeigt Polygonmodelle einer Steinform, die jeweils in einem unterschiedlichen Verfeinerungsgrad vorliegen. Das Kontrollnetz ist in Abbildung 4-4 (oben links) dargestellt. In diesem Beispiel wurde das Kontrollnetz in zwei Schritten erstellt. Zunächst wurde die konvexe Hülle von 60 Zufallspunkten, die sich in einem elliptischen Volumen um den Koordinatenursprung befinden berechnet. Anschließend wurde jeder der zur den konvexen Hülle Punkten $P_0, P_1, \ldots, P_{59}$ gehörende Ortsvektor $\overrightarrow{OP_i}$, $i = 0,1,\ldots,59$ mit einem Faktor

$$d_i = 1 - \frac{t_i}{\left|\overrightarrow{OP_i}\right|}, \quad t_i \in \mathbb{R}, \quad -t \le t_i \le t$$

multipliziert, wobei t_i eine reelle Zufallszahl ist, deren Absolutwert (Betrag) kleiner oder gleich einer vorgegebenen Zahl *t* ist. Die Differenz der Beträge der Vektoren $d_i \cdot \overrightarrow{OP_i}$ und $\overrightarrow{OP_i}$ ist t_i. Dadurch entstehen „Kerben" und „Ausbeulungen".

Das so entstandene Kontrollnetz ist eine geschlossene Fläche, aber nicht notwendig eine konvexe Hülle. Die Zahl *t* ist ein Schwellenwert, der darauf Einfluss hat, in welchem Maß sich die äußere Form des Kontrollnetzes von jener der konvexen Hülle unterscheidet.

Die Idee von Subdivisionsflächen wurde erstmalig von

- Edwin Catmull und Jim Clark und
- Daniel Doo und Malcolm Sabin

durch eine Veröffentlichung im Journal *Computer Aided Design* im November 1978 eingeführt [Lee00]. Von diesem Zeitpunkt an ist eine Vielzahl von Unterteilungsverfahren für Subdivisionsflächen entwickelt worden *(Loop Subdivision, Modified Butterfly Subdivision und Catmull-Clark Subdivision)*. Subdivisionsflächen wurden im Bereich der Computergraphik so populär, dass sie für die Filmproduktion von zum Beispiel *Geri's Game*, *Toy Story* und *Toy Story II* eingesetzt wurden. *Geri*'s Hände, Kopf, Jacke und Hose wurden jeweils mit einer Subdivisionsfläche modelliert [Lee00].

Die verschieden Verfahren für Subdivisionsflächen unterscheiden sich in der Art und Weise, wie die Polygone unterteilt werden und wie die Glättung berechnet wird. In der folgenden Tabelle werden Subdivisionsverfahren genannt, eingeteilt in approximative und interpolierende Verfahren, wie sie in JavaView zur Verfügung stehen.

approximative Verfahren	Catmull-Clark S., Sqrt3 S., Doo-Sabin S., Loop S.
interpolierende Verfahren	Modified Butterfly S., Four-Point S.

Tabelle 4-1: approximative und interpolierende Subdivisionsverfahren[5]

Einen Überblick über die verschiedene Wirkung der einzelnen Verfahren gibt Abbildung 4-5. Sie zeigt Subdivisionsflächen, deren Kontrollnetz vier Tetraeder sind. Die vier Tetraeder sind räumlich derart angeordnet, dass jeweils drei Tetraeder zusammen drei gemeinsame Eckpunkte besitzen. Das Kontrollnetz ist in Abbildung 4-5 (oben links) dargestellt. Die Subdivisionsflächen in Abbildung 4-5 ergeben sich alle durch dreimal wiederholte Verfeinerung dieses Kontrollnetzes bei Anwendung der entsprechenden Subdivisionsverfahren. In Abbildung 4-5 ist die resultierende Butterfly Subdivisionsfläche (oben rechts), die Doo-Sabin Subdivisionsfläche (unten links) und schließlich die Loop Subdivisionsfläche (unten rechts) dargestellt. Die graphische Darstellung dieser Subdivisionsflächen einschließlich des Kontrollnetzes wurde von JAMIN erzeugt – unter Verwendung von JavaView berechnet und durch Idx3d visualisiert. Anhand der graphischen Darstellungen soll vorgestellt werden, wie unterschiedlich die geometrische Form der verfeinerten Netze für eine beispielhafte Anwendung des Butterfly-, Doo-Sabin- und Loop Subdivisionsverfahrens ist. Deutlich zu erkennen ist auch der Unterschied zwischen approxima-

[5] [ACSE02], [Kilt03]

tiven und interpolierenden Verfahren. So zeigt die graphische Darstellung der Butterfly Subdivisionsfläche in Abbildung 4-5 (oben rechts), dass das Kontrollnetz enthalten bleibt. Eine detaillierte Beschreibung dieser und noch weiterer Subdivisionsverfahren kann den Quellen [ACSE02] und [Kilt03] entnommen werden. Hier sollte nur ein Einblick in die Idee der Subdivisionsflächen gegeben werden.

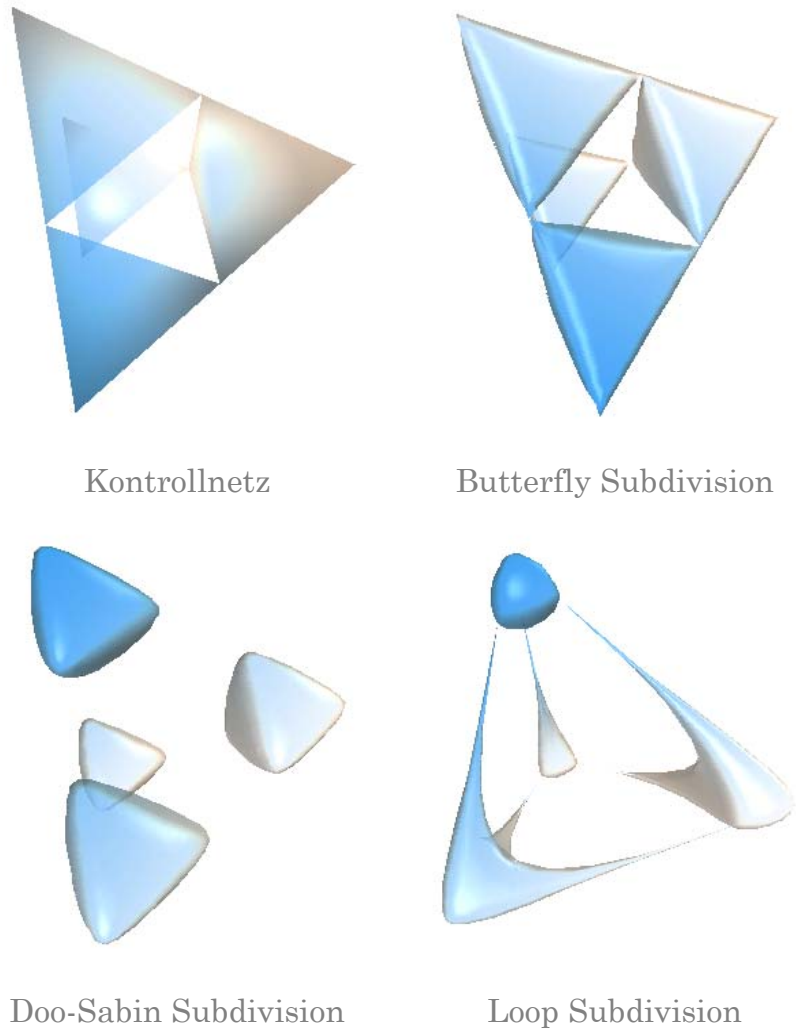


Abbildung 4-5: Subdivisionsverfahren

4.3 Stream Bubbles

Stream Bubbles sind eine weitere Möglichkeit für die Modellierung geschlossener und glatter Oberflächen. Eine Besonderheit dabei ist, dass sich mit Stream Bubbles interaktive Verformungen des 3D Modells umsetzen lassen.

Zunächst sollen die Begriffe Stream Bubble, NURBS und Basis Splines erläutert werden. Eine Stream Bubble ist eine NURBS *(Non Uniform Rational Basis Splines)* Fläche mit der besonderen Eigenschaft, dass die Fläche geschlossen ist. Eine NURBS Fläche wird als eine Summe von B-Spline (Basis Spline) Funktionen berechnet. Sie ist visuell glatt und lässt sich auf einfache Weise, durch Variieren von Kontrollpunkten, deformieren. Die einfache Manipulierbarkeit der Ober-

fläche in einer stets visuell glatten Darstellung ist eine Eigenschaft, die NURBS für interaktive Verfahren geeignet macht.

Nachfolgend wird beschrieben, wie Basis Splines, NURBS Flächen und Stream Bubbles mathematisch definiert sind. Eine B-Spline Funktion $N_i^k(t)$ der Ordnung *k* ist ein segmentweise definiertes Polynom vom Grad $k-1$, das an den Segmentübergängen C^{k-2}-stetig ist. Gegeben sei ein Vektor (auch Trägervektor genannt), dessen Komponenten geordnete Parameterwerte sind:

$$T = (t_0, t_1, \ldots, t_{n-1}, t_n, t_{n+1}, \ldots, t_{n+k}), \quad t_0 < t_1 < t_2 < \ldots, \quad t_0, t_1, t_2, \ldots \in \mathbb{R},$$

die Stützstellen für die Approximation darstellen. Mit Hilfe der B-Spline Funktion $N_i^k(t)$ soll eine in den Punkten t_i gegebene Funktion *f* approximiert werden. Damit erhält man die folgende Menge bestehend aus Segmenten:

$$N^k = \left\{ N_i^k(x) \middle| \; t_i \le x \le t_{i+1}, \; i = 1, \ldots, k \right\}$$

Die Basis Spline Funktionen $N_i^k(t)$ sind rekursiv definiert nach Cox de Boor:

$$N_i^1(t) = \begin{cases} 1 & \textit{für} \;\; t_i \le t < t_{i+1} \\ 0 & \textit{sonst} \end{cases}$$

$$N_i^k(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_i^{k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1}^{k-1}(t), \quad i = 0, \ldots, n$$

Das heißt N_i^1 sind in dem *i*-ten Intervall identisch Eins sonst Null. Im zweiten Schritt berechnet man N_i^2 in dem Intervall $[t_i, t_{i+2}]$ nach der oben dargestellten Rekursion und setzt diesen Vorgang fort.

Eine NURBS Fläche $Q(u,v)$ der Ordnung *k* mit $(m+1)\cdot(n+1)$ Kontrollpunkten, $m+1$ Kontrollpunkten in *u*-Richtung und $n+1$ Kontrollpunkten in *v*-Richtung, mit dem Trägergebiet

$$T = (u_0, u_1, \ldots, u_{m-1}, u_m, u_{m+1}, \ldots, u_{m+k}) \times (v_0, v_1, \ldots, v_{n-1}, v_n, v_{n+1}, \ldots, v_{n+k})$$

hat die Parameterdarstellung:

$$Q(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} P_{i,j} N_i^k(u) N_j^k(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} N_i^k(u) N_j^k(v)}, \; u \in [u_{k-1}, u_{m+1}], \; v \in [v_{k-1}, v_{n+1}]$$

Die Kontrollpunkte $P_{i,j}$ einer NURBS Fläche werden auch de-Boor-Punkte genannt. Die Menge $\{P_{i,j}\}$ bildet das so genannte de-Boor-Netz. Die Gewichtungsfaktoren $w_{i,j}$ stellen ein zusätzli-

ches Designelement der NURBS Fläche dar [HoLa92]. Sie haben Einfluss auf die äußere Gestalt der NURBS Fläche.

Eine Stream Bubble ist eine geschlossene NURBS Fläche, die einer dreidimensionalen Gitterzelle einbeschrieben ist. Diese Gitterzelle heißt Kontrollform der Fläche. Verändert man die Position eines Eckpunktes der Kontrollform, so ändert man die äußere Gestalt der Stream Bubble. Die geschlossene NURBS Fläche passt sich an die Kontrollform derart an, dass sie immer innerhalb der Kontrollform liegt. In den meisten Fällen, wie in Abbildung 4-6 dargestellt, ist die Gitterzelle ein Quader. Jedoch ist auch ein Vierflächner, zum Beispiel eine tetraederförmige Gitterzelle als Kontrollform möglich [ZhPa00].

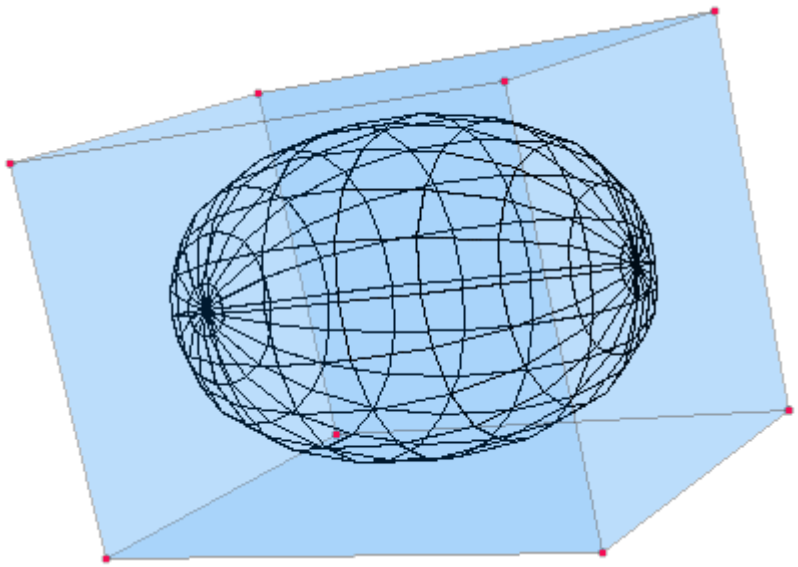


Abbildung 4-6: Stream Bubble vierter Ordnung in einer sechsflächigen Kontrollform

Stream Bubbles werden in JAMIN unter anderem für die Modellierung von Felsen und größeren Steinen verwendet. Mit Stream Bubbles lassen sich dreidimensionale Steinmodelle erzeugen, deren Gestalt durch das Verändern einzelner Eckpunkte der Kontrollform beeinflusst wird. Abbildung 4-7 zeigt die beispielhafte Modellierung eines Felsens durch eine Stream Bubble vierter Ordnung, deren Kontrollform 66 Eckpunkte besitzt. Die rot dargestellten Punkte in Abbildung 4-7 (oben) sind die Eckpunkte der Kontrollform. Sie lassen sich mit der Maus bewegen, so dass beispielsweise die Form eines Felsens erzeugt wird, die in Abbildung 4-7 (oben rechts) dargestellt ist. Demnach zeigt diese Abbildung ein Feature von JAMIN für die interaktive Erstellung verschiedener Fels- und Steinformen.

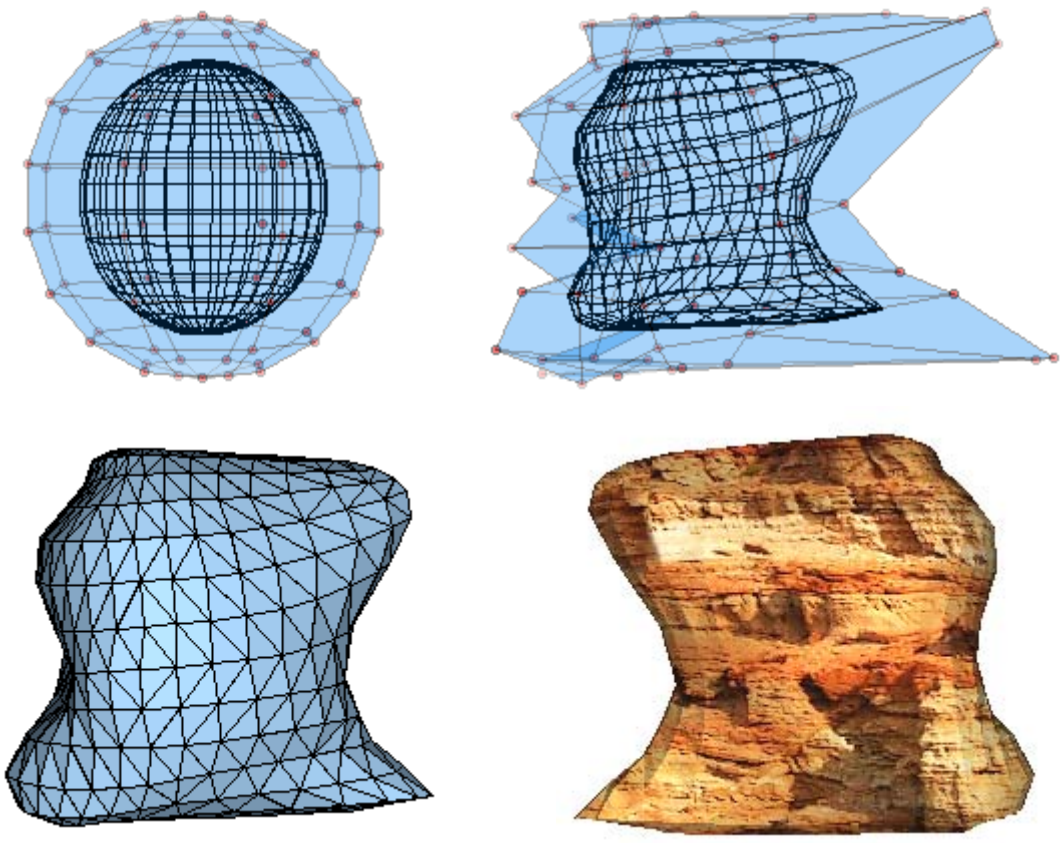


Abbildung 4-7: Erstellung eines 3D Modells für einen Felsen

Im Folgenden werden Stream Bubbles beschrieben, die einer sechsflächigen Kontrollform einbeschrieben sind. Die Kontrollform entspricht dem in Abbildung 4-8 dargestellten Einheitswürfel mit den folgenden 8 Eckpunkten $P_0,\ldots,P_7$:

$$P_0=(0,0,1) \quad P_1=(0,1,1) \quad P_2=(0,1,0) \quad P_3=(0,0,0)$$
$$P_4=(1,0,1) \quad P_5=(1,1,1) \quad P_6=(1,1,0) \quad P_7=(1,0,0)$$

Die Menge der Kontrollpunkte $\{P_{i,j}\}$ sei in Form einer Matrix G angegeben mit:

$$G=(g_{ij})_{i=0,\ldots,m;\,j=0,\ldots,n}, \quad g_{ij}=P_{i,j}$$

Die Matrix G heißt Geometriematrix und ist eine genaue Beschreibung des de-Boor-Netzes. Die Einträge der Geometriematrix sind die Kontrollpunkte der NURBS Fläche (Abbildung 4-9). Die Parameterform der NURBS Fläche lässt sich in einer entsprechenden Matrixschreibweise mit der Geometriematrix wie folgt darstellen:

$$\sum_{i=0}^{m}\sum_{j=0}^{n}P_{i,j}N_i^k(u)N_j^k(v) \;=\; \left(N_0^k(u) \;\cdots\; N_m^k(u)\right)\cdot\begin{pmatrix}P_{0,0} & \cdots & P_{0,n}\\ \vdots & \ddots & \vdots\\ P_{m,0} & \cdots & P_{m,n}\end{pmatrix}\cdot\begin{pmatrix}N_0^k(v)\\ \vdots\\ N_n^k(v)\end{pmatrix}$$

Die linke Seite der obigen Gleichung ist dabei die Parameterform der Fläche ohne die Gewichtungsfaktoren $w_{i,j}$.

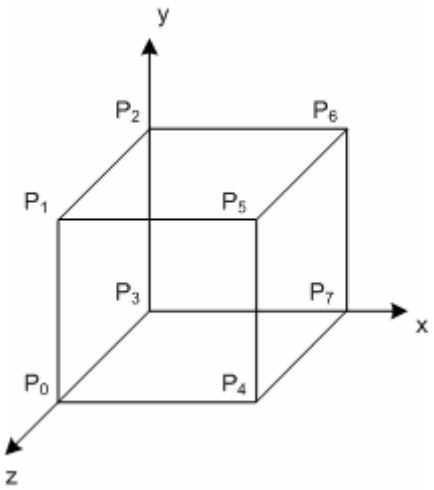


Abbildung 4-8: Würfel als Gitterzelle einer Stream Bubble

In Abbildung 4-9 ist die Geometriematrix der Stream Bubble dritter Ordnung, welche dem Würfel einbeschrieben ist, dargestellt. Man kann erkennen, dass sich in jeder Zeile/Spalte die ersten zwei Einträge am Ende der Zeile/Spalte wiederholen. Die Anzahl der sich wiederholenden Einträge pro Zeile/Spalte entspricht genau der Ordnung minus Eins. Dies ist schließlich der Grund dafür, dass die in diesem Fall gegebene NURBS Fläche - die dem Würfel einbeschriebene Stream Bubble - geschlossen ist.

$$\begin{pmatrix} P_0 & P_1 & P_6 & P_7 & P_0 & P_1 \\ P_4 & P_5 & P_2 & P_3 & P_4 & P_5 \\ P_7 & P_6 & P_1 & P_0 & P_7 & P_6 \\ P_3 & P_2 & P_5 & P_4 & P_3 & P_2 \\ P_0 & P_1 & P_6 & P_7 & P_0 & P_1 \\ P_4 & P_5 & P_2 & P_3 & P_4 & P_5 \end{pmatrix}$$

Abbildung 4-9: Geometriematrix der dem Würfel einbeschriebenen Stream Bubble

4.4 Interpolierte Implizite Flächen

Das im Rahmen dieser Arbeit entwickelte Werkzeug JAMIN kann Polygonmodelle für implizite Flächen in einem vorgegebenen Intervall erstellen. Im Folgenden werden implizite Flächen sowie eine spezielle Teilklasse, die *Blobby Surfaces*, beschrieben. JAMIN verwendet beim Erstellen der Polygonmodelle für unter anderem *Blobby Surfaces* eine Approximation der Gaußfunktion. Die Entwicklung dieser Approximation ist auch ein Beitrag der vorliegenden Arbeit und wird am Ende dieses Kapitels beschrieben.

Geschlossene Flächen im $\mathbb{R}^3$ sind häufig durch eine so genannte implizite Darstellung gegeben. Im Unterschied zur expliziten Darstellung (in der die Punkte berechnet werden) ist die implizite Darstellung (in der eine Bildungsvorschrift für die Punkte direkt angegeben wird) eine Funktion zusammen mit einer Bedingung, also eine Lösungsmenge. Man verwendet eine Funktion *f*, hier implizite Funktion genannt, mit

$$f : \mathbb{R}^3 \to \mathbb{R}$$

und beschreibt eine Fläche *F* als Lösungsmenge

$$F = \left\{ q \in \mathbb{R}^3 \,\middle|\, f(q) = k \right\}$$

für eine Konstante $k \in \mathbb{R}$. So ist beispielsweise die Einheitskugel durch die Funktion

$$f(q) = 1 - |q|$$

für Punkte $q \in \mathbb{R}^3$ definiert. Alle Punkte, die auf der Kugeloberfläche liegen, erfüllen jeweils die Gleichung $f(q) = 0$.

Hier ist eine spezielle Teilklasse von Flächen, die *Blobby Surfaces* interessant. Die *Blobby Surfaces* kann man sich als Flächen mit blasenartiger Oberfläche vorstellen. Solche Flächen erzeugt man durch Zusammensetzung primitiverer Flächen, den *Blinn Objekten*, oder kurz *Blobs*. *Blinn Objekte* werden häufig auch als *Metaballs* bezeichnet und gehen auf James F. Blinn[6] zurück. *Blinn Objekte* werden durch Gaußfunktionen $g : \mathbb{R}^3 \to \mathbb{R}$ mit

$$g(q) = \frac{1}{\sigma\sqrt{2\pi}} e^{-|q-c|^2/2\sigma^2} \;,\; q, c \in \mathbb{R}^3 \,,\; \sigma \in \mathbb{R}^+$$

und Lösungsbedingungen

$$B_g = \left\{ q \in \mathbb{R}^3 \,\middle|\, g_{\sigma,c}(q) = k \right\}$$

charakterisiert. Sie sind radialsymmetrisch[7] (aufgrund der strengen Monotonie von *e*). Es ist *c* der Mittelpunkt und σ die Standardabweichung des *Blinn Objektes*. Durch Variieren der Standardabweichung kann der Radius des *Blinn Objektes* angepasst werden [TuBr02]. Zusammengesetzt ergeben mehrere *Blinn Objekte* B_i mit den impliziten Funktionen g_i und Konstanten k_i eine *Blobby Surface* durch die Bildungsvorschrift:

$$f(q) = -k + \sum_{i=1}^{n} g_i(q) \,,\; k = \sum_{i=1}^{n} k_i$$

---

[6] James F. Blinn hat auch 1976 das Bump Mapping entwickelt und arbeitet beim Jet Propulsion Laboratory des California Institute of Technology, das unter anderem computererzeugte Animationen entwickelt [Sult04].

[7] Eine Funktion *g*: $\mathbb{R}^n \to \mathbb{R}$ heißt radialsymmetrisch um den Punkt c, wenn $g(q_1) = g(q_2)$ für alle $q_1, q_2 \in \mathbb{R}^n$ mit $\|q_1\text{-}c\| = \|q_2\text{-}c\|$ gilt.

und die Lösungsbedingung

$$B_f = \left\{ q \in \mathbb{R}^3 \,\middle|\, f(q) = a \right\}$$

Der Wert *a* ist eine Konstante, die eine bestimmte Fläche aus der Schar der Flächen spezifiziert, die jeweils durch die Summe der Gaußfunktionen definiert sind. Wenn die Mittelpunkte zweier *Blinn Objekte* dicht genug aneinander liegen, sieht die implizite Fläche so aus als seien die zwei kugelförmigen Oberflächen miteinander verschmolzen (Abbildung 4-13).

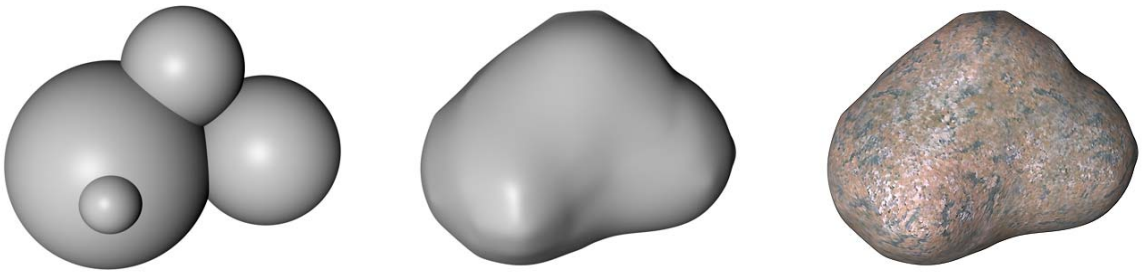


Abbildung 4-10: Blobby Surface mit 4 Blobs für die Modellierung eines Findlings

In Abbildung 4-10 ist ein beispielhaftes 3D Modell für einen Findling (hier Granit) dargestellt. Das Modell ist eine Blobby Surface, welche sich aus vier unterschiedlich großen Blobs, dargestellt in Abbildung 4-10 (links), zusammensetzt. In diesem Beispiel wurde für das 3D Modell eine Granit-Textur verwendet.

An dieser Stelle soll noch kurz begründet werden, warum die radialsymmetrischen Gaußfunktionen und nicht die bekannten Kugelfunktionen (die ja ebenfalls radialsymmetrisch sind) verwendet werden. Würden anstelle zweier Gaußfunktionen zwei Kugelfunktionen, zum Beispiel die Funktionen

$$f_1(q) = 4 - |q|, \;\; f_2(q) = 4 - |q - (6,0,0)|$$

addiert und die Lösungsbedingung

$$B_f = \left\{ q \in \mathbb{R}^3 \,\middle|\, f_1(q) + f_2(q) = 0 \right\}$$

verwendet werden, so enthält die implizite Fläche zwei kugelförmige Oberflächen, die nicht wie in Abbildung 4-13 miteinander verschmolzen sind. In Abbildung 4-11 ist diese implizite Fläche durch JAMIN graphisch dargestellt.

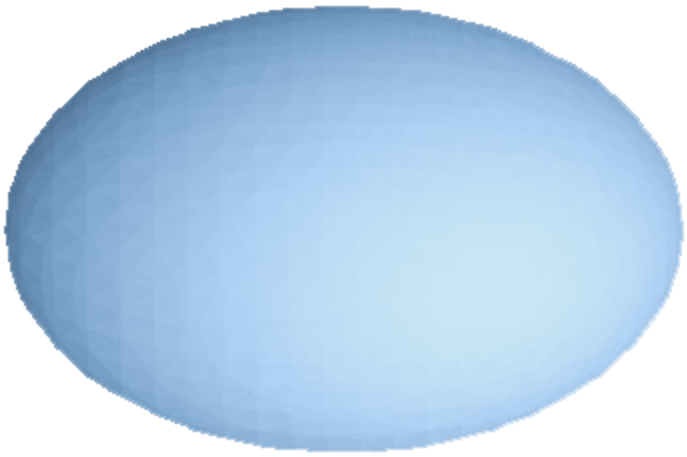


Abbildung 4-11: Implizite Fläche definiert als Summe zweier Kugelgleichungen

Gaußfunktionen sind in der Berechnung sehr aufwendig, daher werden diese häufig durch Polynome angenähert [TuBr02]. Hierfür soll eine Approximation der Gaußfunktion vorgestellt werden, die in ihrer Genauigkeit an das Problem angepasst werden kann.

Wegen $g(q_1) = g(q_2)$ falls $|q_1| = |q_2|$ genügt es, eine einstellige Funktion p zu finden, so dass gilt:

$$p(|q-c|) \approx g_{\sigma,c}(q).$$

Die radialsymmetrische Funktion g mit den Parametern c und σ lässt sich auch schreiben als:

$$g_{\sigma,c}(q) = g_{\sigma,0}(|q-c|,0,0).$$

Sei p ein biquadratisches Polynom, durch welches die Gaußfunktion g angenähert werden soll. Der Verlauf der Kurve eines biquadratischen Polynoms, das ein lokales Maximum und zwei lokale Minima besitzt, ähnelt etwa dem der Kurve von g. Das Polynom hat entsprechend die Form

$$p(x) = a_0 + a_2x^2 + a_4x^4 \ldots + a_{2n}x^{2n} = \sum_{i=0}^{n} a_{2i}x^{2i}\ ,\ x \in \mathbb{R}$$

Damit p die Gaußfunktion im Intervall $[-n \cdot t, n \cdot t]$ annähert, wird festgelegt, dass:

- $p(0) = g(0)$ also $a_0 = g(0)$ und
- $\frac{\partial}{\partial x} p(x) = \frac{\partial}{\partial x} g(x)$ für gewisse Stützstellen $x_1 = t, x_2 = 2 \cdot t, \ldots, x_n = n \cdot t\ ,\ t \in \mathbb{R}^+$

erfüllt sein soll. Die positive reelle Zahl t ist ein Parameter für die Genauigkeit der Approximation. In dem Intervall $[-n \cdot t, n \cdot t]$ entspricht in gleichen Abständen die erste Ableitung des Polynoms der ersten Ableitung der Gaußfunktion. Dieser Abstand wird durch den Parameter t

bestimmt. Mit Hilfe der zwei oben genannten Bedingungen lassen sich die Koeffizienten a_i, $i = 2,4,\ldots,2n$ von *p* über das folgende Gleichungssystem bestimmen:

$$\frac{\partial}{\partial x} p(t) = \frac{\partial}{\partial x} g(t),\ \frac{\partial}{\partial x} p(2t) = \frac{\partial}{\partial x} g(2t),\ \ldots,\ \frac{\partial}{\partial x} p(nt) = \frac{\partial}{\partial x} g(nt)$$

Schreibt man ausführlich die erste Ableitung des Polynoms als Summe auf, so erhält man

$$\begin{aligned} 2a_2 \cdot 1t + 4a_4 \cdot (1t)^3 + \ldots + 2na_{2n} \cdot (1t)^{2n-1} &= g'(1t) \\ 2a_2 \cdot 2t + 4a_4 \cdot (2t)^3 + \ldots + 2na_{2n} \cdot (2t)^{2n-1} &= g'(2t) \\ &\vdots \\ 2a_2 \cdot nt + 4a_4 \cdot (nt)^3 + \ldots + 2na_{2n} \cdot (nt)^{2n-1} &= g'(nt) \end{aligned}$$

beziehungsweise in Matrixschreibweise

$$\begin{pmatrix} a_2 \\ a_4 \\ \vdots \\ a_{2n} \end{pmatrix} = \begin{pmatrix} 2\cdot(1t) & 4\cdot(1t)^3 & \cdots & 2n\cdot(1t)^{2n-1} \\ 2\cdot(2t) & 4\cdot(2t)^3 & \cdots & 2n\cdot(2t)^{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 2\cdot(nt) & 4\cdot(nt)^3 & \cdots & 2n\cdot(nt)^{2n-1} \end{pmatrix}^{-1} \cdot \begin{pmatrix} g'(1t) \\ g'(2t) \\ \vdots \\ g'(nt) \end{pmatrix}$$

Werden alle Gaußfunktionen g_i durch solche Polynome p_i angenähert, erhält man die Summe:

$$-k + \sum_{i=1}^{m} p_i(x) = -k + \sum_{i=1}^{m} \sum_{j=0}^{n_i} a_{ij} x^{2j}$$

Anhand der obigen Gleichung ist zu erkennen, dass neben dem geringeren Berechnungsaufwand von *p* gegenüber *g* ein weiterer Vorteil der Approximation darin besteht, dass sich die Polynome auf eine einfachere Weise addieren lassen als die Gaußfunktionen.

In Abbildung 4-13 sieht man im Vergleich eine *Blobby Surface* berechnet durch die Gaußfunktion und dieselbe *Blobby Surface* angenähert durch *p*. Gegeben sei dazu eine *Blobby Surface* bestehend aus zwei *Blinn Objekten*:

$$f(q) = -k + g_1(q) + g_2(q),\ B_f = \left\{ q \in \mathbb{R}^3 \,\middle|\, f(q) = 0 \right\}$$

Beide *Blinn Objekte* besitzen in diesem Beispiel die Standardabweichung $\sigma_1 = \sigma_2 = 2$. Eines der *Blinn Objekte* hat den Mittelpunkt $(0,0,0)$, das andere hat den Mittelpunkt $(7,0,0)$. Für dieses Beispiel wird die Konstante $k = 0,04$ gewählt. Die Approximation der Gaußfunktion $g(q)$ mit $|q - c| = x$ erfolgt durch das Polynom $p(x)$ vom Grad 10 ($n = 5$) mit dem Parameter $t = 1$. Nach Einsetzen der Parameter erhält man die Funktionen:

$$g(x) = \frac{1}{2\sqrt{2\pi}} e^{-x^2/(2\cdot 2^2)}$$

$$p(x) = -1{,}41 \cdot 10^{-8} x^{10} + 1{,}36 \cdot 10^{-6} x^{8} - 5{,}9 \cdot 10^{-5} x^{6} + 1{,}53 \cdot 10^{-3} x^{4} - 2{,}49 \cdot 10^{-2} x^{2} + 1{,}99 \cdot 10^{-1}$$

In der Abbildung 4-12 ist der Funktionsgraph der Gaußfunktion *g* (rot) und der Funktionsgraph des Polynoms *p* (blau) dargestellt. Die *Blobby Surface* wird durch *p* in dem Intervall $[-5,5]$ angenähert. Dieses Intervall ist in Abbildung 4-12 hervorgehoben. Die Beträge $|q-c|$, die nicht in dem Intervall $[-5,5]$ liegen, werden gleich der oberen Intervallgrenze gesetzt.

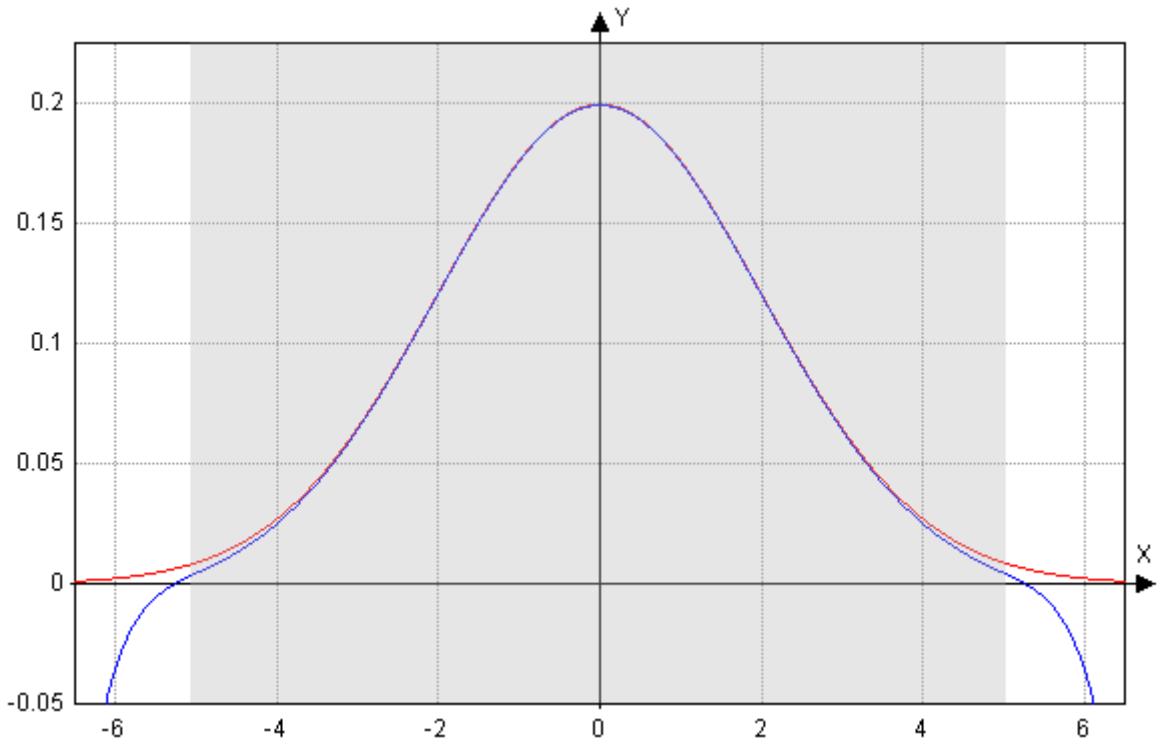


Abbildung 4-12: Funktionsgraphen von *g* und *p*

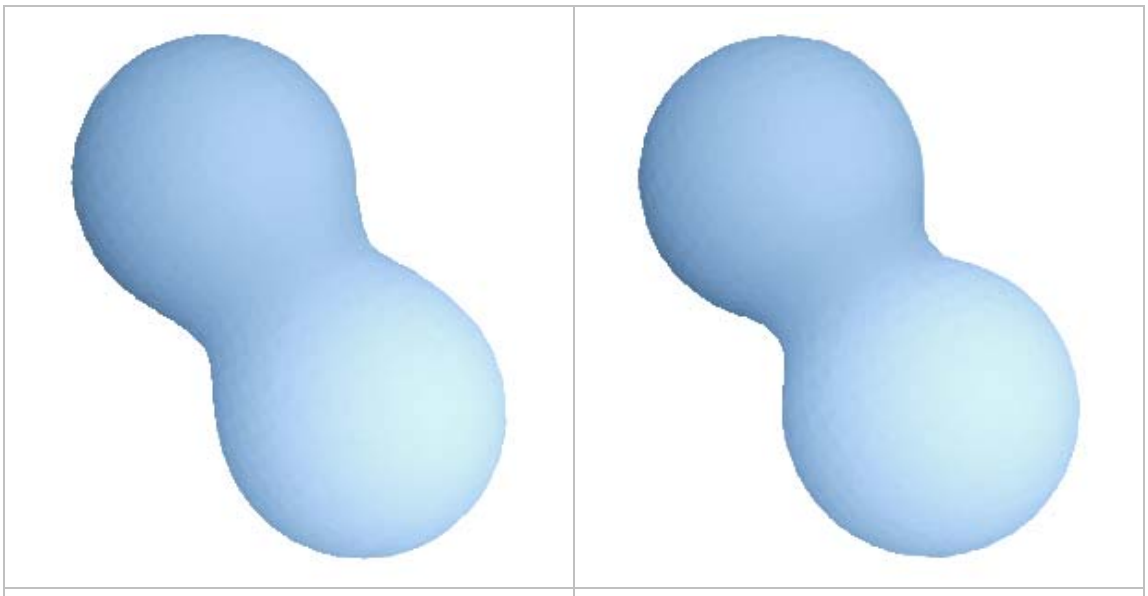
Betrachtet man das Intervall, in dem das erste *Blinn Objekt* durch das Polynom angenähert wird, so erkennt man, dass die Abweichung der Funktionswerte $g(x)$ von $p(x)$ für Beträge $|q-c|$ größer als vier und kleiner als fünf am deutlichsten ist. Zur Veranschaulichung dieser Abweichung gibt Tabelle 4-1 einige Funktionswerte der beiden Funktionen an.

x	$g(x)$	$p(x)$
4,0	0,0269955	0.02496
4,2	0,0219918	0.0196016
4,4	0,0177373	0.0149797
4,6	0,0141635	0.0110175
4,8	0,0111973	0.00758069
5,0	0,00876415	0.00442969

Tabelle 4-2: Funktionswerte der Funktionen *g* und *p* für $x \in [4,5]$

In Abbildung 4-13 sind die zwei *Blinn Objekte* unter Verwendung des Werkzeugs durch *JavaView* graphisch dargestellt. Man kann durch die graphische Darstellung in Abbildung 4-13 er-

kennen, wie sich die Abweichung, die durch die Approximation der Gaußfunktion entsteht, auf die äußere Gestalt der *Blobby Surface* auswirkt.



Berechnung durch g	Approximation durch p

Abbildung 4-13: Blobby Surface (links) angenähert durch ein Polynom (rechts)

5 Das Werkzeug JAMIN

5.1 Kristallgeometrische Daten aus Webseiten und externen Tools

Im dritten Kapitel wurden unterschiedliche Beschreibungsmöglichkeiten für Kristallformen vorgestellt. Entsprechend diesen Möglichkeiten liegen die geometrischen Daten über Kristalle auf verschiedenen Webseiten in unterschiedlichen Beschreibungen vor. Die Tabelle 5-1 zeigt eine Übersicht von Webseiten, welche Kristallformen veröffentlichen, die durch Netzebenen und durch kartesische Koordinaten (blau hervorgehoben) beschrieben sind.

Webseite	Autor
http://www.crystalgrowing.com	Udo J. A. Behner
http://www.webmineral.com	David Barthelmy
http://database.iem.ac.ru/mincryst/	Anatoly V. Chichagov, Dmitry A. Varlamov (unterstützt von der Russian Foundation of Basic Research)
http://www.geosystems.no	The Geosystems Company (VRML V1.0 ascii)
http://www.iumsc.indiana.edu/database/	Indiana University Molecular Structure Center

Tabelle 5-1: Webseiten, die kristallgeometrische Daten veröffentlichen

Es gibt auch Programme, die nach benutzerdefinierten Vorgaben Kristallformen erzeugen, graphisch darstellen und in eine Datei abspeichern. In den meisten Fällen liegen kristallgeometrische Daten entsprechend der Beschreibung durch Netzebenen vor. Die benutzerdefinierten Vorgaben zur Erzeugung von Kristallformen sind bei den Programmen *Faces*, *WinXMorph* und *JCrystal* eine Menge L, deren Elemente Paare sind, die aus Miller'schen Indizes (hkl) bestehen und einer reellen Zahl, die den Abstand der durch (hkl) beschriebene Netzebene vom Symmetriezentrum angibt:

$$L \subset M \times \mathbb{R},\ M = \left\{ (h_i, k_i, l_i) \middle|\ i = 1, 2, \ldots \right\}$$

Damit lassen sich zum Beispiel die auf der Webseite von David Barthelmy zu einem Kristall angegebenen Miller'schen Indizes mit den Ebenenabständen als Eingabedaten für die Programme

Faces, *WinXMorph* und *JCrystal* verwenden. Diese Programme stellen die Kristallform graphisch dar und ermöglichen das Speichern der polygonalen Darstellung dieser Geometrie in eine Ausgabedatei. Die Ausgabedatei speichert also die geometrische und topologische Information der Kristallform. Das Programm *Faces* erstellt dazu eine GEO Datei. Das Format der in diesen Dateien gespeicherten Polygonmodelle ist nicht standardisiert und kann nur vom Programm *Faces* und von dem Werkzeug JAMIN, das im Rahmen dieser Arbeit entwickelt wurde, gelesen werden. Dagegen werden in die Projektdateien des Programms Faces unter anderem Miller'sche Indizes gespeichert. Diese Projektdateien lassen sich mit dem Programm JCrystal von Dr. Steffen Weber öffnen. Außerdem stellt JCrystal ungefähr 670 vorgefertigte Kristallformen zur Verfügung, welche in Textdateien durch Miller'sche Indizes beschrieben sind, die auf der Webseite von David Barthelmy veröffentlicht wurden. Das Programm WinXMorph von Prof. Dr. Werner Kaminsky befindet sich auf der Webseite der University of Washington, Department of Chemistry (Tabelle 5-2). Ähnlich wie bei JCrystal werden zur Beschreibung einer Kristallform die Miller'schen Indizes eingegeben. Mit den Programmen *JCrystal* und *WinXMorph* lassen sich Kristallformen im VRML V2.0 Format speichern. Im nächsten Kapitel werden einige Austauschformate für Polygonmodelle vorgestellt, die das Werkzeug verwendet.

Programm	Autor	Webseite
JCrystal	Steffen Weber	http://www.jcrystal.com
Faces	Georges Favreau	http://www.minerapole.com/a_/b_ofc27.html[8]
WinXMorph	Werner Kaminsky	http://cad4.cpac.washington.edu/winXmorphHome/winXmorph.htm
McMaille	B.M. Karuki	http://www.cristal.org/McMaille/
Shape	Shape Software	http://www.shapesoftware.com

Tabelle 5-2: Programme, die Kristallformen erzeugen und visualisieren

5.2 Austauschformate für dreidimensionale Kristallformen

Das Werkzeug JAMIN verwendet mehr als zehn verschiedene Austauschformate für Polygonmodelle. Die Zahl der verwendbaren Formate ist auf die Integration von *JavaView* zurückzuführen. Die Klassenbibliothek *JavaView* stellt Klassen für das Laden und Speichern der Polygonmodelle in diesen verschiedenen Formaten zur Verfügung. Dazu zählen unter anderem die folgenden Formate:

- Virtual Reality Modeling Language (VRML)
- Drawing Interchange File (DXF)
- Wavefront OBJ Format
- Brigham Young University Format (BYU)

[8] Das Programm FACES kann bei Herrn Favreau, dem Autor des Programms, persönlich via Email (favreaug@aol.com oder georges.favreau@mail.titn.alcatel.fr) angefragt werden.

- Object File Format (OFF)
- *JavaView* XML (JVX) Format

Die zwei zuletzt genannten Formate, OOF und JVX werden gegenüber den anderen genannten Formaten für die Beschreibung der Polygonmodelle häufiger verwendet, da sie intuitiver verständlich sind. VRML ist eine Beschreibungssprache für dreidimensionale Szenen. In VRML-Dokumenten wird jeweils ein Szenengraph beschrieben. Sie können als Austauschformat für die Kristallformen ausgeklammert werden.

5.2.1 Das OFF - Object File Format

OFF wurde im Jahre 1986 bei der *Digital Equipment Corporation's Workstation Systems Engineering* für den Austausch und die Archivierung von 3D Objekten entwickelt. In diesem Format lassen sich Polygonmodelle als Eckenliste beschreiben. Im Aufbau einer OFF Datei findet man folglich die topologische Datenstruktur Eckenliste wieder. Die OFF Datei enthält *n* Zeilen mit den Koordinaten der *n* Eckpunkte. Die erste Zeile, die Eckpunktkoordinaten enthält, beinhaltet die Koordinaten des Eckpunktes mit dem Index 1. Der Eckpunkt, dessen Koordinaten die nächste Zeile enthält, hat den Index 2 und so weiter. Nachdem alle Eckpunktkoordinaten zeilenweise aufgelistet sind, folgen Zeilen, welche die Indizes der Eckpunkte angeben, die zu einem Polygon gehören. Zeilen, in denen diese Indizes angegeben werden, beginnen mit der Anzahl der Eckpunkte des Polygons. Eine OFF Datei gibt in der obersten Zeile die Anzahl der Punkte, Polygone und Kanten an. Das untenstehende Beispiel aus [Rost89], ist ein Würfel im *Object File Format*.

```
 8       6       12
-1.0    -1.0     1.0
-1.0     1.0     1.0
 1.0     1.0     1.0
 1.0    -1.0     1.0
-1.0    -1.0    -1.0
-1.0     1.0    -1.0
 1.0     1.0    -1.0
 1.0    -1.0    -1.0
 4       1       2       3       4
 4       5       6       2       1
 4       3       2       6       7
 4       3       7       8       4
 4       1       4       8       5
 4       8       7       6       5
```

5.2.2 Das JVX Format

Das *JavaView* XML Dateiformat ist wie der Name schon sagt ein XML Format. Es dient der plattformunabhängigen und applikationsunabhängigen Speicherung von geometrischen Daten [PKPR02]. Die Tatsache, dass das JVX Format auf XML basiert, erlaubt die automatische Validierung der syntaktischen und semantischen Korrektheit einer beliebigen JVX Datei.

Das *JavaView* XML Format wird auch für den Austausch von geometrischen Daten zwischen *JavaView* und der Software von Drittanbietern verwendet. Beispiele hierfür sind die Software Maple und Polymake [PKPR02]. Eine JVX Datei beinhaltet die Informationen des 3D Modells (Polygonmodells), der Geometrie in seiner diskreten Darstellung. Neben diesen Informationen beinhaltet eine JVX Datei optionale Informationen über beispielsweise Autor und Erstellungsprogramm. So kann jede JVX Datei eine Beschreibung des in dieser Datei gespeicherten 3D Modells oder Informationen über den Autor der JVX Datei und das Programm, mit welchem die-

se JVX Datei erstellt wurde enthalten. In Abbildung 5-1 ist die Struktur der 3D Modelle im *JavaView* XML Format dargestellt. Wie bereits in Kapitel 2.3 erwähnt, enthält das 3D Modell die geometrische und die topologische Information über die Geometrie. Dementsprechend beinhaltet eine JVX Datei die geometrische und die topologische Information der Geometrie, die in der JVX Datei gespeichert ist (Abbildung 5-1).

Jedes 3D Modell im *JavaView* XML Format enthält zunächst Eckeninformationen in Form von Ortsvektoren. Die Form der topologischen Information der einzelnen Geometrie kann variieren. Es gibt JVX Modelle, die zusätzlich zu den Ecken auch Kanten speichern. Häufiger werden jedoch JVX Modelle verwendet, die neben den Ecken die Außenflächen also ein Polygonmodell speichern. Ein entscheidender Vorteil der zuletzt genannten JVX Modelle ist die Möglichkeit der Überprüfung, welche Flächen sichtbar und welche durch andere Flächen verdeckt sind. Zu jeder Fläche, zu jeder Kante und zu jedem Punkt können weitere Informationen wie beispielsweise eine Farbe oder eine Normale angegeben werden.

Eine JVX Datei kann auch mehrere Geometrien (in ihrer diskreten Darstellung) enthalten. Dies ist Voraussetzung, wenn man Teile des 3D Modells für ein Mineral unterschiedlich texturieren will. Beispiele dafür sind die in Kapitel 4.1 vorgestellten Modelle für Steinschnitte und Steinscheiben. Die Schnittfläche und die Steinhülle in dem 3D Modell für einen solchen Steinschnitt sind separate Geometrien und werden auch als solche in eine JVX Datei gespeichert.

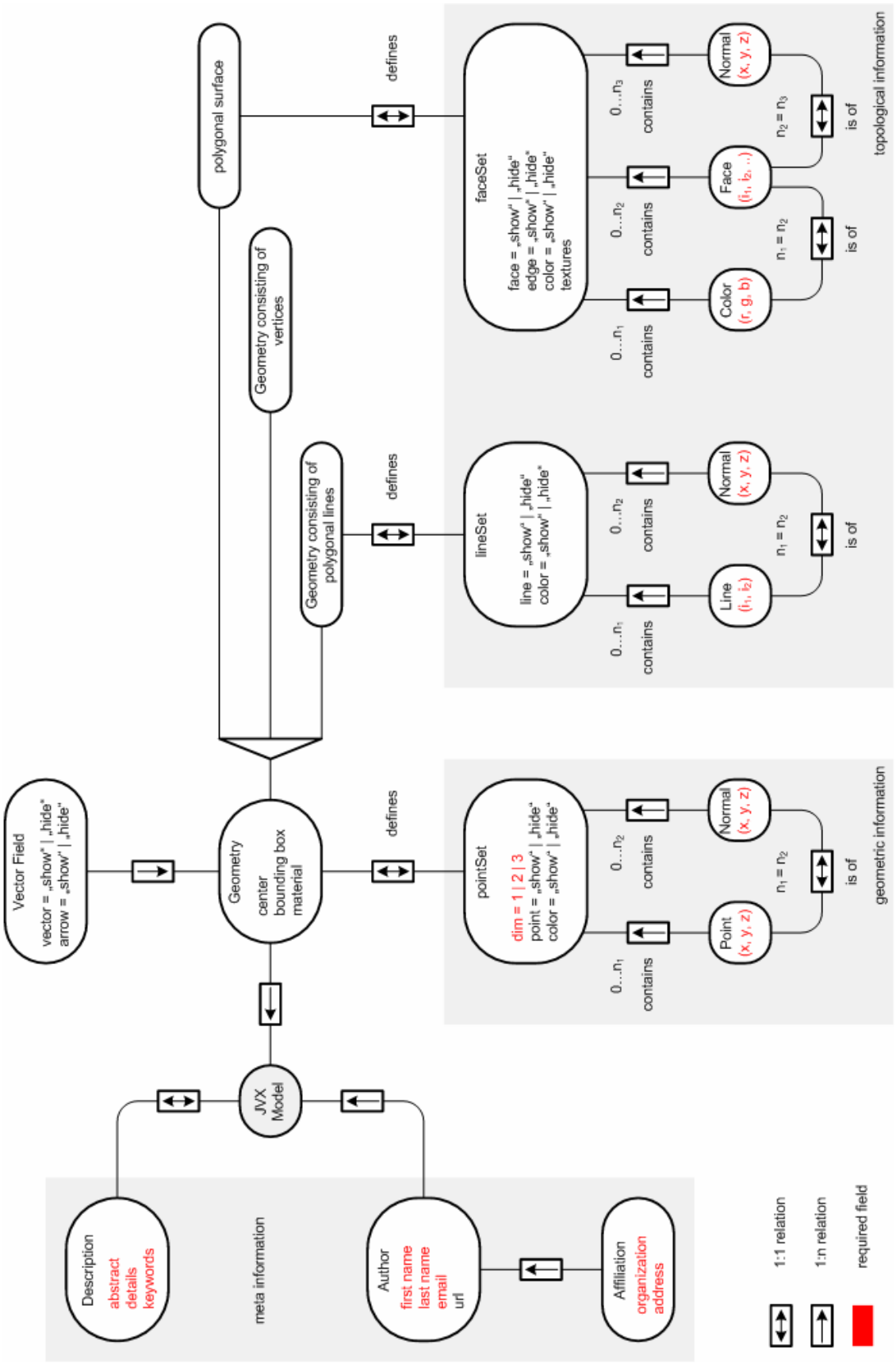


Abbildung 5-1: Wertestruktur von JVX Modellen

5.3 Bilddaten aus Galerien im Web

Möchte man die Kristalle und Steine graphisch realistischer darstellen, so muss man für die Kristall- beziehungsweise Steinformen Bilder als Textur verwenden. In Tabelle 5-3 sind Webseiten aufgelistet, welche Bilder veröffentlichen, die als Texturgraphiken für Kristall- und Steinformen geeignet sind. Geeignete Texturgraphiken sind Ausschnitte von Bildern, welche deutlich die Farbe und Struktur der Kristalloberfläche illustrieren.

Webseite
http://www.noctua-graphics.de
http://www.granat.at
http://www.freefoto.com
http://www.foto123.de

Tabelle 5-3: Webseiten, die Bilder von Mineralien und Kristallen veröffentlichen

Die Abbildung 5-2 zeigt einige der Beispieltexturen für Steine, die auf der *Noctua Graphics* Webseite zur Verfügung stehen. Sämtliche auf der *Noctua Graphics* Webseite verfügbaren Texturen für Steine sind nahtlos und liegen jeweils in der Größe 512×512 Pixel oder 1024×1024 Pixel vor.

Abbildung 5-2: *Noctua Graphics* Beispieltexturen

In Abbildung 5-3 ist eine Graphik (rechts) dargestellt, die beispielhaft für eine Gruppe von Quarzkristallen angewendet wurde (links). Die Texturgraphik zeigt den Ausschnitt aus der Oberfläche eines Quarzgesteins. Die graphische Darstellung der Gruppe von Quarzkristallen in Abbildung 5-3 (links) wurde mit JAMIN, dem im Rahmen dieser Arbeit entwickelten Werkzeug erstellt.

In JAMIN sind verschiedene Texturprojektionsverfahren implementiert, mit Hilfe deren die Texturgraphik auf ein 3D Modell projiziert wird.

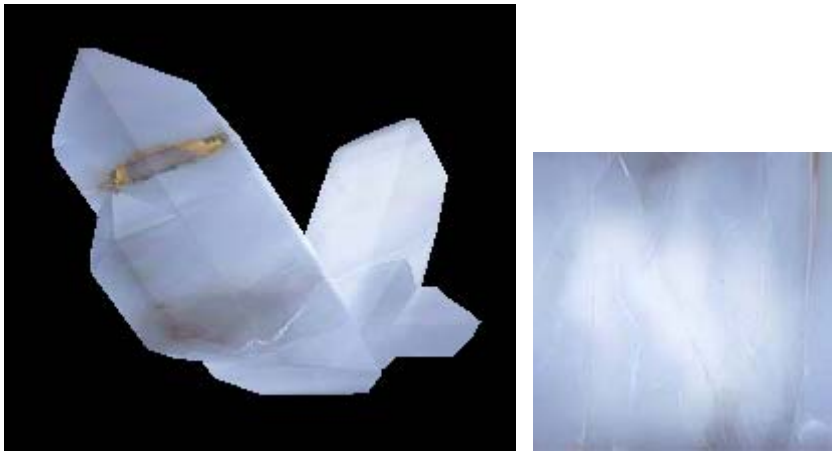


Abbildung 5-3: Gruppe von Quarzkristallen mit Textur

5.4 Die Klassenbibliotheken JavaView, Idx3d und JAMIN

Das Werkzeug JAMIN besteht aus mehreren Teilsystemen denen unterschiedliche Funktionen zugeordnet sind. Ein Teilsystem dient zum Beispiel ausschließlich der Visualisierung. Für die Visualisierung wurden die Klassenbibliotheken *JavaView* und *Idx3d* integriert.

Im Folgenden werden die zwei Klassenbibliotheken, *JavaView* und *Idx3d* vorgestellt. Hierfür soll der Aufbau beider Klassenbibliotheken in Form von UML Klassendiagrammen illustriert werden. Weiter wird beschrieben, wie *JavaView* und *Idx3d* in das Werkzeug JAMIN integriert sind. Es wird auch gezeigt, wie die Klassenbibliothek *JavaView* verwendet wird und was für deren Integration zu berücksichtigen ist. Im letzten Teil dieses Kapitels wird zu der Beschreibung des Werkzeugs JAMIN ein Verwendungsbeispiel gegeben.

5.4.1 Die Klassenbibliothek JavaView

Die Klassenbibliothek *JavaView* wurde im Rahmen eines Projektes an der *Technischen Universität Berlin* entwickelt. Die erste Version von *JavaView* wurde im November 1999 herausgegeben. Derzeit verfügt die Klassenbibliothek über mehr als sechshundert Klassen. Der Funktionsumfang von *JavaView* ist entsprechend groß. So stellt *JavaView* zum Beispiel Klassen und Methoden zur Verfügung für:

- Die Triangulierung von Geometrien. Voraussetzung ist hierfür, dass diese Geometrien in einer polygonalen Darstellung vorliegen.
- Das Erzeugen der Polygonmodelle für analytische Flächen, deren Parameterform gegeben ist.
- Das beliebig mehrfache Anwenden von Subdivisionsverfahren auf Polygonmodelle.
- Das Importieren aus und das Exportieren in verschiedene Austauschformate. Einige diese Formate wurden bereits in Kapitel 5.2 genannt.

- Die Auswahl bestimmter Optionen für die Darstellung der dreidimensionalen Geometrien. Dazu gehören zum Beispiel die Einfärbung der Begrenzungsflächen in Abhängigkeit ihrer relativen Ausgangsposition zum Koordinatenursprung und das Hinzufügen von Lichtquellen.

Nachfolgend wird beschrieben, wie der Hauptteil der Klassenbibliothek *JavaView* aufgebaut ist und wie die Polygonmodelle in *JavaView* erstellt werden.

Der Aufbau des Hauptteils der Klassenbibliothek *JavaView* ist ähnlich dem Aufbau des *JavaView* XML Formats, das bereits in Kapitel 5.2.2 beschrieben wurde. Wie in dem UML Klassendiagramm in Abbildung 5-4 dargestellt erbt die Klasse *PgPointSet*, welche die geometrische Information des 3D Modells in Form von Vektoren (*PdVector*-Objekte) speichert, von der Klasse *PgGeometry*. Die Klasse *PgElementSet* erweitert die Klasse *PgPointSet*. Sie enthält neben der geometrischen Information auch die Polygone (*PiVector*-Objekte) des 3D Modells. Zusammenfassend kann man sagen, dass ein *PgElementSet*-Objekt das Polygonmodell einer Geometrie enthält. Die Klasse *PvDisplay* erbt von der AWT Klasse *Canvas* und dient ausschließlich der graphischen Darstellung der 3D Modelle.

In Kapitel 5.2.1 wurde ein Würfel im *Object File Format* angegeben. Der untenstehende Quellcode ist die Implementierung des Polygonmodells dieses Würfels.

```
PgElementSet myPgElementSet = new PgElementSet();

myPgElementSet.addVertex(new PdVector(-1, -1, 1));
myPgElementSet.addVertex(new PdVector(-1, 1, 1));
myPgElementSet.addVertex(new PdVector(1, 1, 1));
myPgElementSet.addVertex(new PdVector(1, -1, 1));
myPgElementSet.addVertex(new PdVector(-1, -1, -1));
myPgElementSet.addVertex(new PdVector(-1, 1, -1));
myPgElementSet.addVertex(new PdVector(1, 1, -1));
myPgElementSet.addVertex(new PdVector(1, -1, -1));

myPgElementSet.addElement(new PiVector(0, 1, 2, 3));
myPgElementSet.addElement(new PiVector(4, 5, 1, 0));
myPgElementSet.addElement(new PiVector(2, 1, 5, 6));
myPgElementSet.addElement(new PiVector(2, 6, 7, 3));
myPgElementSet.addElement(new PiVector(0, 3, 7, 4));
myPgElementSet.addElement(new PiVector(7, 6, 5, 4));
```

In der ersten Zeile wird ein *PgElementSet*-Objekt erzeugt. Anschließend werden diesem Objekt die einzelnen Eckpunkte hinzugefügt. In der Reihenfolge, wie die Eckpunkte hinzugefügt werden, wird diesen Eckpunkten jeweils ein Index zugeordnet. Der Eckpunkt, der zuerst hinzugefügt wird, erhält den Index Null, der zweite den Index Eins, der dritte den Index Zwei und so weiter. Anders als bei der Indizierung der Eckpunkte im *Object File Format*, ist hier der kleinste Index nicht Eins sondern Null. Im letzten Schritt werden dem *PgElementSet*-Objekt die Polygone als Quadrupel der Indizes der Eckpunkte hinzugefügt.

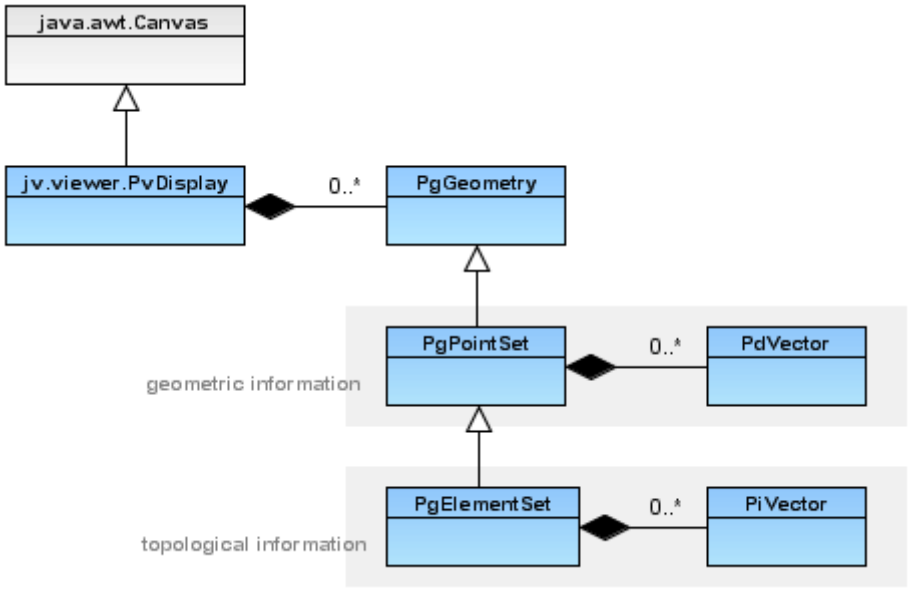


Abbildung 5-4: Klassenmodell der Hauptelemente von *JavaView*

Bei der Verwendung der Klassenbibliothek *JavaView* wird man mit einer großen Menge von Klassen und Methoden konfrontiert. Nachfolgend wird beschrieben, was bei der Integration von *JavaView* zu berücksichtigen ist und einige Richtlinien für die Programmierung mit *JavaView* vorgeschlagen:

- Wenn man die Anzahl der Punkte eines *PgElementSet* ermitteln will, sollte man den folgenden Aufruf vermeiden:

  ```
  int number = myPgElementSet.getVertices().length;
  ```

  da das *PgElementSet*-Objekt aus Effizienzgründen intern die Arraylänge für die Speicherung der Punkte vergrößert. Somit entspricht die Arraylänge für die Punkte des *PgElementSet* nicht der tatsächlichen Punktanzahl sondern ist größer als diese. Um die tatsächliche Punktanzahl eines *PgElementSet* zu ermitteln, verwendet man:

  ```
  int number = myPgElementSet.getNumVertices();
  ```

- Bei der Erstellung der Polygonmodelle mit Hilfe von Algorithmen höherer Komplexität sollte die Verwendung eines *PdVector*-Objektes als temporäre Variable vermieden werden. Bei Algorithmen mit hoher Schleifentiefe ist die Benutzung primitiver Datentypen gegenüber von beispielsweise *PdVector*-Objekten innerhalb der Schleifen speicher- und zeiteffizienter.

  ```
  for (int i = 0; i < 100; i++){
    for (int j = 0; j < 100; j++){
      double x = i;
      double y = j;
      z[i*100+j] = f(i,j);
    }
  }
  ```

In dem oben dargestellten Quellcodebeispiel wurde innerhalb der Schleifen statt der Erzeugung eines *PdVector*-Objektes Variablen vom Typ double verwendet.

- Erstellt man ein Polygonmodell mit sehr vielen Eckpunkten und Polygonen, so sollte man dem *PgElementSet*-Objekt die Eckpunkte und Polygone jeweils in Form eines Arrays übergeben:

```
PdVector[] vertices = …;
myPgElementSet.setNumVertices(vertices.length);
myPgElementSet.setVertices(vertices);

PiVector[] polygons = …;
myPgElementSet.setNumElements(polygons.length);
myPgElementSet.setElements(polygons);
```

  Damit wird der mehrfach wiederholte Aufruf für das Hinzufügen eines Eckpunktes

```
myPgElementSet.addVertex(new PdVector(x, y, z));
```

  beziehungsweise für das Hinzufügen eines Polygons

```
myPgElementSet.addElement(new PiVector(i, j, …));
```

  vermieden. Die Übergabe der Eckpunkte und Polygone jeweils in Form eines Arrays ist zeiteffizienter als das mehrfach wiederholte Hinzufügen eines einzelnen Eckpunktes beziehungsweise eines einzelnen Polygons.

5.4.2 Die Klassenbibliothek Idx3d

Idx3d ist eine mit Java 1.1 entwickelte Klassenbibliothek für 3D Echtzeitrendering Anwendungen. Sie wurde von Peter Walser entwickelt und liegt zurzeit in der dritten Version vor.

Ähnlich wie die Klasse *PvDisplay* in *JavaView* dient die Klasse *idx3d_Scene* in *Idx3d* der graphischen Darstellung von 3D Modellen. In *Idx3d* erfolgt die diskrete Darstellung einer Geometrie durch Dreiecke. Demnach werden wie auch in *JavaView* Polygonmodelle verwendet. Ein Polygonmodell ist in *Idx3d* durch die Instanz der Klasse *idx3d_Object* gegeben. Die Beschreibung eines Polygonmodells in *Idx3d* ist analog zu jener in *JavaView*. In *JavaView* ist ein Polygonmodell durch die Instanz der Klasse *PgElementSet* gegeben.

Die Abbildung 5-5 illustriert die vereinfachte Struktur der Klassenbibliothek *Idx3d* in Form eines UML Diagrams. Die Klassen für die Beschreibung eines Polygonmodells in *Idx3d* sind in Abbildung 5-5 hervorgehoben.

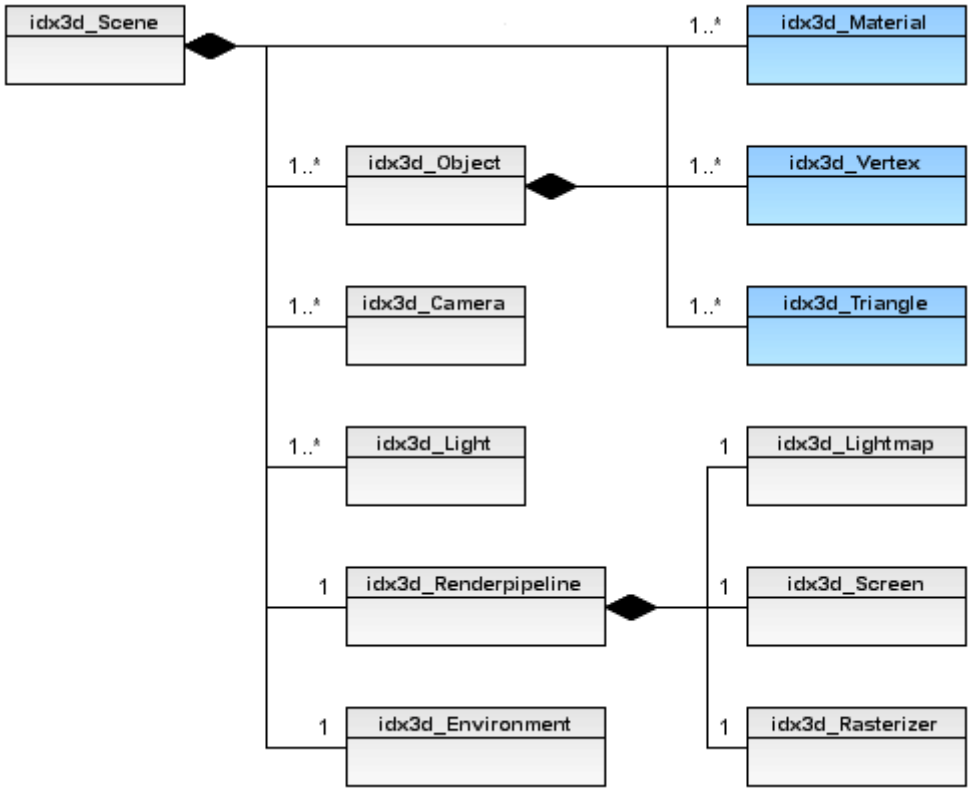


Abbildung 5-5: Klassenmodell der Hauptelemente von *Idx3d*[9]

5.4.3 Die Klassenbibliothek JAMIN

In Kapitel 3 und 4 wurden Methoden für die Erstellung von Kristall- und Steinformen vorgestellt. Diese wurden für das Werkzeug JAMIN implementiert. So existieren in JAMIN verschiedene Klassen für die Berechnung entsprechender Polygonmodelle. Im Folgenden wird die Klassenstruktur von JAMIN vereinfacht dargestellt. Es wird auch gezeigt, welche Klassen wie verwendet werden, um Polygonmodelle für Kristall- und Steinformen zu berechnen und zu visualisieren.

Die Klassenbibliothek JAMIN ist ein Werkzeug für die Entwicklung von 3D Anwendungen, welche vor allem Mineralien visualisieren. So verfügt JAMIN über eine Menge *Factory*-Klassen für die Erstellung von Kristall- und Steinformen. Wie in Abbildung 5-6 dargestellt, sind das zum Beispiel die Klassen *FractalFactory*, *CrystalFactory* und *StoneFactory* in dem Paket *jamin.math*. Verschiedene GUI Klassen wie beispielsweise die Klasse *GeometryCreationDialog* aus dem Paket *jamin.gui* verwendet diese *Factory*-Klassen und stellt deren Funktionen über graphische Benutzungsschnittstellen zur Verfügung. Das erstellte 3D Modell für die Kristall- oder Steinform liegt in einer polygonalen Darstellung als ein *PgElementSet*-Objekt oder als eine Menge von *PgElementSet*-Objekten vor. So liefert zum Beispiel die Berechnung eines 3D Modells für die in Kapitel 4.1 vorgestellte Steinscheibe durch JAMIN insgesamt drei *PgElementSet*-Objekte: zwei Schnittflächen und eine Mantelfläche.

[9] [Wals00]

Die Klasse *Canvas* aus dem Paket *jamin.gui* besitzt diese 3D Modelle und stellt sie graphisch dar. Es gibt mehrere Klassen, welche die Klasse *Canvas* erweitern und jeweils die 3D Modelle auf ihre Art visualisieren. Dazu gehören zum Beispiel die Klassen *CanvasIdx3d* und *CanvasJavaView*.

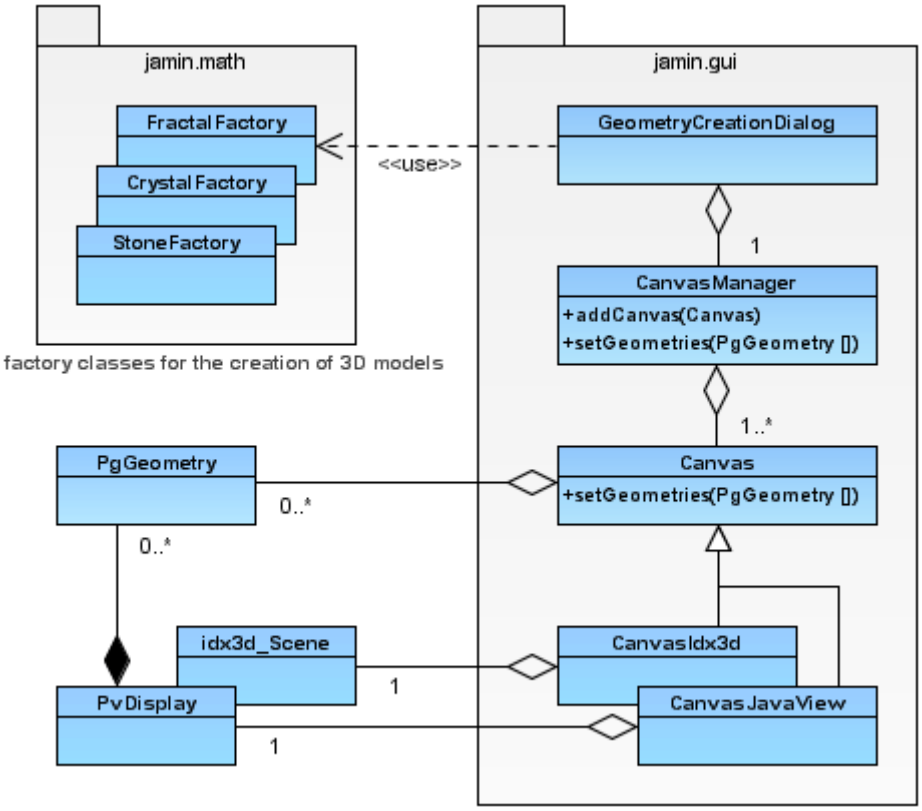


Abbildung 5-6: Die vereinfacht dargestellte Klassenstruktur von JAMIN

Im Folgenden wird gezeigt, wie die Klassenbibliothek JAMIN verwendet wird. Dazu soll ein konkretes Beispielprogramm dienen. Der untenstehende Quelltext ist der wesentliche Ausschnitt des Programms. Das Beispielprogramm erzeugt ein Polygonmodell des Quarzkristalls unter Verwendung der *TrigonalFactory* und visualisiert das Modell anschließend durch die *CanvasIdx3d*. Die Abbildung 5-7 zeigt einen Screenshot dieses Beispielprogramms.

```
import jamin.gui.CanvasManager;
import jamin.gui.Canvas;
import jamin.gui.CanvasIdx3d;
import jamin.math.crystalsystem.TrigonalFactory;
import …;

public static void main(String[] args){
 Canvas canvas = new CanvasIdx3d();
 CanvasManager canvasManager = new CanvasManager(canvas);
 canvasManager.setPgGeometry("quarz", TrigonalFactory.createQuarz());
 canvasManager.setTransparency(50);
 canvasManager.setEnvmap(new File("glass.jpg"));
 canvasManager.setTexture(new File("crystals.jpg"));
```

```
  JFrame frame = new JFrame("JAMIN: Beispiel mit Quarzkristall");
  frame.setSize(400,400);
  frame.getContentPane().setLayout(new BorderLayout());
  frame.getContentPane().add(canvas, BorderLayout.CENTER);
  frame.setVisible(true);
 }
```

In dem Grundlagenkapitel sind in Abbildung 2-1 Beispielkristalle zu jedem der sieben Kristallsysteme dargestellt. Die Klassenbibliothek JAMIN verfügt entsprechend über sieben Factory-Klassen für die Erzeugung der Kristallformen. In den Factory-Klassen sind jeweils einige bekannte Kristallformen der sieben Kristallsysteme wie beispielsweise Würfel, Oktaeder, Rhombendodekaeder und Korund vorgefertigt. 3D Modelle für seltenere Kristallformen können auf dieselbe Weise, wie in Kapitel 5.4.1 am Beispiel des Würfels gezeigt, manuell erstellt werden.

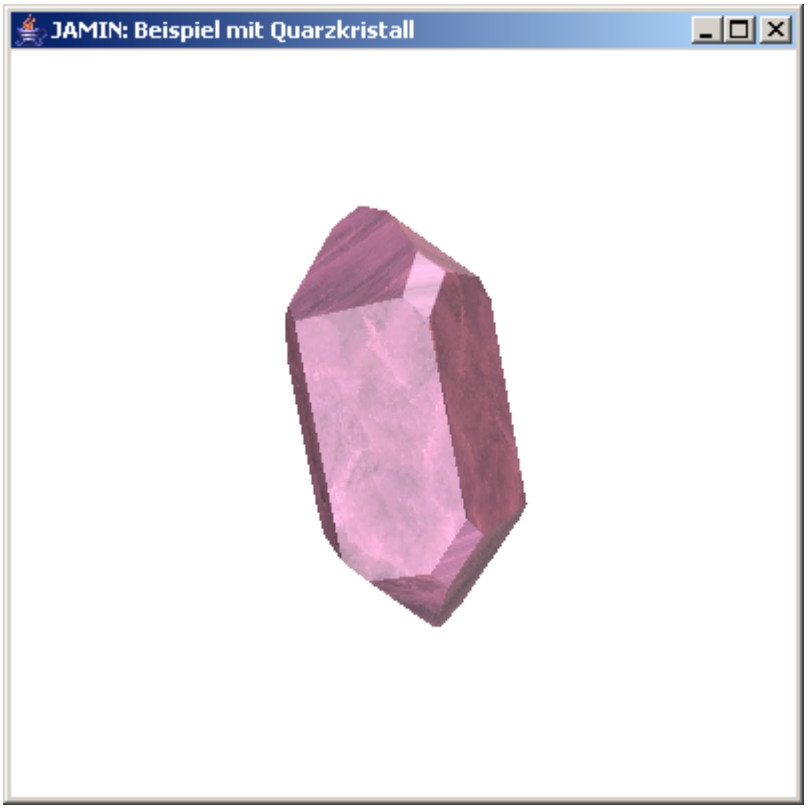


Abbildung 5-7: Screenshot eines JAMIN Beispielprogramms

6 Zusammenfassung und Ausblick

Das Ergebnis der vorliegenden Arbeit ist ein Werkzeug, das 3D Modelle für Mineralien erstellt und visualisiert. Es wurde in dieser Arbeit gezeigt, dass sich Mineralien mit dem Computer nachbilden lassen. Die 3D Modelle für Kristall- und Steinformen entsprechen weitgehend den originalen Kristall- und Steinformen.

Derzeit wird JAMIN um Möglichkeiten der graphischen Darstellung erweitert. Im speziellen wird eine weitere Klasse entwickelt, die von der Klasse *Canvas* aus dem *jamin.gui* Paket erbt. Diese Klasse (*CanvasOGL*) nutzt die OpenGL Schnittstelle und kann somit komplexere Modelle visualisieren, unter Einbeziehung von zum Beispiel dynamischen Lichtverhältnissen, verschiedenen Materialien und Darstellungsmodi wie beispielsweise *Cartoon* oder *Painterly*. Der Nachteil der Klasse *CanvasOGL* liegt in der Tatsache, dass sie nicht unmittelbar für die Visualisierung der 3D Modelle auf Webseiten verwendet werden kann.

Ideen der Weiterentwicklung sind die Entwicklung und Integration von Algorithmen für die Erstellung fraktaler Oberflächen. Zudem soll eine größere Bibliothek von Nachbildungen unterschiedlicher Mineralien erstellt werden. Demzufolge werden zusätzliche Polygonmodelle mit entsprechenden Texturen erstellt.

Die hier vorgestellten Methoden für die Modellierung von steinförmigen Oberflächen wurden in JAMIN umgesetzt, lassen sich jedoch auch auf andere Problemstellungen übertragen. So können mit Hilfe von Stream Bubbles geschlossene Oberflächen für die Modellierung von beispielsweise modernen Kunstgegenständen oder Figuren erstellt werden.

Quellenverzeichnis

[ACSE02] AKLEMAN, ERGUN; CHEN, JIANER; SRINIVASAN, VINOD; ERYOLDAS, FUSUN: Handle Reconstruction by Using A New Corner Cutting Subdivision Scheme with Tension. http://www-viz.tamu.edu/faculty/ergun/research/topology/cat02/, Texas A&M University, 2002.

[Atesi04] ATESI, NOURIDA; NEUMANN, CHRISTINE; WAGNER, HERBERT; U. A.: Siemens in Aserbaidschan. Neue Brücke - Yeni Körpü, Deutsch-Aserbaidschanisches Journal, Ausgabe Nr. 2 - April 2004, Gencevi Institut für aserbaidschanische Kultur e. V.

[Baie00] RUDOLF H. BAIERL: Natürliche konvexe Polyeder, Nr.2000,1.

[Baie99] RUDOLF H. BAIERL: Konvexe Polyeder. Klassische und hybride numerische Generierungsmethoden, Nr.99,2.

[Baku00] BAKU CARTOGRAPHICAL FACTORY OF THE STATE COMMITTEE ON GEODESY AND CARTOGRAPHY OF THE AZERBAIJAN REPUBLIC: The Informative Cartographic Reference Book of Azerbaijan. ZAMAN-i Company, Baku, 1999-2000.

[Hecht05] HECHT, FLORIAN: Level-of-Detail Verfahren. Ausarbeitung im Rahmen des Seminars “Modellierung”, 2005. Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik, Universität Karlsruhe.

[Hild02] HILDRETH, STEPHEN: Introduction to Crystallography and Mineral Crystal Systems. http://www.geoclassroom.com/mineralogy/xlexercise.shtml, 1996-2002, University of South Dakota.

[HoLa92] HOSCHEK, JOSEF; LASSER, DIETER: Grundlagen der geometrischen Datenverarbeitung. 2., neu bearbeitete und erweiterte Auflage. B. G. Teubner Stuttgart 1992.

[IICS05] THE INTERNATIONAL INSTITUTE FOR CASPIAN STUDIES (IICS): http://www.caspianstudies.com/basicinfo/basic-info-azerbaijan.htm, aktualisiert am 25. September 2005.

[Kilt03] STEVEN L. KILTHAU: Weak Refinement for Uniform Subdivision. Master's Thesis in Computer Science, Simon Fraser University, April 2003.

[Lee00] LEE, AARON: Building your own Subdivision Surfaces. Gama Network, http://www.gamasutra.com/features/20000908/lee_01.htm, 8. September 2000.

[MBFM99] MÄURERS; BAUFELD; FRIEDRICH; MÜLLER; WABNITZ; MÜHLE: Java - Das Grundlagenbuch. 1. Auflage. DATA BECKER GmbH & Co. KG, Düsseldorf 1999.

[Natu04] JAKOBER, NORBERT; KUGI, KLAUS: Der große Naturführer - Mineralien und Edelsteine. Neuer Kaiser Verlag Gesellschaft mbH. Klagenfurt 2004.

[PKPR02] POLTHIER, KONRAD; KHADEM, SAMY; PREUß, EIKE; REITEBUCH, ULRICH: Publication of Interactive Visualizations with JavaView. Erschienen in: J. Borwein, M. Morales, K. Polthier, J.F. Rodrigues: Multimedia Tools for Communicating Mathematics, Springer Verlag 2002.

[Rost89] RANDI J. ROST: OFF - A 3D Object File Format. Digital Equipment Corporation, Workstation Systems Engineering. Palo Alto, 6. November 1986, aktualisiert am 12. Oktober 1989

[Sult04] SULTANOW, E.: Implizite Flächen. Potsdam 2004.

[TuBr02] GREG TURK, JAMES F. O'BRIEN: Modelling with Implicit Surfaces that Interpolate. ACM Transactions on Graphics, Vol 21, No 4, Oktober 2002.

[Wagn17] WAGNER, P.: Lehrbuch der Geologie und Mineralogie. Verlag von B.G. Teubner, Leipzig und Berlin 1917.

[Wals00] WALSER, PETER: Idx3d Realtime 3D Engine. http://www.idx3d.ch/, 1999-2000.

[Weis05] ERIC W. WEISSTEIN: MathWorld - the web's most extensive mathematics resource. http://mathworld.wolfram.com/, Oktober 2005.

[ZhPa00] ZHANG, BING; PANG, ALEX: NURBS Blobs for Flow Visualization. Jack Baskin School of Engineering, University of California, Santa Cruz, 27 November 2000.

Anhang

Anlage 1: Finden von Mineralien in Dashkesan

In den Kapiteln 2 bis 5 wurde gezeigt, wie sich Mineralien mit dem Computer nachbilden lassen. In diesem Kapitel wird eine Region vorgestellt, die einerseits reich an natürlichen Vorkommen von Mineralien und Kristallen ist und andererseits aber nur wenigen Spezialisten bekannt ist.

Nachfolgend werden beispielhafte Vorkommen von Mineralien aus dem kaukasischen Gebirge in Aserbaidschan vorgestellt. Die aserbaidschanische Republik mit einer gesamten Flächenausdehnung von etwa $86.600 km^2$ [IICS05] grenzt im Norden an die Russische Föderation, im Nordwesten an Georgien, im Westen an Armenien, im Süden an den Iran und im Osten an das kaspische Meer (Abbildung 6-1). Das Gebirge, welches im Westen Aserbaidschans von der armenischen Ostgrenze durchdrungen wird, heißt *kleiner Kaukasus*. Das Gebirge im Nordosten Aserbaidschans wird als *großer Kaukasus* bezeichnet. Das Gebiet um und zwischen den zwei Städten *Gäncä* und *Dashkesan* nahe des *kleinen Kaukasus* (Abbildung 6-1) verfügt über Baryt- und Achatvorkommen [Baku00].

Mineralien und Edelsteine gibt es an vielen Orten auf der Erde. Mineralanreicherungen, deren Abbau sich lohnt, heißen Lagerstätten. Demgegenüber werden die Stellen mit Einzelfunden als Fundort bezeichnet. Der übergeordnete Begriff für Lagerstätte und Fundort ist Vorkommen. Die Verteilung der Minerallagerstätten über die Erde ist nicht gleichmäßig. Zu den Regionen, in denen es viele Minerallagerstätten gibt, gehören das südliche Afrika, Süd- und Ostasien, der Ural sowie Australien, Brasilien und die Gebirgszonen der USA. Es gibt auch Minerallagerstätten und Fundstellen im kaukasischen Gebirge.

Dashkesan übersetzt aus dem Aserbaidschanischen bedeutet *Steinschneider*. Neben Eisenerz wurde in *Dashkesan* auch Kupfer und Kobalt entdeckt [Atesi04]. Zwischen 1860 und 1870 erwarb die deutsche Firma Siemens Produktionsstätten in *Dashkesan*. Von 1867 bis 1916 gewann Siemens jährlich etwa 70000 Puds Kobalt ($1\ pud = 16,3805\ kg$). Im Erzbergbau Aserbaidschans übernahm Siemens eine führende Rolle [Atesi04].

Abbildung 6-1: Topographische Karte von Aserbaidschan[10]

Dashkesan ist auch der Name der Region, in der sich die Stadt *Dashkesan* befindet. In dieser Region gibt es Magnetit-, Epidot- und Andraditvorkommen. In der Stadt *Dashkesan* ist eine Fundstelle für Kristalle, einschließlich des Calcits. Calcit in Form von körnig spätigen Massen, die durch Gesteinsumwandlung entstanden sind, wird auch als Marmor bezeichnet [Natu04]. Eine Fundstelle der aserbaidschanischen Stadt *Dashkesan*, wo unter anderem Calcit gefunden werden kann ist in Abbildung 6-2 dargestellt. Abbildung 6-3 zeigt ein dort gefundenes Exemplar für eine Menge von Calcitkristallen, die in einem Felsblock eingeschlossen sind.

[10] [IICS05]

Abbildung 6-2: Fundstelle für Calcit in *Dashkesan*

Von der in Abbildung 6-2 gezeigten Fundstelle aus kann man ein Teil der Stadt *Dashkesan* sehen. Das Gebirge im Hintergrund der Abbildung 6-2 ist ein Teil des *kleinen Kaukasus*.

Abbildung 6-3: Calcitkristalle, eingeschlossen in einem Felsblock

Im Internet gibt es zahlreiche Webseiten, die Informationen über Mineralien und über deren Fundorte wie beispielsweise über Dashkesan veröffentlichen. Ein Beispiel ist die Mineral-Datenbank *mindat.org*. Die dort hinterlegten Informationen werden häufig aktualisiert und um neue Informationen ergänzt. In Tabelle 6-1 sind Fotos von Mineralien aus Dashkesan einschließlich einer kurzen Beschreibung zu jedem Foto gegeben, die aus diesen Webseiten entnommen sind.

	Zwei 6,5cm × 6,5cm große Magnetit-Kristalle in Dodekaederform auf einer flachen Steinplatte. Dave Bunk Minerals http://davebunkminerals.com
	Vergrößerte Darstellung von miteinander verwachsenen Epidot-Kristallen. Mineralienatlas http://www.mineralienatlas.de
	Dunkelrote bis schwarze lichtdurchlässige Andradit-Kristalle in einer Größe bis zu 11mm. Mineral-Datenbank mindat.org http://www.mindat.org

Tabelle 6-1: In Webseiten veröffentlichte Mineralien aus Dashkesan