Fachhochschule Köln
University of Applied Sciences
Campus Gummersbach

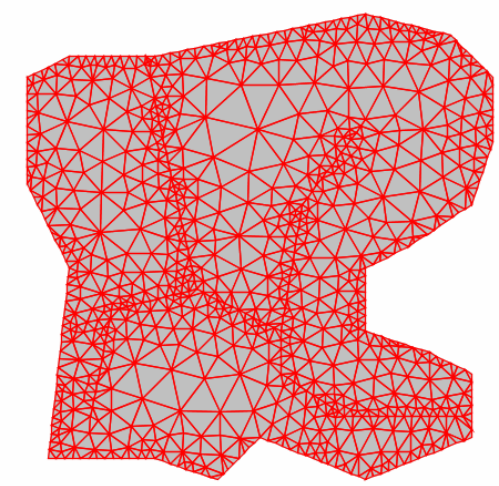
Fachbereich Informatik

Diplomarbeit

zur Erlangung des Diplomgrades

Diplom-Wirtschaftsinformatikerin (FH)

Modellierung raumbezogener Informationen mit Dreiecksnetzen



von Liane Ziethen-Aksoy

Weckenbergstraße 6
51643 Gummersbach
Matrikelnummer: 11024910

Erstprüfer: Prof. Dr.-Ing. Jackson Roehrig

Zweitprüfer: Prof. Dr. H. Faeskorn-Woyke

Juni 2005

Inhaltsverzeichnis

Teil III - Implementierung eines Netzgenerators

1 Vorwort

Die vorliegende Diplomarbeit entstand am Institut für Tropentechnologie der Fachhochschule Köln. Sie ist Teil eines Projektes zur Erstellung eines entscheidungsunterstützenden Systems in der Gewässerkunde (Hydrologie).

1.1 Hydrologie

Jegliches Leben auf der Erde ist auf das Vorhandensein von Wasser angewiesen. Schon seit Beginn der Menschheit war man in mancherlei Hinsicht vom Wasser abhängig: Trinkwasserbeschaffung für Mensch und Tier, Nahrungsversorgung durch Fischfang und Pflanzen, deren Wachstum erst das Wasser ermöglicht, Fortbewegung von Lasten und Menschen auf dem Wasser. Die heutige fortschreitende kulturelle und zivilisatorische Entwicklung brachte neue Abhängigkeiten: Gewerbe und Industrie benötigen Wasser für den Produktionsablauf, der Wasserbedarf der Haushalte ist mit der weitgehenden Verdrängung manueller Tätigkeiten durch Maschinen um ein Vielfaches gestiegen, Volkswirtschaft und Haushalte verlangen nach elektrischer Energie, die zum Teil der Wasserkraft entstammt; schließlich kann die sprunghaft wachsende Erdbevölkerung vor allem in den Entwicklungsländern nur mit Hilfe der künstlichen Bewässerung in ausreichendem Maße mit pflanzlichen Produkten versorgt werden (vgl. [Kind 03/2]).

Bevölkerungsdichte, Siedlungsstruktur und Formen der Wassernutzung beeinflussen die Wassernachfrage und stellen verschiedene Ansprüche hinsichtlich der Menge, der zeitlichen Verteilung und der Güte des Wassers. Es ist jedoch nicht mehr selbstverständlich, dass das Wasser für die Befriedigung der Bedürfnisse der Menschen auf der Erde in ausreichender Menge und Güte zur Verfügung steht.

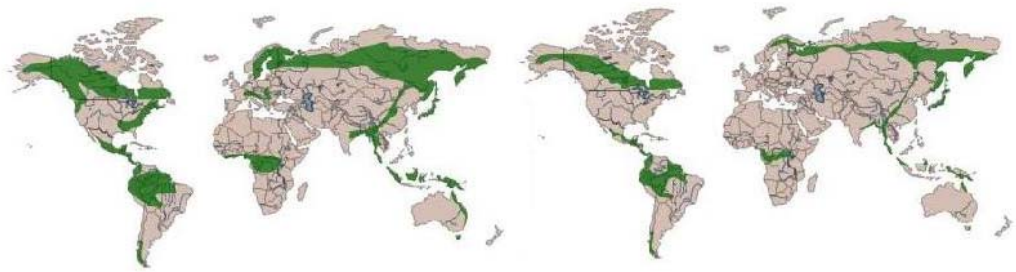


Abb. 1.1: Wasserknappheit: Waldbestand vor 8000 Jahren (li) -- heute (re), [Kind 03/1]

Der heutige Wassermangel hat seine tiefere Ursache in der dauernden Fehleinschätzung der Natur. Um diese besser zu verstehen und damit dem oben erläuterten Problem entgegenzuwirken, befasst sich *die physikalische Hydrologie* mit dem globalen und lokalen Wasserkreislauf einschließlich der zugrundeliegenden Naturprozesse. Durch Entwicklung von Modellen wird eine Gewinnung von Detailkenntnissen angestrebt. Der Bereich der *angewandten Hydrologie* hingegen beschäftigt sich mit der Lösung wasserwirtschaftlicher Probleme - der Nutzung des Wassers durch den Menschen.

Der Einsatz eines *Geografischen Informationssystems (GIS)* kann in der Hydrologie durch Visualisierung gesammelter Erfahrungswerte bereits eine Erleichterung der Entscheidungsfindung bei speziellen Problemstellungen bewirken. In einem GIS werden meist Funktionen vereinigt, die Methoden unterschiedlicher Systeme sind, wie z.B. Möglichkeiten statistischer Berechnungen, Bearbeitungen von Geometriedaten, Verwaltung von großen Datenbeständen, Darstellung von Karten. Weitere Aspekte eines GIS sind Geometrische Analysen, Flächenverschneidung, Möglichkeiten der Verknüpfung verschiedener Abfragen, Kombination verschiedener Datentypen und Geometrien etc. (vgl. [fh-mainz 05/01]).

Diese Aspekte bilden die Grundlage zum Aufbau eines interaktiven räumlichen Informationssystems. Zur Abrundung eines solchen Systems sind jedoch meist weitere Auswertungen und Analysen des Datenmaterials sowie eine Modellbildung nötig, um konkrete, kompetente Entscheidungen über Eingriffe in die Natur unter Berücksichtigung ihrer Auswirkungen zu treffen.

1.2 Entwicklungsprojekt STDSS

Die Anwendung *Spatial Temporal Decision Support Systeme* (STDSS) ist ein Entwicklungsprojekt des Instituts für Tropen Technologie der Fachhochschule Köln. Sie ist für einen spezifischen Aufgabenbereich der Hydrologie konzipiert, im Hinblick auf über- und unterirdisches Wasser sowie Unterteilung in Raum und Zeit. STDSS soll GIS-Komponenten mit Eigenschaften eines *Decision Support System* (DSS) für zeitliche und räumliche Analysen und Modellbildung sowie Ergebnisvisualisierung in einem System vereinigen.

Beispielsweise können Daten, die für unterschiedliche Aufgabenstellungen auswählbar sind, durch verschiedene kognitive Herangehensweisen ausgewertet und zur Modellbildung herangezogen werden. Eine Möglichkeit der Modellierung stellt die Dreiecksnetzgenerierung dar.

STDSS ist in der Programmiersprache *Java* Version 1.4. realisiert und implementiert.

1.3 Danksagung

Herrn Dipl.-Inform. Peter Wagner danke ich für das Zur-Verfügung-Stellen eines Arbeitsplatzes im Labor für Mathematik und seine Anwendungen des Instituts für Informatik am Campus Gummersbach. Weiterhin danke ich Herrn Dipl.-Ing. Christiano Almeida. Er brachte mir den praktischen Umgang mit geografischen Daten näher und gab mir Anregungen und Hinweise zu Verbesserungen.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Jackson Roehrig, der mir als Betreuer immer mit Rat und Tat zur Seite stand und auch Frau Prof. Dr. Heide Faeskorn-Woyke, die mich als Zweitprüfer unterstützt hat.

Zuletzt möchte ich noch meiner Familie für ihre Unterstützung danken.

Teil I

Einführung in die Thematik

2 Einleitung

2.1 Problemstellung

Zur Lösung hydrologischer Problemstellungen tragen oft numerische Berechnungen und Simulationen bei. Geometrische Daten bilden dafür die Grundlage. Nach Burkhard geht diesen Berechnungen und Simulationen immer erst eine Modellbildung voraus (vgl. [Burg 05/1]). Die Erstellung eines Modells unterteilt sich in mehrere Teile. Ausgegangen wird von einer Beschreibung des Problems in Form von Zeichnungen, Daten aus der rechnerunterstützten Konstruktion (CAD[1]), Messgrößen oder Sondierungsdaten. Kontinuierliche Problemen machen eine Aufteilung des Gebietes in kleine Elemente (Diskretisierung) erforderlich. Bei einfachen Problemstellungen ist eine manuelle Diskretisierung möglich, während bei komplexeren zwei- und dreidimensionalen sowie möglicherweise zeitabhängigen Systemen eine automatische Vernetzung erforderlich wird. Eine Vielzahl von Algorithmen und unterschiedlichen Ansätzen, die es inzwischen für eine Vernetzung gibt, macht es schwierig, das geeignete Verfahren für die spezielle Anwendung zu finden. Zusätzlich erschwert die Wahl einer geeigneten programmtechnischen Umsetzung in Bezug auf die Programmiersprache die Realisierung dieses Netzgenerators. Standardisierung und Bibliothekslizenzen können Kriterien für eine Wahl der Programmiersprache sein.

2.2 Ziel der Diplomarbeit

Ziel dieser Diplomarbeit ist ein geeignetes Verfahren zu entwickeln, Daten komplexer zweidimensionaler Gebiete in entsprechender Qualität weiteren Anwendungen des STDSS in Form von Dreiecksnetzen zur Verfügung zu stellen. Die zu vernetzenden Gebiete sind zumeist komplexe Wassereinzugsgebiete. Die sich dadurch ergebende Problemstellung ist durch geeignete Wahl der Datenstrukturen und der Algorithmen zur Raumaufteilung und Vernetzung zu lösen. Auf dieser Grundlage soll eine automatische Dreiecksnetzgenerierung realisiert werden. Die Wahl der Programmiersprache für die Implementierung des Netzgenerators bezieht sich letztlich auf das Gesamtkonzept von STDSS, welches in der Programmiersprache *Java* realisiert wurde.

[1] CAD = Computer Aided Design

Die Diplomarbeit ist in drei Teile gegliedert. Der erste Teil dient der Definition von Grundbegriffen zur Verarbeitung von raumbezogenen Informationen.
So werden im Kapitel 3 die Begriffe Geoinformationssystem, Decision Support System erklärt.

Der zweite Teil beschäftigt sich mit den Grundlagen der Netzgenerierung.

Kapitel 4 erläutert die Integration von räumlichen Daten in ein GIS. Der Stellenwert dieser Daten wird herausgestellt. Im Anschluss wird ein Überblick über Datenformate und Datenstrukturen gegeben, die bei der Vernetzung Anwendung finden.

Die Aufteilung des Raumes bzw. des zu vernetzenden Gebietes nimmt eine wichtige Position bei der Vernetzung ein. In Kapitel 5 werden einige Verfahren und ihre Algorithmen zur räumlichen Indizierung des Raumes vorgestellt. Diese sind unter anderem die Gittergenerierung und das Quadtree-Verfahren.

Verfahren für die Dreiecksnetzgenerierung im zweidimensionalen Raum stehen im Vordergrund von Kapitel 6. Hier werden die direkten Grundlagen der Dreiecksnetzgenerierung erläutert. Dazu gehören die Qualitätsanforderungen für eine Vernetzung sowie die Variantenvielfalt der Algorithmen für die Generierung.

Der dritte Teil zeigt die praktische Umsetzung der Dreiecksnetzgenerierung. Die Funktionsweise eines Netzgenerators für komplexe Gebiete wird in Kapitel 7 dargestellt. Ein erster Überblick wird durch einen Prozeßablaufplan gegeben. Danach werden einzelne implementierte Methoden und Berechnungen eingehender erklärt. Am Ende erfolgt schließlich eine Ergebnisdarstellung.

3 Grundlagen der Verarbeitung raumbezogener Informationen

3.1 Geoinformationssysteme

Karten und Pläne sind seit den Anfängen der Menschheit eine wichtige Orientierungshilfe. In jeglicher Form vermitteln Karten und Pläne ein Bild unserer Umwelt. Topographische Merkmale, Gebäude und Verkehrswege stellen ein Bezugssystem dar, welches uns eine Orientierung erleichtert (vgl. [Bil 91/1], S. 1 ff.).

Ab dem 19. Jahrhundert wurden in Europa analoge (geografische oder raumbezogene) Informationssysteme flächenhaft in Form von Karten und Buchwerken aufgebaut. Heute bedingt die Interdisziplinarität von globalen und lokalen Fragestellungen wie z.B. im Umweltschutz einen schnellen Datenaustausch und damit Vernetzbarkeit, so dass die Umstellung der schon bestehenden, mehr oder weniger vollständigen analogen Informationssysteme auf die Methoden der elektronischen Datenverarbeitung eine unabdingbare Notwendigkeit darstellt (vgl. [Bil 91/1], S. 1 ff.).

Der Begriff GIS stammt aus dem anglo-amerikanischen Sprachraum und steht dort für G*eographic Information System*. Dies bedeutet wörtlich übersetzt *Geografisches Informationssystem*. Nach Bill (vgl. [Bil 91/1], S. 1 ff.) wurde dieser Begriff bereits 1963 von R.F. Tomlinson bei der Einrichtung eines rechnergestützten raumbezogenen Informationssystems in Kanada eingeführt und es erfolgte damit erstmals der Hinweis auf eine neue Technologie – nämlich den Einsatz der elektronischen Datenverarbeitung in der raumbezogenen Datenhaltung.

Im deutschsprachigen Raum etablierte sich der Begriff *Geoinformationssystem* – nachfolgend GIS genannt.

Neben dem Erzeugen von digitalen Karten-Varianten ist auch das langfristige Ablegen der Inhalte als Daten in einem Informationssystem entscheidend. Sie können mit anderen Daten zu neuen Kombinationen, Auswertungen und Vergleichen herangezogen werden. Zum Aspekt der Darstellung kommen somit die langfristig viel bedeutsameren Aspekte der Verfügbarkeit und Konsistenz der Informationen eines solchen Systems hinzu (vgl. [Bil 91/1], S. 1 ff.).

Die Funktionen eines GIS müssen beiden Aspekten Rechnung tragen:

Nach innen hin muss Konsistenz gewahrt werden und nach außen muss sich ein GIS nach jener Vorgehensweise richten, die bei der manuellen Erstellung und Auswertung einer Karte beachtet werden soll.

Aus der Vielzahl der Darstellungs- und Interpretationsmöglichkeiten werden jene ausgewählt, die der jeweils zugrunde liegenden Thematik am besten entsprechen. Beispiele dafür sind Stadtpläne, Wanderkarten, die Wetterkarte im Fernsehen, die grafische Wiedergabe von Gewässergütezonen. Diese werden als Kriterium für die Art der Informationen, das Maß der Genauigkeit und für das Spektrum von möglichen Auswertungen verwendet.

Entwicklung der GIS-Technologie

Tabelle 1 gibt einen Überblick über die Entwicklungsschritte von Geo-Informationssystemen. Heute sind GIS im Umgang mit raumbezogenen Daten nicht mehr wegzudenken. Darum ist eine ständige, unablässige Weiterentwicklung notwendig.

Zeit	Grafik	Geometrie	Geoinformationssysteme
1950 - 1960	Vektor- oder Liniengrafik	Einfache goemetrische Algorithmen, Drahtmodelle	Pilotanwendungen der digitalen Kartierung, Entwicklung digitaler Geländemodelle am Massachusetts Institute of Technology (MIT)
1960 - 1970	Interaktive Computergrafik	Approximationsmethoden für Kurven und Flächen, Entwicklung geometrischer Programmiersprachen	Anwendungen der digitalen Bildverarbeitung, Anwendung digitaler Geländemodelle, Aufbau des Kanadischen GIS
1970 - 1980	Rastergrafik, Standards, Animation, Computerspiele	Eindeutige Darstellung räumlicher Objekte, Komplexitätsbetrachtungen von geometrischen Algorithmen	Aufbau von Landinformationssystemen, Einleitung der ALK, CAD-Kartographie, Digitale Photogrammetrie
1980 - 1990	Kognitive Computergrafik, Bewegung, Computervision	Geometrische Daten- und Methodenbanken, logistische Systeme, Standardisierung, raumbezogene Zugriffsmethoden	Aufbau von Netz-, Raum- und Umweltinformationssystemen, verstärkter Ausbau von LIS (ALK, ATKIS, TOPIS)
1990 - 2000	Hochleistungs-Computergrafik, Ikonische Benutzeroberflächen	Fortsetzung der Standardisierung, wissensbasierte Systeme, Objektorientierte Datenbanken	Hybride Geoinformationssysteme, Digitale photogrammetrische Arbeitsstationen, GPS-Datenerfassung

Tabelle 1: Entwicklung von Computergrafik, Algorithmen und GIS, [Bil 91/1]

Aufbau eines GIS

Ein GIS kann durch ein Vierkomponenten-Modell sowohl im Aufbau als auch in der Aufgabenbewältigung gekennzeichnet werden (vgl. [Bil, 1991/1]):

Definition: Ein GIS ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und grafisch präsentiert werden.

GIS ist also ein System, welches dem Anwender ermöglicht, mit Hilfe von Hardware und Software aus vorhandenen Daten Informationen zu gewinnen, aufzubereiten und zu analysieren. Anschließend wird durch ein integriertes Präsentationssystem die grafische Visualisierung der Informationen ermöglicht.

Nach Bill (vgl. [Bil 91/1], S. 52 ff. und [Bar 95/1], S. 14 ff.) stellt GIS ein logisches Gesamtkonzept dar, welches zu einer realen Arbeitsplatzkonfiguration zusammengefasst werden kann. Eine zentrale Rolle spielt hierbei ein grafisch-interaktiver Arbeitsplatz. Zum Hardware-Profil würden nicht nur ein Rechner mit Bildschirm, Tastatur und Maus gehören sondern auch Digitalisierungsgeräte, Scanner/Bildsensoren, Vermessungsgeräte, Drucker/Plotter und sekundäre, tertiäre Speichermedien sowie diverse Multimedia-Geräte. Zusätzliche Schnittstellen zu *Global Positioning System*[2] (GPS) und Fernerkundungssystemen, wie Satellit und Flugzeug, wären denkbar.

Weitere Werkzeuge sollte die Software bereitstellen, die neben einer grafischen Benutzeroberfläche auch Algorithmen zur Verschneidung, zur Interpolation, zur grafischen Aufbereitung usw. zur Verfügung stellen und das Durchführen von Transformationen ermöglichen sollte (eigene Mitschrift zu. [Roehrig 03/1]).

[2] Global Positioning System (GPS) = Satellitenortungssystem, ermöglicht die Ermittelt der geografischen Position

3.3 Decision-Support-Systeme

Die so genannten entscheidungsunterstützenden Systeme - Decision-Support-Systeme - kurz DSS genannt, gehören zur Gruppe der Managementunterstützungssysteme (MUS oder angelsächsich: Management-Informations-System MIS).

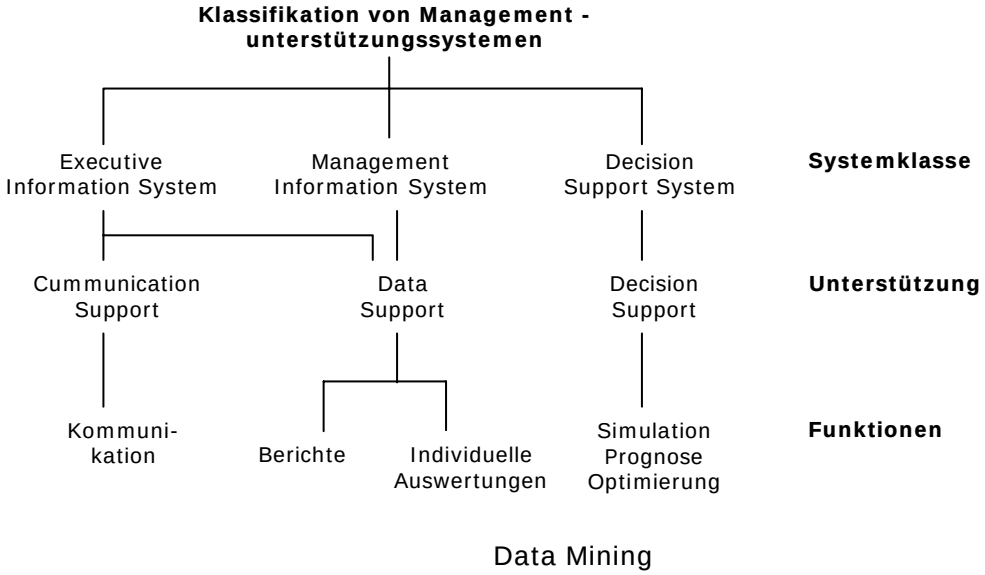


Abb. 3.1: Klassifikation von MUS (vgl. [West 03/1], Seite 9)

Diese Systeme haben ihren Ursprung in der Betriebswirtschaft und sollen der Entscheidungsfindung dienen. Das heißt, ein DSS soll den Entscheidungsträger bei komplexen Problemstellungen unterstützen. Entscheidungen stehen immer dann an, wenn sich zum Beispiel bei einer Problemstellung verschiedene Lösungsansätze anbieten. Es gibt aber auch noch andere Bedingungen, unter welchen eine Entscheidungsunterstützung gefragt ist (vgl. [GIS 04/1]):

- wenn es keine Fakten oder anerkannte klare Regeln gibt, anhand derer eindeutige Entscheidungen zu fällen sind,
- wenn die Informationen, die verwendet werden sollen, nicht besonders übersichtlich sind,
- wenn sich mehrere Lösungsstrategien anbieten,
- wenn ein Problem mit den zur Verfügung stehenden Daten oder Kenntnissen nicht (oder nicht optimal) lösbar ist,
- wenn die Erarbeitung einer Kompromißlösung notwendig ist.

Grundsätzlich ist eine Entscheidungsunterstützung bereits dann gegeben, wenn den Entscheidungsträgern Anleitungen und Hilfen an die Hand gegeben werden, die eine systematische und informativere Datenaufbereitung (z.B. Tabellen, Grafiken, vergleichende Darstellungen) ermöglichen. Geoffron (1983) setzt 6 Elemente für ein DSS voraus (vgl. [Good 04/1]):

- zum Lösen von un- bzw. semistrukturierten Problemen entworfen,
- über ein leistungsfähiges und zugleich leicht bedienbares User-Interface verfügen,
- Fähigkeit zur fexiblen Kombination von Daten und Anwendungsmodellen bieten,
- dem Anwender/Benutzer helfen, einen Lösungsraum (Alternativenraum) zu entwickeln und zu erforschen; d.h., es soll die Möglichkeit bieten, unter Anwendung der Modelle im System eine Serie von möglichen Alternativen zu generieren,
- die Vielfalt von Entscheidungsbildungsarten(-stilen) unterstützen und leicht anpassungsfähig sein; erweiterbar, um neue Fähigkeiten zur Verfügung zu stellen, die den Benutzeranforderungen entsprechen,
- über interaktive und rekursive Lösungsstrategien(-prozesse) mit unterschiedlichen Lösungswegen verfügen und nicht nur lineare sequentielle Abarbeitungspfade bieten.

Aus diesem Grund kann man DSS als computerbasiertes System definieren, welches als Werkzeug den Entscheidungsträger in die Lage versetzt, objektive qualifizierte Lösungen von meist nicht vollständig beschriebenen Problemstellungen zu finden.

Parallel zu den ökonomischen DSS entwickelte sich ein Spatial[3]-DSS-Konzept (SDSS-Konzept) für räumliche Problemstellungen. Die Entwicklung von räumlichen Entscheidungsunterstützungssystemen wurde mit der Notwendigkeit verknüpft, die GIS-Kompetenzen zur Bewältigung von komplexen, semi- bzw. unstrukturierten räumlichen Entscheidungsproblemen zu erweitern, also über die Aufgaben eines DSS hinausgehend die Integration von räumlichen Daten sowie raumbezogner Analysemethoden zu ermöglichen.

[3] Spatial = Raum

Eine zusammenfassende Definition von Geoffron (1983) über SDSS (vgl. Good 04/1) lautet:

”SDSS is an interactive, computer-based system designed to support a user or group of users in achieving a higher effectiveness of decision making while solving a semi-structured spatial decision problem.”

Die für DSS entwickelten Eigenschaften wurden für das SDSS-Konzept übernommen. Zusätzlich wurden diese von Densham (1991) um vier Eigenschaften erweitert:

- Fähigkeit zur Eingabe räumlicher Daten (Vektor-, Rasterdaten);
- Fähigkeit zur Speicherung und Darstellung der komplexen räumlichen Strukturen und Relationen;
- Verfügbarkeit von raumbezogenen analytischen Techniken (Methoden und Modellierungsfunktionen);
- Verfügbarkeit von Output-Funktionen, die räumlichen Daten entsprechen, also kartografische Darstellungen, 3D-Plots.

Ein DSS erweitert um die Möglichkeiten zur Verarbeitung von räumlichen Daten und raumbezogener Analyse- sowie Modellierungstechniken stellt ein räumliches DSS dar. Damit auch Nicht-Experten nach kurzer Einarbeitung ein Umgang mit SDSS möglich ist, besitzen Benutzerfreundlichkeit und ein User-Interface, welche ein intuitives Arbeiten mit SDSS ermöglichen, einen hohen Stellenwert. Da bei räumlichen Entscheidungsprozessen meist mehrere Entscheidungsträger beteiligt sind, muss der Weg zur Entscheidungsfindung transparent und plausibel sein, dafür ist eine Berichtserstellung notwendig.

GIS sind aus der Notwendigkeit entstanden, große Mengen an geografischen Informationen zusammenzuhalten und schnell zugänglich zu machen. Sie entsprechen einem räumlichen Informationssystem, das strukturiertes Wissen repräsentiert. Da GIS weder raumbezogene analytische Modellierungsfunktionen noch entscheidungs-unterstützende Methoden, wie Optimierungs- und Nutzenanalyse-Techniken etc. enthält, können sie nur in Kombination mit DSS-Komponenten zu einem räumlichen DSS (SDSS) werden.

Der Begriff Spatial Decision Support Systems (SDSS) wird im Zusammenhang mit GIS verwendet. Die Entwicklung von räumlichen DSS wurde mit der Notwendigkeit verknüpft, die GIS-Kompetenzen um die Eigenschaft der raumbezogenen analytischen Techniken (Methoden und Modellierungsfunktionen) zu erweitern. Verwendet man GIS als Baustein für SDSS, besteht dadurch die Möglichkeit alle GIS-Funktionalitäten wiederzuverwenden und an die SDSS-Komponenten zu koppeln.

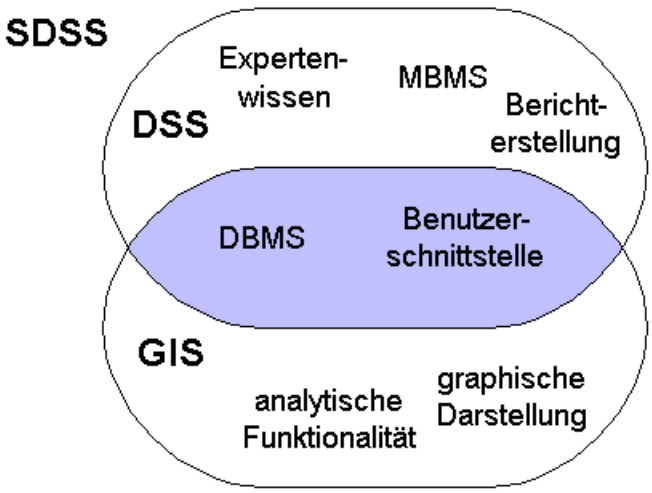


Abb. 3.2: SDSS aus GIS- und DSS-Komponenten ([GIS 04/1], sdss/Kap.8 abgeändert)

In Abbildung 3.2 wird die Schnittmenge der Funktionalitäten bzw. der Module von DSS und GIS verdeutlicht. Man sieht hier, dass die Model-Datenbank (MBMS[4]) als GIS-Komponente nicht vorhanden ist, es ist aber die wichtigste Komponente im DSS. Dieses Modul müsste entweder neu implementiert werden oder von einem bereits implementierten DSS übernommen und durch räumliche Modelle erweitert werden.

[4] MBMS = Modelbase-Management-System

Teil II

Grundlagen der Generierung von Dreiecksnetzen

4 Integration von räumlichen Daten in GIS

4.1 Räumliche Daten

Die Daten bilden den Kern eines GIS. Für räumliche Informationssysteme müssen die Grunddaten in einer bestimmten Form vorliegen. Dabei ist ein wichtiger Aspekt der Zugriff und die Bereitstellung der Daten zur Weiterverarbeitung. GIS ermöglicht dies über integrierte Schnittstellen. Die Daten werden entweder in Dateien gespeichert oder in einem Datenbanksystemen (RDBMS- Relational Database Management System) gehalten. Räumliche Daten stellen einen Teil der Erdoberfläche dar. Sie können die darauf befindlichen technischen und administrativen Einrichtungen sowie geowissenschaftliche, ökonomische und ökologische Gegebenheiten abbilden. Räumliche Daten beschreiben also unterschiedliche Eigenschaften von realen oder abstrakten Objekten unserer Umwelt.

Die wichtigsten Eigenschaften sind nach Bill hierbei (vgl. [Bil 91/1], Seite 208 ff.):

- Strukturelle Eigenschaften
- geometrische (metrische und topologische) Eigenschaften
- thematische Eigenschaften.

Mit *Struktur* ist der Zusammenhalt der Objekte gemeint. Es geht um die Art und Weise, wie man aus atomaren Bestandteilen höherwertige Komplexe schafft.

Der Raumbezug ist das verbindende Element in GIS. Die *Geometrie* kommt in ihm zum Ausdruck.

Definition: Information(en) zu Phänomenen und Sachverhalten, die direkt oder indirekt mit einer Position auf der Erde verknüpft sind, nennt man *raumbezogene Daten* (vgl. [BuReg 00/1], Seite 3). Der *Raumbezug* wird primär durch die Geometrie, d. h. die absolute Lage im Raum, beschrieben.

Alle Objekte weisen mehr oder minder einen gewissen Raumbezug auf und erfüllen somit Voraussetzungen bezüglich der Lage und der Ausdehnung. Beispiele dafür sind Gebäude, Grundstücke, Flüsse und Wasserleitungen.

Geometrie

Ausgehend von der euklidischen Geometrie, die als elementare Geometrie bezeichnet wird und zurückzuführen ist auf den griechischen Mathematiker Euklid (um 300 v. Chr.), stellt sie die Konkretisierung der Mengentheorie für metrische Räume dar. Euklid (vgl. [Endl 85/1], S. 207) strebte einen systematischen Aufbau der Geometrie an, ausgehend von Grundbegriffen, wie Punkte, Gerade, Ebene, Winkel und allgemein verständlichen Axiomen und Forderungen, wie Zugehörigkeit, Anordnung, Deckungsgleichheit, Stetigkeit.

- Euklidischer Raum

  Der Begriff des euklidischen zweidimensionalen Raumes wird eingeführt.

  Definition: Ein *euklidischer Raum* ist in der Mathematik ein Raum, für den die Gesetze der euklidischen Geometrie gelten. Euklidische Räume existieren in beliebigen Dimensionen *n*. Ein zweidimensionaler euklidischer Raum heißt auch euklidische Ebene (vgl. [Knaurs 78/1], S. 718).

  Für die Ebene lässt sich ein euklidischer Raum durch das 2-fache kartesische Produkt der reellen Zahlenmenge R beschreiben. Es wird ein so genanntes kartesisches Koordinatensystem eingeführt, bei dem die zwei Achsen einen Winkel von 90 Grad bilden und zwei Einheitspunkte E1, E2 einen relativ gleichen Abstand zum Nullpunkt O haben. Daraus ergibt sich die bekannte Metrik im $\mathbb{R}^n$ (vgl. [Endl 85/1], S. 98).

  Definition: Es seien $X = (x_1,..., x_n)$, $Y = (y_1,...,y_n)$ zwei Punkte des Raums $\mathbb{R}^n$. Der Punktraum $\mathbb{R}^n$ heisst euklidisch, wenn die Metrik definiert wird durch: $d(X,Y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$

Ein weiteres geometrisches Charakteristikum ist die *Form*. Diese wäre zwar formal aus der Koordinatengeometrie und den nachfolgend erklärten topologischen Kriterien ableitbar, jedoch erweist es sich als praktisch, Formulierungen wie ‘Dreieck‘, ‘Wellenlinie‘, ‘kreisähnlich‘, ‘Form einer Sichel‘ explizit zu berücksichtigen.

Neben *metrischen*, also durch Längen-, Winkel-, Höhen- und Flächenmessungen erfassbaren Eigenschaften, sind spezielle *topologische Eigenschaften* hervorzuheben.

Topologie

Sie äußert sich in Beziehungen der Nachbarschaft, des Enthaltenseins und der Überschneidung und ähnlichem. Im Gegensatz zur Geometrie, welche die absolute Form und Lage im Raum betrifft, sind topologische Beziehungen zwischen Geoobjekten unabhängig von Maßen wie der Distanz (vgl.[Com 05/1]).

Die wichtigsten topologischen Beziehungen zwischen 2 Geoobjekten A und B nach Egenhofer sind:

A ist disjunkt zu B	A gleicht B
A enthält B	B enthält A
A überdeckt B	B überdeckt A
A trifft B	

Tabelle 2 : wichtige topologische Beziehungen zwischen 2 Geoobjekten

Definition: Unter topologischem Modellieren versteht man die Beschreibung, Bearbeitung und Speicherung der Geometrie der Lage von räumlichen Objekten (vgl. [Bil 91/1], S. 221).

Thematische Eigenschaften

Neben den Eigenschaften, die der Geometrie zuzurechnen sind, weist jedes Objekt auch *thematische Eigenschaften* auf. Diese Eigenschaften verwendet man zur Aggregation von Objekten in thematische Bereiche sowie zur Zuordnung der topologischen Grundprimitive hinsichtlich verschiedener Objektcharakteristiken (vgl. [Bil 91/1], Seite 238). Einige Beispiele dafür sind:

Topografie: Gebirge, Talsenken, Gewässer
Klima: Temperatur, Niederschlag, Besonnung
Natürliche Ressourcen: Hydrologie (Wasserhaushalt), Geologie (Lagerstätten)
Ökologische Themen: Boden-, Gewässer- und Luftgüte
Nutzung: Flächennutzung, Infrastruktur

Definition: Unter thematischem Modellieren versteht man die Beschreibung, Bearbeitung und Speicherung der zugrundeliegenden Thematik eines räumlichen Objekts. Als Hilfsmittel dienen thematische Ebenen und Objekthierarchien, in denen verschiedene thematische Inhalte vorgehalten und Objekte zusammengefaßt werden.

Geometrische und thematische Aspekte von Daten beeinflussen einander, aus diesem Grund sind Inkonsistenzen bei der Langzeitspeicherung der Daten zu vermeiden. Darum ist es notwendig, in einer EDV[5]-Umgebung die Konsistenzbedingungen explizit zu definieren. Was ebenfalls den Vorteil hat, dass sämtliche Anwenderprogramme auf diese Voraussetzungen aufbauen können (vgl. [Bil 91/1]).

Für die Erstellung von speziellen Modellen in STDSS liegen die Grunddaten zumeist schon in konsistenter Form als sogenannte *Shapefiles* vor. Ein für Geodaten häufig verwendetes Datei-Format.

[5] EDV = Elektronische Datenverarbeitung

4.1.1 Shapefiles

Die Datenspeicherung in GIS erfolgt über Dateien oder Datenbanken. Das Speichern von topologischen Datensätzen in Geodatenbanken benötigt jedoch oft eine Menge von geografischen und analytischen Prozessen. Dies bedeutet, dass die topologischen Verbindungen zwischen den Objekten erst hergestellt werden müssen, z. B. durch Vergleiche der Objekte unter Berücksichtigung der möglichen Beziehungen nach Egenhofer (siehe Tabelle 2, S. 22). Dies kann sehr zeitaufwendig sein. Außerdem wird für das Abspeichern topologischer Datensätze mehr Speicherplatz benötigt. Für viele GIS-Anwendungen ist eine einfachere Form der Datenspeicherung ausreichend.

So genannte Simple-Feature-Klassen[6] (siehe 4.2.4 Features) speichern Objektformen mit Punkten, Linien und Polygonen, jedoch ohne topologische Verbindungen. Diese Struktur hat den Vorteil der einfachen Darstellung und der hohen Zugriffsgeschwindigkeit. Ein Nachteil ist, dass keine räumlichen Beziehungen zwischen Objekten festgelegt werden können.

Mit dem Standard-Software-Produkt *ArcView*[7]*-GIS-2*, des Unternehmens *ESRI*[8] wurde das Shapefile-Format (.shp) eingeführt, um Mengen einfacher geometischer Objekte, ähnlich wie Simple-Feature-Datasets, verfügbar zu machen (vgl. [Zeiler 99/1], S. 68).

Die drei wichtigsten Dokumente, aus denen ein Shapefile besteht kann, sind ein Hauptdokument, ein Indexdokument und eine dBASE[9]-Tabelle. Diese drei Dokumente repräsentieren räumliche Daten und Eigenschaften (Attribute) von Objekten (Features). Ein Shapefile kann zusätzlich optional andere Dokumente, beispielsweise mit weiteren Indexinformationen enthalten. Die nach *Arc-GIS* zum Shapefile-Format gehörenden Dokumente stellen gemeinsam eine Feature-Klasse dar.

Ein Shapefile kann eine variable Anzahl gleichartiger geografischer Objekte, wie Straßen, Brunnen, Gebiete, durch Punkt-, Linien- oder Flächenobjekte repräsentieren (siehe 4.2.4 Features).

[6] Simple-Feature-Klassen = Einfache Objekt-Klassen
[7] Arc View = Teilprodukt der GIS-Software ArcGis von ESRI
[8] ESRI = Economic and Social Research Institute
[9] dBase = Relationales Datenbankverwaltungssystem f. Personalcomputer von Borland

Jedes einzelne Objekt eines Shapefiles repräsentiert genau ein geografisches Objekt und seine Eigenschaften (Attribute). Die Geometrie und Lage eines Features wird im Hauptdokument mit einer Liste seiner Koordinatenpaare (X,Y) gespeichert, deren Komponenten durch Fließkommazahlen repräsentiert werden. Zusätzlich enthält das Hauptdokument eine *Bounding Box* das minimale, umschließende, achsenparallele Rechteck aller Shapes (vgl. [ESRI 98/01],S.3) .

Attribute werden in einem zugehörigen Dokument im dBASE-Format gehalten. Jeder Attribut-Datensatz hat eine 1:1-Beziehung zu einem Shape-Datensatz im Hauptdokument. Attribute von anderen Objekten können in anderen dBASE-Tabellen gespeichert und über einen Attributschlüssel mit dem Shapefile verbunden werden.

Wegen Fehlen topologischer Abbildungen benötigen Shapefiles weniger Speicherplatz und sind effizient zu lesen und zu beschreiben (vgl. [ESRI 98/01],S.1). Ein Shapefile ist die einfachste Form der Datenspeicherung in ArcGIS.

4.1.2 Layer

Die Abbildung von Simple-Feature-Klassen durch Shapefiles deckt einen großen Teil geografischer Daten ab, weil sie leicht zu realisieren sind und ausreichen, um geografische Daten im Layerprinzip darzustellen.

Das Layerprinzip separiert die Geometriedaten von verschiedener thematischer Bedeutung streng durch die Abspeicherung in verschiedenen Ebenen, die auch als *Layer* bezeichnet werden. Durch die Überlagerung dieser Ebenen wird eine Gesamtdarstellung gewonnen. Das Ebenenprinzip entspricht dem Folienprinzip, welches bei der Herstellung analoger Karten üblich ist (vgl. [Bar 95/1], S. 42): Verschiedene Farbauszüge, die jeweils einem bestimmten Thema entsprechen, werden getrennt hergestellt und dann bei Bedarf überlagert. Wird das Layerprinzip in einem Geoinformationssystem verwendet, bezeichnet E_i, i = 1,2,...,n die jeweilige Ebene. Somit läßt sich das Ebenenprinzip als Vereinigungsmenge aller verfügbaren oder als Teilmenge selektiv gewünschter Ebenen darstellen (vgl. [Bil 91/1] , S. 240):

$$E = E_1 \cup E_2 \cup ... \cup E_n$$

Die Verknüpfung der Ebenen miteinander erfolgt durch den Raumbezug. Für alle Layer müssen die selben Gegebenheiten bezüglich der Metrik, des Maßstabes, der Genauigkeit gelten.

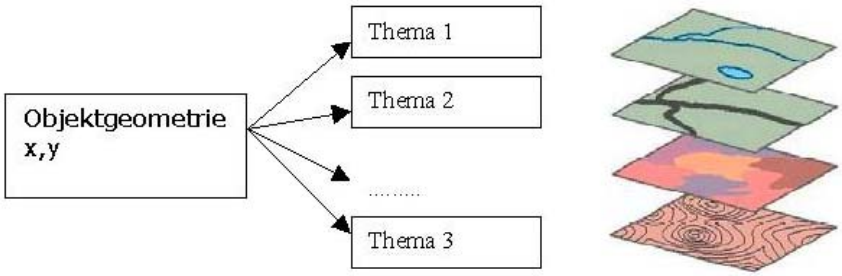


Abb. 4.1: Layer in einem Informationssystem

Das Ebenenprinzip besitzt keine Hierarchie, d.h. einzelne Layer sind voneinander unabhängig. Ebenfalls gibt es auch keine gemeinsamen Daten, zumindest was die Originaldatenbestände betrifft. Für Vergleiche, Verschneidungen und Bilanzen , die sich über mehrere Layers erstrecken, wird ein neuer Ergebnislayer angelegt.

Aus folgenden Gründen erfordert das Ebenenprinzip eine strenge thematische Separation (vgl. [Bar 95/1], S. 43):

- Arbeitsabläufe, Kompetenzen, Zugriffsberechtigungen werden leichter zugeordnet,
- die Übersichtlichkeit bleibt gewahrt,
- die Fehleranfälligkeit reduziert sich auf lokale Bereiche,
- Computerressourcen werden sparsamer verwendet,
- in einer Umgebung mit mehreren Nutzern gibt es weniger Konflikte.

Für die Weiterverarbeitung und Veränderung durch Neumodellierung und numerische Berechnungen der Daten verfolgt man ein objektorientiertes Vorgehen.

4.1.3 Objektorientiertes Modell

Der objektorientierte Gedanke ist ein integrierendes Konzept, welches Softwareengineering, Programmiersprachen und Datenbanken einschließt. Er stellt ein Objekt in den Mittelpunkt. Unter einem Objekt versteht man einen Gegenstand des Interesses. Objekte können Personen, Dinge oder Begriffe sein, z.B. Mitarbeiter, Gebäude, Aufträge. Für den GIS-Bereich wird der objektorientierte Ansatz A.U.Frank und M.J. Egenhofer 1990 in folgender Form charakterisiert (vgl. [Bil 91/2] , S. 331): Man kapsele ein Datenobjekt, welches ein Stück der realen Welt repräsentiert, zusammen mit den zugehörigen Methoden in einer einzelnen Einheit. Andere Programmteile oder Objekte können das so eingekapselte Objekt nur über die mit dem Objekt abgelegten Methoden ansprechen und zu Aktionen veranlassen.

Objekte, die sich durch gleiche Eigenschaften beschreiben lassen, gehören zu einer Klasse, zum Beispiel Linie. Eine Klasse definiert für gleichartige Objekte (Instanzen einer Klasse) deren Struktur (Attribute), Verhalten (Operationen) und Beziehungen zu anderen Objekten (vgl . [BAL 99/1], S. 21).

Als eine Erweiterung erlaubt ein objektorientiertes Datenmodell den Aufbau beliebig komplex strukturierter Objekte, sogenannter zusammengesetzter Objekte. Diese Eigenschaft ermöglicht, eine auf das jeweilige Anwendungsgebiet zugeschnittene Datenstruktur für die betrachteten Realweltgegenstände aufzubauen und diese Struktur als Ganzes oder in ihren Komponenten direkt zu bearbeiten. Durch *Aggregation* besteht die Möglichkeit, Objekte in andere Objekte einzubauen.

Es ist auch möglich, dass ein Objekt in mehreren Objekten enthalten ist. So kann ein Objekt als Objektteile Flächen, Kanten und Knoten besitzen, gleichfalls kann es selbst Flächen-, Kanten- oder Knotenobjekt sein. Das daraus entstehende komplexe Objekt hat andere Merkmale und Eigenschaften als die Teilobjekte. Diese Denkweise bildet die Grundlage zur Strukturierung der geometrisch-topologischen Grundprimitive. Im GIS-Bereich sollte man hier mit der Klassenbildung beginnen.

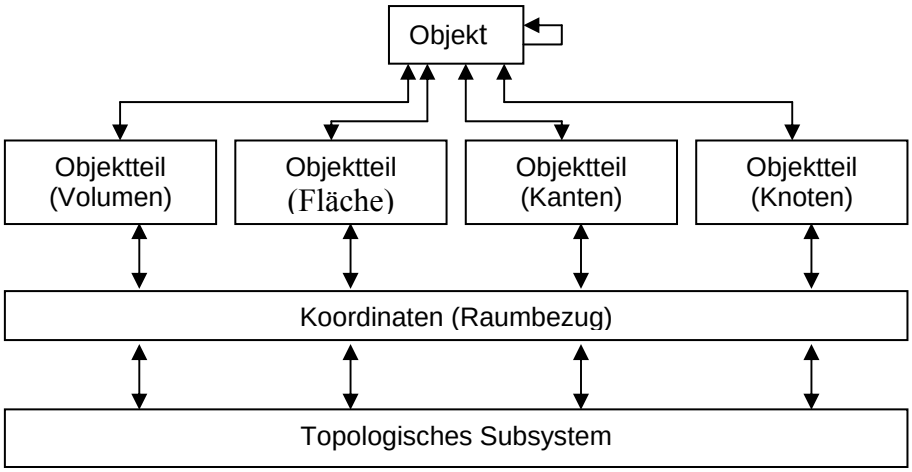


Abb. 4.2: Objektrelationen der topologischen Grundprimitive (vgl. [Bil 91/1] , S. 247):

Eine zweite Erweiterung der Objektbildung - parallel zur Aggregation - stellt die *Assoziation* dar. Ein Komplexobjekt besteht aus Einfachobjekten, zum Beispiel könnte das Objekt "Straße" aus den Einfachobjekten "Straßenachse", "Fahrbahn", "Gehsteig", "Verkehrszeichen" usw. bestehen.

Verallgemeinerung (*Generalisierung*) stellt die dritte Möglichkeit der Objektbildung dar. Sie wird sprachlich durch eine "*ist ein*"-*Beziehung* ausgedrückt. Ein Kennzeichen der objektorientierten Strategie ist die Möglichkeit der *Vererbung*. Sie ist eng mit der "*ist ein*"-*Hierarchie* gekoppelt. Auf einem Objekt sind nicht nur die Eigenschaften und Methoden der Klasse definiert, der es direkt zugeordnet ist, sondern darüber hinaus "erbt" es die Eigenschaften aller hierarchisch übergeordneten Klassen (vgl. [Bil 91/2], S. 340).

Man unterscheidet zwei Vererbungsarten. Bei der Einfachvererbung gibt es jeweils nur eine übergeordnete Klasse, die ihre Eigenschaften an die jeweils untergeordnete Klasse weitergibt. Als Mehrfachvererbung bezeichnet man einen Zustand, bei dem eine Objektklasse Eigenschaften von verschiedenen anderen Objektklassen erben kann.

Objektorientierte Programmierung

In den 90er Jahren hat sich die Programmiersprache C++ durchgesetzt. In den letzten Jahren gewinnt aber die Programmiersprache Java, die ursprünglich von der Firma Sun Microsystems entwickelt wurde, auch in GIS-Anwendungen an Popularität.

Gründe für die Realisierung der Anwendung STDSS und seiner hier teilweise beschriebenen Komponenten in Java sind die plattformübergreifenden Fähigkeiten von Java-Anwendungen. Durch die Betriebssystemunabhängigkeit ist eine einfache Portierung[10] der Anwendung auf verschiedene Systeme gewährleistet. Dies wird erreicht, weil der Java-Compiler keinen Maschinencode für eine spezielle Plattform generiert, sondern einen architekturneutralen Bytecode erzeugt (vgl. [Ullenboom 02/1]). Dieser wird dann von einer Java-Laufzeitumgebung interpretiert. Die Laufzeitumgebung ist plattformabhängig und stellt dem Java-Programm eine Implementierung abstrakter Schnittstellen für systemabhängige Operationen zur Verfügung (z.B. für Zugriff auf Dateisysteme, auf Netzwerkschnittstellen, grafische Oberflächen etc.).

Ein weiterer Vorteil ist das Fehlen von Zeigern auf Variablen und Funktionen, denn eine unsaubere Programmierung mit Zeigern ist die Hauptursache für kritische Abstürze einer Anwendung. Java verfügt über eine automatische Speicherverwaltung. Da Java eine Interpretersprache ist, führen schwere Laufzeitfehler (z.B. Zugriff auf ungültige Objektreferenzen) nicht zu unkontrollierten Systemabstürzen. Das Programm wird von der Laufzeitumgebung kontrolliert beendet. Kritische Fehler können somit schneller gefunden und beseitigt werden. Außerdem gibt es in Java keine globalen Variablen und Funktionen, denn alle Programmobjekte sind in Klassen definiert.

Die zunehmende Popularität hat zu einem weiteren Vorteil, speziell im GIS-Bereich, geführt. Mittlerweile werden spezielle kommerzielle Zusatzbibliotheken angeboten (z.B. Map Objects 2.0 von ESRI), in welchen Klassen für einfache und komplexe geometrische Objekte zur Verfügung gestellt werden. Einfache geometrische Objekte lassen sich inzwischen durch Klassen der Standard-Java-Bibliothek *java.awt.geom.* erzeugen. Komplexere Objekte können von diesen abgeleitet und selbst programmiert werden.

Objektorientierte Datenmodelle eignen sich, um raumbezogene Problemstellungen zu bearbeiten. In der Praxis, das gilt auch für das STDSS, werden bisher jedoch meist hybride Strukturen verwendet, die neben den objektorientierten Datenschemata auch relationale Aspekte, zum Beispiel bei Sachdaten, berücksichtigen sowie das Layerprinzip verfolgen.

[10] Portierung = von Portierbarkeit: Übertragbarkeit von Software auf verschiedene Systemplattformen.

4.2 Datenstrukturen

Mit Hilfsmitteln des geometrischen, topologischen und thematischen Modellierens können raumbezogene Daten strukturiert werden:

- Das *thematische Modell* fasst Geometriedaten verschiedener thematischer Bedeutung zusammen und separiert diese streng durch die Abspeicherung in verschiedenen Ebenen (vgl. 4.1.2 Layer).
- Das *topologische Modell* ist die Beschreibung, Bearbeitung und Speicherung der Geometrie der Lage von räumlichen Objekten (vgl. [Bil 99/1], S. 221). Sie äußern sich in Beziehungen der Nachbarschaft, des Enthaltenseins und der Überschneidung und ähnlichem (vgl. 4.1 Daten, Topologie).
- Im *geometrischen Modell* wird die Geometrie von räumlichen Objekten durch die Form - auch geometrische Beschreibung genannt - und die relative Lage von Punkten - etwa durch Koordinaten - vollständig beschrieben (vgl. [Bil 99/1], S. 21).

Die für GIS relevanten Daten sind hinsichtlich des Umfangs und ihrer räumlichen Ausdehnung meist heterogen. Solche Daten können z. B. sein:

Flächenhaft- kontinuierliche, flächenhaft- diskontinuierliche, linienhafte und punkthafte Informationen sowie Zwischentypen.

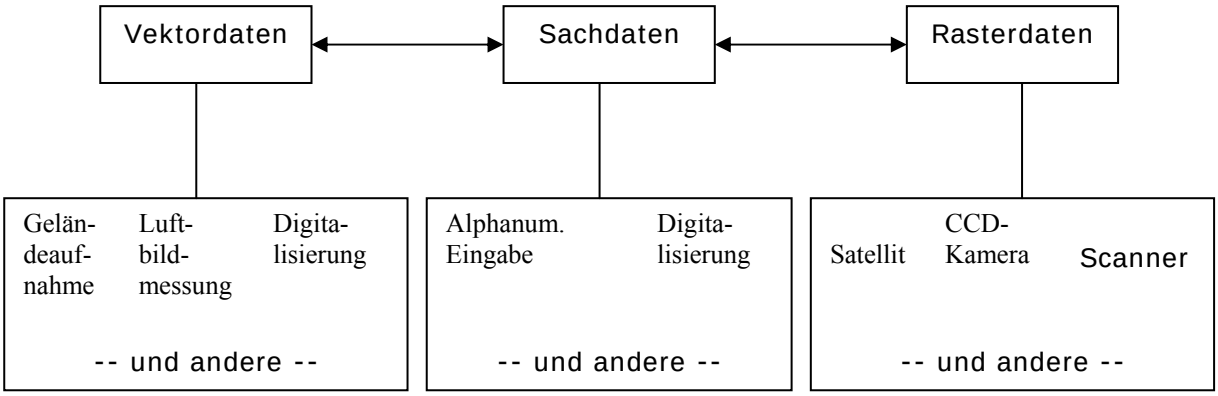


Abb. 4.3: Datenquellen für GIS (vgl. [Bil 99/1], S. 36)

Informationen, die einen räumlichen Bezug haben, aber keine eigentlichen Geometriedaten sind, werden als *Sachdaten* (Attribute) bezeichnet. Zur Verwaltung von Geometriedaten in einem GIS unterscheidet man unterschiedliche Datenformate, wie *Rasterdaten*, *Vektordaten* oder *Dreiecksnetze*.

4.2.1 Sachdaten

Alle Informationen, die einen räumlichen Bezug haben, aber keine eigentlichen Geometriedaten sind, nennt man Sachdaten oder Attribute. Sie repräsentieren sämtliche nichtgeometrischen Elemente wie Texte, Zahlensammlungen, Messwerte, Nummern, Namen, Eigenschaften etc. (vgl. [Bil 99/1], S. 29).

Sachdaten in digitaler Form sind zumeist in Tabellen gespeichert. Diese Tabellen bestehen aus Zeilen und Spalten. Jede Zeile beschreibt einen Zustand oder ein Objekt. Eine Spalte enthält Informationen vom gleichen Typ (vgl.[Roehrig 03/1], S. 12). Der Spaltenname wird Feld genannt. Tabellen werden mit räumlichen Objekten durch ein gemeinsames Feld verknüpft. Die Sachdaten werden so über einen gemeinsamen Schlüsselwert dem Objekt zugeordnet.

4.2.2 Rasterdaten

Rasterdaten entstehen durch Scannen von Plänen, Luft- und Satellitenbildern u.ä. Geometrische Sachverhalte lassen sich mit Rasterdaten relativ einfach darstellen. Die *Rasterzelle* ist der einzige Objekttyp als goemetrisches Element (vgl. [Bar 95/1], S. 95).

Die Rasterzelle ist quadratisch und überdeckt ein Gebiet mit homogener Bedeutung. Alle Rasterzellen sind gleich groß und unterliegen einer regelmäßigen Anordnung. Dabei entsteht eine zeilen- und spaltenartige Raumunterteilung in Form eines regelmäßiges Gitters (*Grid*). Den Zellen fehlen jedoch logische Verbindungen untereinander. Sie enthalten lediglich Werte über Eigenschaften. Darum eignen sie sich auch zur Darstellung von Daten allgemeiner Bedeutung mit kontinuierlichem Verlauf, die lediglich stetigen Änderungen unterliegen. Beispiele hierfür sind (vgl. [Roehrig 03/1], S.8):

- Daten für Analyse und Modellierung, meist aus Statistiken, die relativ grob für rechteckige Bereiche gegeben sind (Waldschäden, Niederschlag).
- Aus Messpunkten: Bodenanalyse, Wasseranalyse, usw.
- Aus interpretierten Bildern: Bodenklassifizierung, Landnutzung.
- Klassifizierung nach Kategorien (Temperaturunterschiede u.a.).

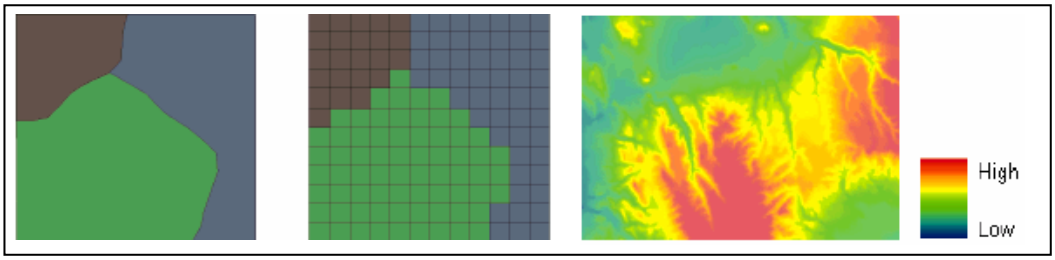


Abb. 4.4: Modellierung und Analyse von Rasterdaten (vgl. [Zeiler 99/1])

Jede Rasterzelle ist Träger von Informationen. Der Farbwert ist nicht wörtlich als eine Information zu interpretieren; vielmehr steht er als Sekundärinformation stellvertretend für eine thematische Aussage. Die Genauigkeit der Darstellung hängt von der Auflösung ab.

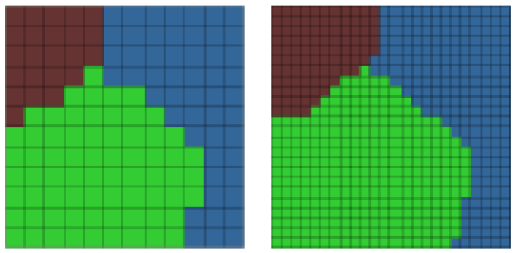


Abb. 4.5: Auflösung von Rasterdaten, Treppeneffekt (vgl. [Zeiler 99/1])

Für Rasterdaten gilt, je kleiner die Zellengröße, desto höher ist die Auflösung. Für homogene Flächen gilt, durch eine höhere Auflösung wird die Randdarstellung genauer und mildert den so genannten Treppeneffekt. Dies hat jedoch einen höheren Speicherbedarf zur Folge, da sich die Anzahl homogener Zellen drastisch erhöht.

4.2.3 Vektordaten

Anders als bei der flächenhaften Beschreibung, wie den Rasterdaten, werden bei Vektordaten die geografischen Elemente in einem Koordinatensystem als Vektoren mit x- und y-Koordinaten bei einer zweidimensionalen Darstellung gespeichert. Der Punkt, die Linie und die Fläche sind die Grundelemente von Vektordaten, wobei der Punkt der Träger der geometrischen Information ist. Alle höheren Strukturen, wie beispielsweise Linien und Flächen, bauen auf dem Punkt auf. Für diese höheren Strukturen lassen sich sämtliche geometrischen Aussagen, wie die Länge von Verbindungen, der Flächeninhalt, der Abstand zweier geometrischer Figuren und anderes ableiten.

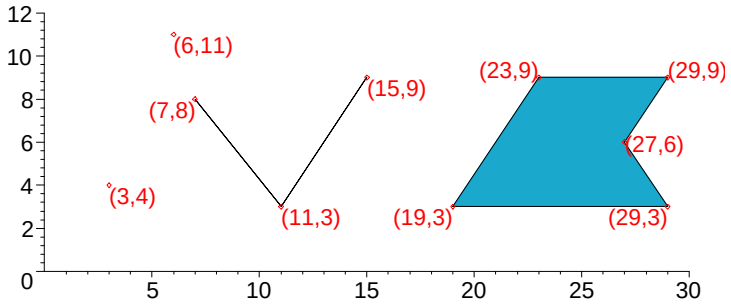


Abb. 4.6: Elemente des Vektormodells (vgl. [Zeiler 99/1])

Zur Vereinfachung der Modellbildung existieren auch neben den Grundelementen von Vektordaten wie Punkt, Linie und Fläche einige Aggregate, wie Mulitpoint, Multilinie, und Multifläche. Werden diese elementaren geografischen Elemente mit Attributen versehen, entstehen räumliche Objekte, so genannte *Features*.

4.2.4 Features

Features sind geografische Objekte der Welt. Sie existieren in unterschiedlichen Kontexten. Features besetzen Punkte, Linienzüge oder Gebiete – haben Nachbarn und beeinflussen umliegende Objekte. Features besitzen außerdem Attribute, welche Werte, Summen, Kategorien oder Beschreibungen repräsentieren. Durch den Datenverlauf der Attribute kann man mögliche Reaktionen auf diverse Einflüsse anderer Objekte vorhersagen.

Für viele Applikationen ist das Vector-Datenmodell die geeignete Repräsentation zur Darstellung von geografischen Objekten mit sich kaum ändernden Grenzen. Feature-Eigenschaften, wie Vector-Formen, Beziehungen, Attribute und Verhalten versuchen den vielfältigen Rahmen geografischer Objekte abzudecken (vgl. [Zeiler 99/1], S. 83).

Die Form eines Features wird durch seine Geometrie bestimmt. Jedes Feature hat eine mit ihm assoziierte Geometrie oder Form. In einer Tabelle ist die Geometrie in einem speziellen Feld mit Attributnamen "shape" der Feature Klasse gespeichert.

Im geometrischen Objekt-Modell gibt es zwei Stufen von Geometrien – jene, die die Form von Feature-Objekten definieren und andere, welche die Komponenten der definierten Formen beschreiben.

Features können durch folgende Geometrien repräsentiert werden:

Punkt und Multipunkt (point, multipoints)

Punkte sind *0-Dimension-Geometrien*, was bedeutet, dass sie keinen Rand (*boundary*) besitzen. Sie haben eine x-Koordinate und eine y-Koordinate mit einem optionalen Attribut (z). Sie repräsentieren kleinere Objekte, z.B. Brunnen.

Multipoint ist eine ungeordente Ansammlung von Punkten. Es repräsentiert ein Set von Punkten mit gemeinsamen Attributen, wie eine Reihe von Brunnen, welche eine kleine Einheit bilden.

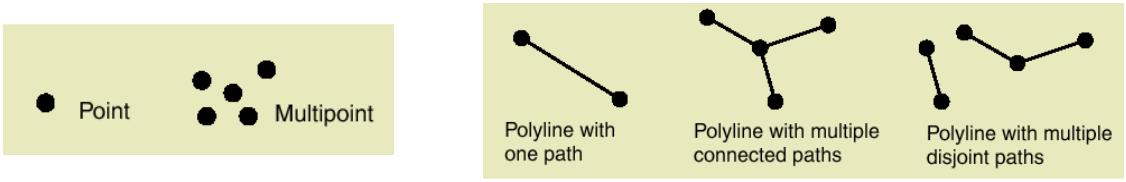


Abb. 4.7: Punkte und Multipunkt (li) – Polyline (re), (vgl. [Zeiler 99/1], S. 102)

Polyline

Eine Polyline ist eine geordnete Kollektion von Strecken, die verbunden oder unterbrochen sein können. Polylines werden benutzt um lineare Objekte darzustellen. Beispielsweise repräsentieren sie Straßen, Flüsse und Konturen. Einfache lineare Objekte werden durch Polylines mit einer Verbindungslinie dargestellt. Komplexe Linienobjekte, beispielsweise ein Straßennetz, werden durch Polylines mit mehreren Strecken dargestellt. Polyline sind *1-Dimension-Geometrien*, was bedeutet, dass Anfangs- und Endpunkt den Rand bilden.

Polygon (area)

Ein Polygon ist eine Kollektion von Ringen, welche nach Inhaltsvereinbarungen sortiert sind. Polygone stellen Flächenobjekte dar. Einfache Flächenobjekte werden durch einen Ring repräsentiert. Falls es sich um verschachtelte Ringe handelt, wird zwischen so genanntem Interior-Ring (innen) und Island-Ring (Insel) unterschieden. Ein Polygon kann mehrere innere Ringe enthalten, welche einzelne Polygone darstellen, die sich jedoch nicht überlappen. Sie gehören zur Kategorie der *2-Dimension-Geometrien*, bei welcher die umschließenden Segmente den Rand bilden.

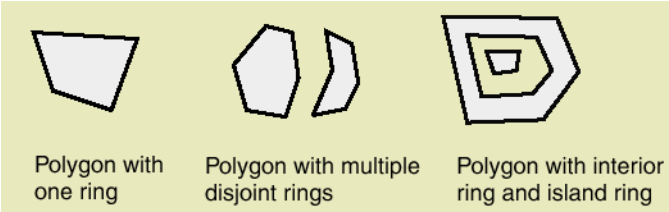


Abb. 4.8: Polygon (vgl. [Zeiler 99/1], S. 103)

Die Komponenten der Feature-Geometrie werden als Segmente, Paths und Ringe bezeichnet und nachfolgend erklärt:

Segmente

Ein Segment besteht aus einem Start- und einem Endpunkt und einer funktionsdefinierten Kurve zwischen den Punkten. Es gibt vier Segmenttypen:

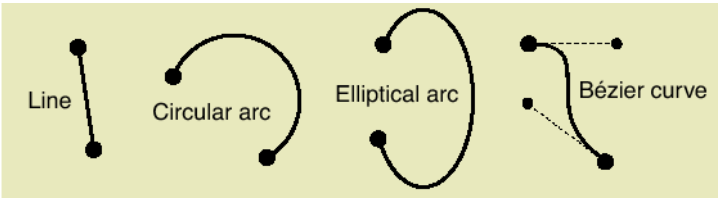


Abb. 4.9: Segmente (vgl. [Zeiler 99/1], S. 103)

- Eine *Line* (Linie) ist die einfachste Form, ein gerades Segment, begrenzt durch zwei Endpunkte. Dient meist der Darstellung von Landparzellen oder Autobahnabschnitten.
- Ein *Circular Arc* (Kreisbogen) ist ein Ausschnitt eines Kreises. Er repräsentiert häufig Straßenverbindungen an Autobahnkreuzen.
- Ein *Elliptical Arc* (elliptischer Bogen) ist ein Teil einer Ellipse. Es wird nur selten zur Feature-Definition benutzt. Sie ähneln jedoch Übergangsfiguren, wie zum Beispiel Autobahnauffahrten.
- Eine Bézier Curve ist mit vier Kontrollpunkten definiert. Sie stellen Objekte mit fließenden Übergängen dar, wie Konturen oder Ströme.

Path

Ein Path ist eine Sequenz von verbundenen Segmenten. Diese Segmente in einem Path können nicht unterbrochen sein. Ein Path kann verschiedene Kombinationen von Linien, Circular und Elliptical Arcs und Bézier Curves beinhalten. Aus Paths entstehen Polylines.

Manchmal berühren sich die Segmente eines Path. Das bedeutet, die Segmente sind an einem Randpunkt verbunden und tangieren einander. Ein Beispiel könnten Höhenlinien sein, welche ein ganzes Set von verbundenen und tangierenden Bézier Kurven darstellen.

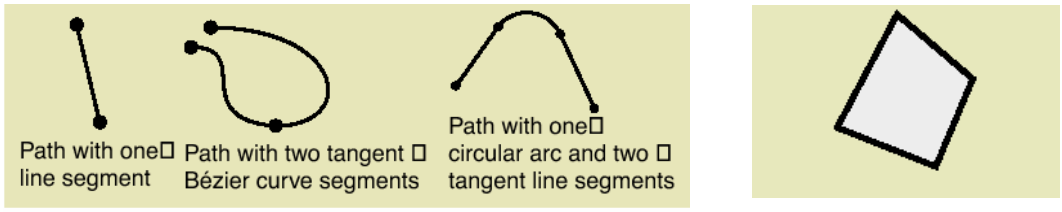


Abb. 4.10: Path (li) -- Ring (re), (vgl. [Zeiler 99/1], S. 103)

Ringe

Ein Ring ist ein geschlossener Path, der eine eindeutige Innen- und Außenseite hat, dadurch entsteht ein Polygon. Für diesen Path sind die Koordinaten des Start- und Endpunktes die gleichen.

Weitere Feature-Geometrien:

Envelope

Ein Envelope (Umschlag) repräsentiert die spatiale Ausdehnung eines Objektes. Alle Geometrien haben Envelopes. Ein Envelope ist ein Rechteck, welches die kleinsten und größten Koordinaten einer Geometrie umspannt. Die Seiten eines Envelope sind immer parallel zum Koordinatensystem.

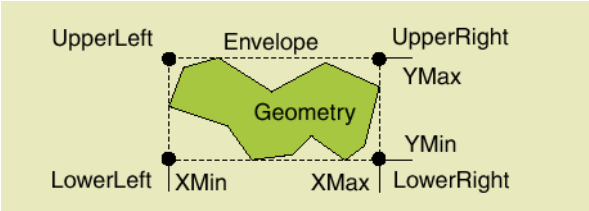


Abb. 4.11: Envelope (vgl. [Zeiler 99/1], S. 103)

Durch Kombination von einfachen und aggregierten Geometrien in einer Feature-Klasse ist eine flexible Gestaltung eines individuellen Datenmodells gegeben. Beispielsweise kann eine Feature-Klasse mit Polyline-Geometrie einfache oder aggregierte Polylinien beinhalten. Eine Feature-Klasse mit Polygon-Geometrie kann einfache oder aggregierte Polygone enthalten.

Weiterhin kann man Features nach ihren Ausprägungen unterscheiden:

- Einfache Ausprägungen von Features nennt man *Simple Features*, sie haben eine elementare geometrischen Vektorform, topologische Verbindungen und festgelegte Beziehungen, definierte Attribute und unterliegen speziellen festgelegten Regeln.
- *Komplex-Features* entstehen durch Zusammensetzen von elementaren Features zu höherwertigen Strukturen mit Hilfe von hinzugefügtem Softwarecode. Sie können ebenfalls eigene Charakteristika erhalten.

4.2.5 Komplex-Features

Eine spezielle Form von Komplex-Features sind Linien- und Flächennetze.

Dreiecksnetze

Die Dreiecksvernetzung bildet eine sinnvolle Alternative zu den Rasterdaten. Den Nachteilen des Treppeneffektes und des hohen Speicherbedarfs begegnet man mit einem Spezialfall der netzartig aufgeteilten Struktur von Flächen.

Ein aus unregelmäßig verteilten Punkten generiertes Netz aus Dreiecken nennt man TIN (*Triangulated Irregular Network*). Die Erzeugung eines TIN verfolgt spezielle Regeln für die Dreiecksbildung. Diese werden in Kapitel 6 und 7 ausführlich behandelt.

Abb. 4.12: Generiertes Netz aus Dreiecken, TIN (vgl. [Zeiler 99/1])

Ein Dreiecksnetz besteht aus Knoten (nodes) und Kanten (edges). Es wird in den Bereichen verfeinert, wo mehr Daten nötig sind, z.B. in Randgebieten. In Bereichen, in denen homogene Daten vorliegen, können die Dreiecke wachsen. Dies hat den Vorteil einer genauen Abbildung. Ebenfalls werden durch das Wachsen der Dreiecke in homogenen Flächen im Vergleich zur Rastervermaschung weniger Speicherressourcen benötigt.

Dreieck

Dreiecke sind die Elemente einer vernetzten Oberfläche. Ein Dreieck ist das kleinste mögliche Polygon. Es besitzt keine inneren Löcher. Drei Knoten (nodes) für die Ecken und Kanten (edges) für die Seiten definieren die Grundstruktur eines Dreiecks. Die "Wohlgeformtheit" eines Dreiecks drückt sich in der Gleichseitigkeit aus. Die damit in Verbindung stehenden Eigenschaften (z.B. Umkreisradius, Innenkreisradius, Schwerpunkt, Fläche) können frei vereinbart werden.

Gleichfalls ist es möglich, topologische Eigenschaften zu vereinbaren. Beispielsweise durch Festlegung der Nachbarschaftsbeziehungen, welche als Gittertopologie bezeichnet wird und in Kapitel 6 näher erklärt wird.

Weitere Informationen über Features und Objekt-Klassen kann man den nachfolgend beschriebenen Spezifikationen des GIS-Konsortiums, [OpenGis 01/1], entnehmen.

4.2.6 Spezifikation

Features und ihre Geometrien werden von der International Standardization Organisation[11] (ISO) und vom Open-Gis-Consortium spezifiziert in:

- ISO/ TC 211 19107 Geographic information – Spatial schema,
- OpenGIS Feature Geometry des Open-Gis-Consortioums.

[11] International Standardization Organisation = Internationale Standardisierungsorganisation, Sitz in Genf, die meisten Hard-/Software-Normen wurden entworfen und verabschieded.

Diese internationale Standardisierung (vgl. [OpenGis 01/1], Seite 1) spezifiziert konzeptionelle Schemata nach ISO 19107 für die Beschreibung der räumlichen Charakteristiken von geografischen Objekten und einer Menge von räumlichen Operationen, welche mit diesen Schemata einhergehen.

Es werden Vektor-Geometrien und –Topologien bis hin zur 3. Dimension betrachtet. Es definiert standarisierte räumliche Operationen für den Zugang, die Auswahl, Verwaltung und Auswertung sowie das Austauschen von geometrischen Informationen über räumliche Objekte.

Zur Beschreibung und Darstellung dieser konzeptionellen Schemata bedient man sich der *Unified Modeling Language* [12](UML). Diese definierten konzeptionellen Klassen werden in Anwendungsschemata, Profilen und Implementierungs-Spezifikationen verwendet und können (dürfen) weiterentwickelt werden.

[12] *Unified Modeling Language* (UML) = Modellierungssprache, die eine grafische Sprache zur Visualisierung, Spezifikation u. Konstruktion verschiedener Elemente von Informationssystemen zur Verfügung stellt.

5. Räumliche Indizierung

In diesem Kapitel wird die Organisation der anfallenden Datenmenge in Hinblick auf eine konsistente und effiziente Bereitstellung dieser Daten, für Speicherung, Zugriff und Weiterverarbeitung sowie Ergebnispräsentation beschrieben.

Jedes GIS muss in der Lage sein, spezielle Aufgaben, für die es entwickelt wurde, zu bewältigen, dazu gehört auch ein effizienter Datenzugriff. Häufig stellt die aufkommende Datenfülle ein Problem in der GIS-Technologie dar, wie auch die Effizienz beim grafisch-interaktiven Arbeiten. Dabei werden für die Datenspeicherung sowie ihre Bereitstellung Prinzipien angewendet, die durch eine entsprechende Datenorganisation eine Reduktion der Anzahl der Zugriffe zum Datenspeicher bewirkt. Was bedeutet, dass die gewählte Zugriffsstruktur die Verarbeitung einer großen Menge von Datensätzen unterstützen muss und Änderungen an der Datenbasis möglichst mit geringem Aufwand übernommen werden. Der Zugriff auf die Daten muss auf die erforderlichen Anfragen zugeschnitten sein und diese möglichst performant unterstützen.

In den meisten Anfragesituationen eines GIS spielen räumliche und zeitliche Beziehungen im Hinblick auf Objekte eine große Rolle (siehe [Bar 95/1], S. 208ff):

Punktsuche:	- Suche nach einem Punkt über seine Koordinaten - Suche der nächstliegenden *n* Punkte
Regionsuche:	- Suche nach Punkten/Objekten in einem Rechteck - Suche aller Punkte/Objekte in einem gegebenen Polygon - Suche Objekte, die sich mit einer Region überlappen
Relationsuche:	- Suche Objekte, die mit einem gegebenen Objekt in räumlicher oder topologischer Beziehung stehen.
Suche nach Metriken:	- Suche Objekte, die einen geringeren Abstand als den vorgegebenen haben.
Zeitlich abhängige Daten:	- Historischer Verlauf von Daten - Historie und Zeitangaben zu Veränderungen - Zeitliche Beschränkungen gültiger Versionen von Objekten

Tabelle 3: Übersicht der Anfragemöglichkeiten für Objekte

Räumliche Daten können nicht wie in einem Datenbank-Management-System linear sortiert werden, so dass im Raum benachbarte Objekte auch im Index benachbart abgespeichert sind. Daher wurden für Daten mit zwei und mehr Dimensionen eigene Indexstrukturen entwickelt, welche unter dem allgemeinen Begriff *räumliche Indizierung* zusammengefasst werden.

Unter *räumliche Indizierung* ist im Allgemeinen die Aufteilung des Raumes zu verstehen (siehe [Bar 95/1], S. 218).

Betrachtet man die oben genannten Anfragesituationen, stellt man fest, dass dem Punkt – der einfachsten Repräsentation einer Vektorstruktur – eine bedeutende Rolle zukommt, denn alle anderen Strukturen können auf ihn zurückgeführt werden. Dies gilt zunächst für topologische Strukturen, aber auch thematische Inhalte in einem GIS werden auf die Lage – und damit auf die Koordinaten von Punkten – bezogen. Aus diesem Grund gewährleisten Punkte die Basis eines raumbezogenen Zugriffs.

Für eine tiefgehende Betrachtung von hier nicht aufgeführten Verfahren der Indizierung wird auf einschlägige Algorithmenbücher wie [Berg 97/1] und [Ottm 02/1] u.a. verwiesen. Um einen Überblick zu geben, werden nachfolgend einige Verfahren beschrieben.

5.1 Gitterverfahren

Soll ein Punkthaufen so gespeichert werden, dass ein raumbezogener Zugriff (z.B. über Koordinaten) möglich ist, ohne alle Punkte sequentiell durchsuchen zu müssen, sollten diese der Lage nach geordnet sein. Eine einfache und effiziente Methode für die Beibehaltung von Nachbarschaftsbeziehungen zwischen Punkten in der Ebene besteht darin, ein künstliches Gitter zu konstruieren, das die zu durchsuchende Fläche in kleine Quadrate zerlegt. Jede Gitterzelle enthält im allgemeinen mehrere Punkte, während umgekehrt jedem Punkt genau eine Masche zugeordnet wird.

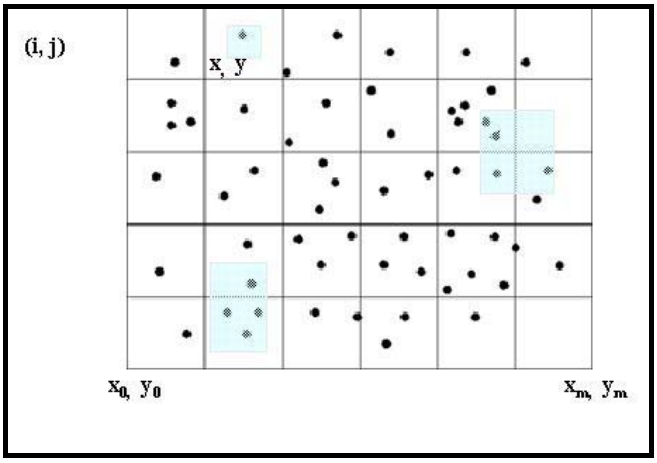


Abb. 5.1: Gitter zur Verwaltung eines Punkthaufens (vgl. [Bar 95/1], S. 219 geändert)

Bei Vorgabe eines Suchbereiches brauchen nur diejenigen Zellen durchsucht zu werden, welche sich mit dem Suchbereich überschneiden (siehe Abb. 5.1). Im ungünstigsten Fall bewegt man sich am Kreuzungspunkt von vier Zellen. Innerhalb einer solchen Zelle muss dann sequentiell gesucht werden.

Der Zugriff auf eine Zelle erfolgt über den Zeilen- und Spaltenindex und ist für jede Zelle konstant O(c). Sind die Punktkoordinaten bekannt, kann man ohne viel Aufwand auf die Zelle zugreifen, indem man den Zeilen- und Spaltenindex ermittelt (Bezeichungen wie in Abb. 5.1):

i = Spaltenindex ; j = Zeilenindex; (x,y) = Punktkoordinate

$$i = \frac{x - x_0}{x_m - x_0} \qquad j = \frac{y - y_0}{y_m - y_0} \tag{5.1}$$

Wichtig ist, die Größe des Gitters den Erfordernissen anzupassen. Zu wenige Zellen würden die Zeit für lokales Suchen überproportional ansteigen lassen. Zu viele Zellen wirken sich ungünstig auf die Verwaltung des Gitters aus. Außerdem gäbe es viele leere Gitterquadrate und die Suche würde sich ausweiten. Die Größe des Gitters sollte so gewählt werden, dass die Anzahl der Gitterquadrate ein konstanter Bruchteil der Gesamtzahl der Punkte ist. Dann ist zu erwarten, dass die Anzahl der Punkte in jedem Quadrat ungefähr gleich einer gewissen kleinen Konstante ist.

Um die Auflösung des Gitters zu bestimmen, kann man folgendermaßen vorgehen (vgl. [Sed 92/01], S. 433):

Die Größe G_q und Anzahl A_q (n mal n Gitterquadrate) der Gitterquadrate hängt von der Anzahl der Punkte, des verfügbaren Speicherplatzes und vom Bereich der Koordinatenwerte ab. Um bei N Punkten M Punkte pro Gitterquadrat zu erhalten, sollte für A_q das gerundete Ergebnis aus $\sqrt{N/M}$ gewählt werden. Um G_q zu ermitteln, wird die höchste Punktkoordinate durch $\sqrt{N/M}$ dividiert und gerundet.

Die Effizienz des Gitterverfahrens steht und fällt mit der gleichmäßigen Verteilung der Punkte im zu betrachtenden Bereich. Liegen die Punkte gehäuft beieinander, würden sich alle Punkte im schlechtesten Fall in einer Gitterzelle befinden. Die Inhomogenität von Geodaten verdeutlicht, dass das Gitterverfahren zur Indexierung nicht dynamisch genug ist. Aus diesem Grund benötigt man verfeinerte Konzepte.

5.2 Baumstrukturen

Bäume gehören zu den wichtigsten in der Informatik verwendeten Datenstrukturen. Entscheidungsbäume, Syntaxbäume, Ableitungsbäume, baumartig strukturierte Suchräume, Suchbäume usw. belegen die Vielfalt.

Bäume sind verallgemeinerte Listenstrukturen. Ein Element, sprich Knoten, hat nicht, wie im Falle linearer Listen, nur einen Nachfolger, sondern eine endliche, begrenzte Anzahl von *Kindknoten* (children). In der Regel ist einer der Knoten als *Wurzel* (root) des Baumes ausgezeichnet, dieser ist zugleich der einzige Knoten ohne Vorgänger. Jeder andere Knoten hat einen (unmittelbaren) Vorgänger, der auch *Elternknoten* (parent) genannt wird. Jeder von der Wurzel verschiedene Knoten eines Baumes ist genau durch einen Pfad mit der Wurzel verbunden, d.h. eine Folge $p_0, ...,p_k$ von Knoten eines Baumes, welche die Bedingung erfüllt, dass p_{i+1} Sohn von p_i ist für $0 \leq i < k$, heißt Pfad mit Länge k, der p_0 mit p_k verbindet.

Bäume kann man als spezielle planare, zyklenfreie Graphen auffassen. Die Knoten des Baumes sind die Knoten des Graphen. Je zwei Knoten p und q sind durch eine Kante miteinander verbunden, wenn q *child* von p und damit p *parent* von q ist. Ist unter den *children* eines Knotens eine Anordnung definiert, so dass man vom ersten, zweiten, dritten usw. *child* eines Knotens sprechen kann, ist der Baum geordnet. Dies ist jedoch nicht mit der Ordnung des Baumes zu verwechseln – darunter versteht man die maximale Anzahl von *children* eines Knotens (vgl. [Ottm 02/1], S. 251).

Bäume finden ihre Bedeutung in der Speicherung von Schlüsseln mit mindestens drei wichtigen zugehörigen Operationen wie Suchen nach einem in einem Baum gespeicherten Schlüssel, Einfügen und Löschen eines neuen Knotens mit gegebenem Schlüssel. In manchen Anwendungen treten praktisch keine Einfügungen und Entfernungen von Knoten auf, sondern das Suchen ist die bei weitem überwiegende Operation. Dann kann man einen statischen Suchbaum konstruieren und dabei gegebenenfalls unterschiedliche Suchhäufigkeiten für verschieden Schlüssel festlegen.

Diese Strukturen finden nun auch Verwendung im Umgang mit der Dynamik von Geodaten. Sie eignen sich zur Speicherung von Punkten im eindimensionalen Raum derart, dass sich die für Punkte typischen Operationen Suchen, Einfügen und Entfernen effizient ausführen lassen. Diese Suchbäume haben zum Ziel, Punkte im 2-, 3- - allgemein im k-dimensionalen Raum - so abzuspeichern, dass die für Punkte typischen Operationen gut unterstützt werden.

Exkurs: Binärbäume

Geordnete Bäume der zweiten Ordnung nennt man binäre Bäume oder Binärbäume. Da die Menge der Knoten eines Baumes stets als endlich vorausgesetzt wird, muss es Knoten geben, die keine *children* haben. Diese Knoten werden üblicherweise als Blätter bezeichnet; alle anderen Knoten nennt man innere Knoten.

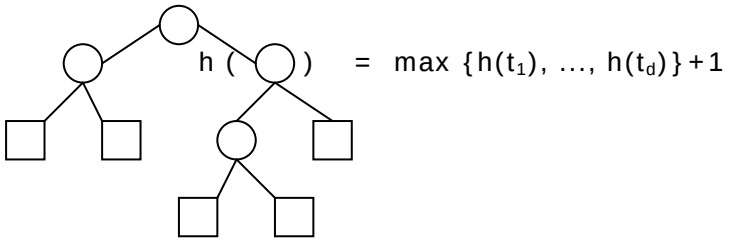


Abb. 5.2: Beispiel eines Binärbaumes (vgl. [Ottm 02/1], S. 253)

Die *Höhe* (h) eines Baumes ist der maximale Abstand eines Blattes von der Wurzel. Der Binärbaum in Abb. 21 hat die Höhe 3. Die Höhe kann rekursiv definiert werden (siehe Abb.21). Die *Tiefe* eines Knotens ist sein Abstand zur Wurzel, d.h. die Anzahl der Kanten auf dem Pfad von diesem Knoten zur Wurzel. Knoten gleicher Tiefe faßt man zu *Niveaus* (level) zusammen. Bäume heißen vollständig, wenn sie auf jedem Niveau die maximal mögliche Knotenzahl besitzen und sämtliche Blätter dieselbe Tiefe haben.

5.2.1 2D-Bäume

Zweidimensionale (2D-) Bäume sind dynamische, anpassbare Datenstrukturen, die Binärbäumen ähneln. Sie teilen einen geometrischen Raum für Anwendungen mit Bereichssuche und anderen räumlichen Problemen auf. Es werden Punkte in den Knoten des binären Suchbaums gespeichert und dabei die Koordinaten y und x der Punkte in alternierender Reihenfolge als Schlüssel benutzt. Zum Einfügen eines Knotens wird der gleiche Algorithmus verwendet wie bei normalen binären Suchbäumen, nur dass an der Wurzel die y-Koordinate verwendet wird – gehe nach links, wenn die y-Koordinate des einzen Punktes kleiner ist als der Punkt an der Wurzel, andernfalls gehe nach rechts. Auf dem nächsten Niveau die x-Koordinate, dann auf dem folgenden wieder die y-Koordinate usw., bis der äußere Knoten erreicht wird.

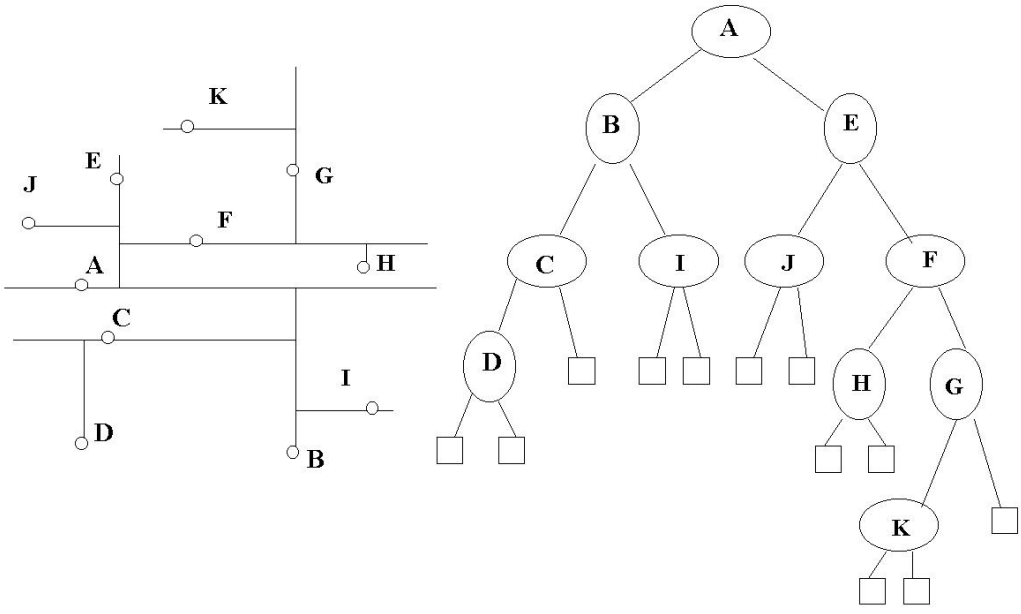


Abb. 5.3: Zerlegung mittels 2D-Baumes (vgl. [Sed 92/01], S. 435)

Jeder äußere Knoten des Baumes entspricht einem bestimmten Rechteck in der Ebene. Jedes Gebiet entspricht einem äußeren Knoten im Baum. Jeder Punkt liegt auf einem horizontalen oder vertikalen Geradenabschnitt, der die Zerlegung definiert, die bei diesem Punkt im Baum vorgenommen wurde. Die Konstruktion eines 2D-Baumes aus N zufälligen Punkten erfordert im durchschnittlichen Fall O(2N ln N) Vergleiche.

Für zufällig verteilte Punkte haben 2D-Bäume die gleiche Leistungsfähigkeit wie binäre Suchbäume. Beide Koordinaten spielen die Rolle zufälliger Schlüssel. Damit die Suche funktioniert, muss bei der Initialisierung des Programmabschnitts, welcher den Baum durchläuft, sorgfältig auf die Anfangsbedingungen geachtet werden, da andernfalls lästige Komplikationen auftreten, wenn nach x-Koordinaten gesucht wird, während der Baum y-Koordinaten hat und umgekehrt. Für die Bereichssuche wird der Punkt bei jedem Knoten mit dem Bereich längs der Dimension verglichen, die benutzt wurde, um die Ebene dieses Knotens zu zerlegen. Für die Bereichssuche mit einem 2D-Baum sind ungefähr O(r + log n) Schritte erforderlich, um in einem n Punkte enthaltenden Gebiet r Punkte in sinnvollen Bereichen zu finden (vgl. [Sed 92/01], S. 439).

Laut Sedgewick [Sed 92/01] wurde dieses Verfahren noch nicht umfassend analysiert, und die angegebene Komplexität ist eine Vermutung, die ausschließlich auf empirischen Untersuchungen beruht. Die Leistungsfähigkeit und Analyse ist stark von dem verwendeten Bereichstyp abhängig. Im Vergleich sind 2D-Baum und Gitterverfahren gleichwertig. Der 2D-Baum ist jedoch weniger von der "Zufälligkeit" in der Punktmenge abhängig.

5.2.2 KD-Bäume

2D-Bäume lassen sich unmittelbar auch auf mehr als zwei Dimensionen verallgemeinern. Bewegt man sich im Baum abwärts, durchläuft man die Dimensionen einfach zyklisch, so wie es auch oben am zweidimensionalen Baum erläutert wurde, in dem man zwischen x- und y- Koordinate wechselt. Ein Beispiel für einen KD-Baum stellt die binäre Teilung dar, wobei jeweils zwei Folgeblätter angelegt werden.

Der Konstruktions-Algorithmus ist für die Ebene einfach zu realisieren. Die Wurzel (root) repräsentiert die gesamte Punktemenge P. Durch eine vertikale Teilungslinie wird P an der Wurzel in zwei Teilmengen zerlegt. Die Teilungslinie kann entweder streng durch die geometrischen Halbierungspunkte gelegt werden oder Schwerpunkte der Objektbelegung berücksichtigen.

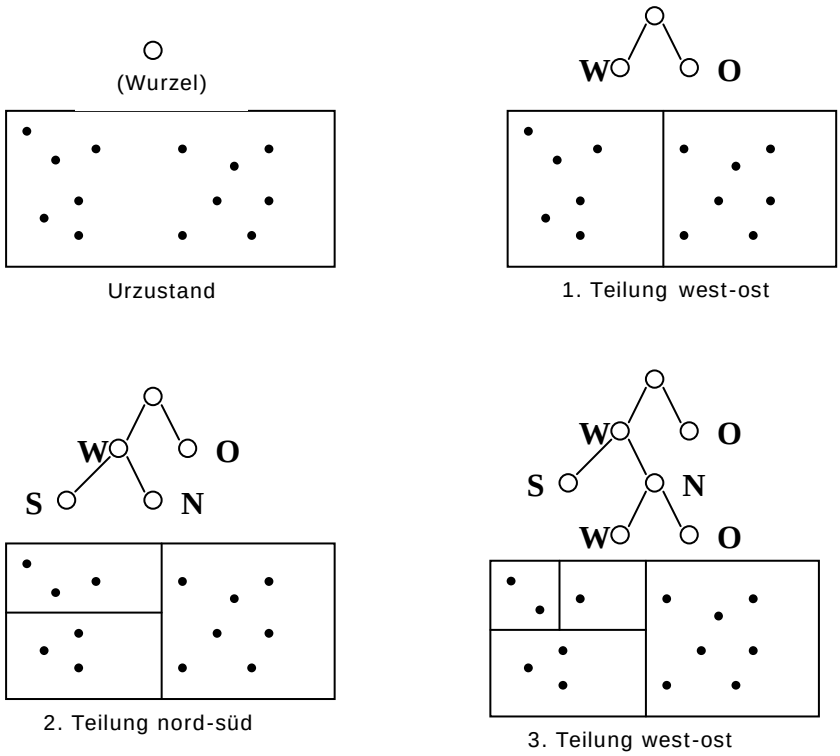


Abb. 5.4: Aufteilung nach den Regeln eines KD-Baumes(vgl. [Bar 95/1], S. 222)

Im Beispiel der Abbildung 5.4 geschieht das Anlegen der zwei Folgeblätter im KD-Baum jeweils alternierend, einmal längs der Nordrichtung, das andere Mal längs der Ostrichtung, d. h. bei root wird P basierend auf der ersten Koordinate der Punkte geteilt. Die dadurch entstandenen Kindknoten von root (Tiefe 1) werden basierend auf der zweiten Koordinate, die Knoten der Tiefe 2 auf der dritten Koordinate und so weiter bis zur Tiefe d –1 partitioniert. Bei Tiefe d wird dieses Verfahren wiederholt, begonnen bei der ersten Koordinate der Punkte. Die Rekursion ist beendet, wenn nur ein Punkt übrig bleibt, welcher in einem Blattknoten gespeichert wird.

Ein d-dimensionaler KD-Baum für eine Menge von *n* Punkten gleicht einem Binärbaum mit *n* Blattknoten und benötigt darum für die Speicherung O(d•n) Operationen. Die Konstruktionszeit liegt bei O(d•n log n). Knoten in einem d-dimensionalen KD-Baum sind mit einem Gebiet (Ebene) verbunden.

5.2.3 Quadtree

Ein Quadtree ist ein gerichteter Baum (ähnlich dem Binärbaum), in welchem jeder interne Koten 4 Kinder besitzt. Jeder Knoten entspricht einem Quadrat. In freier Übersetzung nach Berg (vgl. [Berg 97/1], S. 292): Wenn ein Knoten *v* Kinder hat, dann sind die zugehörigen Quadrate die 4 Quadranten des Quadrates von *v* – daher der Name Quadtree.

In seinem Werk "*Applications of spatial data structures: computer graphics, image processing and GIS*" beschreibt der Verfasser H. Samet den Quadtree-Algorithmus ausführlich. Er wurde grundsätzlich für den zwei- und dreidimensionalen (Octrees) Raum konzipiert. Laut H.Samet wird der Begriff Quadtree benutzt, um eine Klasse von hierarchischen Datenstrukturen zu beschreiben, deren Prinzip auf einer rekursiven Zerlegung (Dekomposition) des Raumes basiert (vgl. [Samet 95/1], S. 2). Es wurden verschiedene Varianten definiert, die sich beispielsweise in den Regeln des Dekompositionsprozesses unterscheiden oder einen bestimmten Datentyp repräsentieren, wie Punkte, Linien, Areas, Curves, Surfaces (Oberflachen), Körper.

Die bekannteste Repräsentation ist das *Region-Quadtree*. Es stellt eine pixelorientierte 2-dimensionale Datenstruktur dar. Eine Figur wird als Bitcodierung dargestellt, d.h. jedes Pixel der Figur hat den Wert = 1, alle anderen Pixel haben den Wert = 0. Deckt die Figur nicht die gesamte Fläche (image array) ab, dann wird sie in Quadranten, Subquadranten zerlegt, solange bis ein Block lediglich nur aus 1 oder aus 0 besteht, d. h. ein Block liegt innerhalb oder außerhalb der Figur.

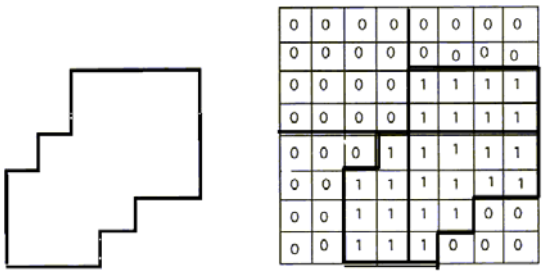


Abb. 5.5: Region-Quadtree – Set(1) = innen, Set(0) = außen (vgl. [Samet 95/1], S. 3)

Weitere Formen sind das

- *Point-Quadtree* – es ist dem schon erklärten KD-Baum ähnlich.
- *PR-Quadtree* – *P* steht für Point und *R* für Region. Es ist wie das Region-Quadtree strukturiert und repräsentiert Punktdaten, kann aber auch für Regionen benutzt werden, die aus einer Kollektion von Polygonen besteht.
- *PM-Quadtree* – Regionen werden durch Spezifikation ihrer Ränder (boundaries) gespeichert. Auch hier gibt es unterschiedliche Ausprägungen.

Alle diese Varianten und andere sind ausführlich in [Samet 95/1] erläutert. In einer freien Übersetzung der Ausführungen in [Berg 97/1] ab Seite 291 wird die Bildung eines Quadtrees für eine Punktmenge und dazu gehörende Algorithmen erklärt:

Wie schon erwähnt besitzt jeder interne Knoten eines Quadtrees vier Kinder. Die Kinder entsprechen je einem Quadrat. Diese zugehörigen Quadrate wiederum sind die Quadranten des übergeordneten Knotens. Weil die Quadrate der Blätter eine Teilung des Quadrates der Wurzel vornehmen, nennt man diese Teilung *Quadtree-Teilung.* Berg u.a. verdeutlichen mit der Abb.5.6, dass ein Knoten der Tiefe *i* mit einem Quadrat der Kantenlänge $s/2^i$ verbunden ist.

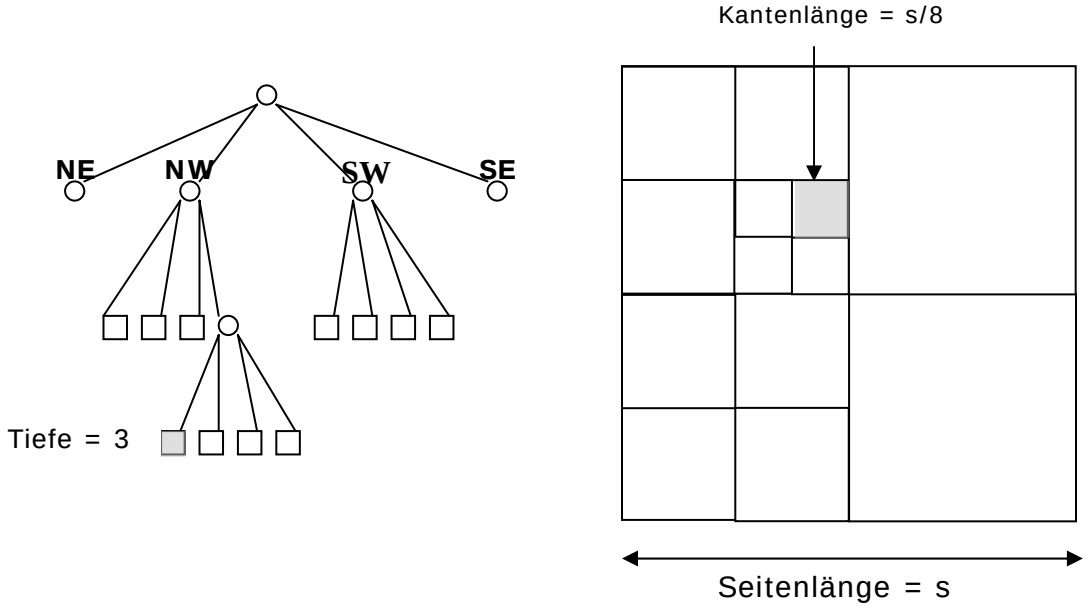


Abb. 5.6: Ein Quadtree und seine zugehörige Teilung (vgl. [Berg 97/1], S. 292/294)

Um die Quadranten festzulegen, erhalten die Kindknoten der Wurzel (root) die Bezeichnungen NE, NW, SW, SE. NE steht für Nord-Ost-Quadrant, NW für Nord-West-Quadrant usw. Das Ergebnis einer Quadtree-Teilung ist die Form eines Quadrates. Die 4 Punkte (positions) der Ecken nennt man *corner vertices*. Die Linie von einer Ecke zur anderen nennt man *Seite* des Quadrates (sides of the square). Die Kanten einer Quadtree-Teilung, welche im Rand des Quadrates enthalten ist, nennt man *Kanten* des Quadrates (edges of the square). Es ist also möglich, dass eine Seite wenigstens eine Kante aber auch mehrere Kanten enthalten kann. Zwei Quadrate sind Nachbarn, wenn Sie eine Kante teilen.

Erzeugung des Quadtrees:

Quadtrees können zum Speichern unterschiedlicher Daten benutzt werden. Beim Einfügen einer Punktemenge wird das rekursive Teilen der Quadrate so lange vorgenommen, bis nur noch ein Punkt in einem Quadrat enthalten ist.

Die Definition des Quadtree einer Menge von Punkten P in einem Quadrat σ ist wie folgt: $\sigma := [x_\sigma : x'_\sigma] \times [y_\sigma : y'_\sigma]$.

Wenn $P \leq 1$ ist, besteht das Quadtree nur aus einem einzigen Blattknoten, in welchem P und das Quadrat σ gespeichert werden.

Anderenfalls werden die vier Qudranten σ_{NE}, σ_{NW}, σ_{SW} und σ_{SE} des Quadrates σ angelegt mit

$x_{mid} := (x_\sigma + x'_\sigma)/2$ und $y_{mid} := (y_\sigma + y'_\sigma)/2$ und definiert als

$$P_{NE} := \{p \in P : p_x > x_{mid} \text{ and } p_y > y_{mid}\},$$
$$P_{NW} := \{p \in P : p_x \leq x_{mid} \text{ and } p_y > y_{mid}\},$$
$$P_{SW} := \{p \in P : p_x \leq x_{mid} \text{ and } p_y \leq y_{mid}\},$$
$$P_{SE} := \{p \in P : p_x > x_{mid} \text{ and } p_y \leq y_{mid}\}.$$

Das Quadtree besteht nun aus einer Wurzel *v*, in welcher das Quadrat σ gespeichert ist, d.h. das Quadrat ist gespeichert in *v* mit σ(*v*). Zusätzlich hat *v* vier Kinder:

- das NE-Kind ist die Wurzel (root-cell) eines Quadtrees für eine Punktemenge P_{NE} im Quadrat σ_{NE},
- das NW-Kind ist die Wurzel (root-cell) eines Quadtrees für eine Punktemenge P_{NW} im Quadrat σ_{NW},
- das SW-Kind ist die Wurzel (root-cell) eines Quadtrees für eine Punktemenge P_{SW} im Quadrat σ_{SW},
- das SE-Kind ist die Wurzel (root-cell) eines Quadtrees für eine Punktemenge P_{SE} im Quadrat σ_{SE}.

Der Grund, warum man in der Definition von P_{NE}, P_{NW}, P_{SW}, P_{NE}, "kleiner-gleich" und "größer als" verwendet, ist eine vertikale Teilungslinie zu den linken zwei Quadranten und eine horizontale Teilungslinie zu den unteren Quadranten.

Jeder Knoten *v* eines Quadtrees speichert sein Quadrat σ(*v*). Alternativ würde es ausreichen, nur das Quadrat der Wurzel des Quadtree-Baumes zu speichern. Wenn man tiefer in den Baum geht, könnte man durch Berechnung das Quadrat des durchlaufenden Knotens erhalten. Dieses Verfahren benötigt weniger Speicherplatz und wird bei Anfragen an das Quadtree verwendet.

Die rekursive Definition eines Quadtrees übersetzt in einen rekursiven Algorithmus:
Teile das aktuelle Quadrat in vier Quadranten, partitioniere die Punktemenge demgemäß und erstelle rekursiv Quadtrees für jeden Quadranten mit seiner übereinstimmenden Punktemenge. Die Rekursion ist beendet, wenn eine Punktemenge weniger als zwei Punkte enthält.

Lediglich das Quadrat mit welchem man ein Quadtree beginnt zu konstruieren, ist vom rekursiven Algorithmus nicht abzuleiten. Manchmal ist das Quadrat durch mögliche Eingabewerte zu ermitteln. Wenn dies nicht der Fall ist, wird das kleinste die Punktemenge einschließende Quadrat berechnet, anhand der Extrempunkte in x- und y-Richtung.

In jedem Schritt wird ein Quadrat, welches eine Punktemenge enthält, in vier kleinere Quadrate geteilt. Dies bedeutet nicht, dass die Punktemenge gut verteilt ist. Es ist möglich, dass alle Punkte im gleichen Quadranten liegen, was den Baum unausgeglichen macht. Es ist nicht möglich, die Größe und Tiefe eines Quadtrees in Abhängigkeit auf die Anzahl der Punkte zu bestimmen. Die Tiefe eines Quadtrees ist abhängig vom Abstand zwischen den Punkten und der Größe des initialisierten Quadrates.

Lemma: Die Tiefe eines Quadtrees für eine Punktemenge P in der Ebene beträgt meist $\log(s/c)+\frac{3}{2}$, wobei c den kleinsten Abstand zwischen zwei Punkten aus P und s die Seitenlänge des initialisierten Quadrates darstellt, welches P beinhaltet.

Daraus folgt, dass die Tiefe des Quadtrees genau eine Teilungsstufe mehr hat als die maximale Tiefe eines inneren Knotens im Baum (siehe Abb.5.6).

Die Größe und Konstruktionszeit sind abhängig von der Tiefe des Quadtrees und der Anzahl der Punkte.

Theorem: Ein Quadtree der Tiefe (depth) d, welches eine Menge von n Punkten speichert, hat $O((d+1)n)$ Knoten und kann in einer Zeit von $O((d+1)n)$ gebildet werden.

Die meiste Zeit des rekursiven Algorithmus wird für die Verteilung der Punkte auf die Quadranten des aktuellen Quadrates benötigt.

Neighbor finding - Algorithmus

Das Finden von Nachbarn (neighbor finding) ist eine oft benötigte Operation. Gegeben ist ein Knoten *v* und eine Richtung – Norden, Osten, Süden, Westen . Es soll ein Knoten *v'* gefunden werden, wo *σ(v')* benachbart zu *σ(v)* ist, und die Richtung zurückgegeben werden. Der gegebene Knoten ist ein Blattknoten (wird im weiteren auch als *leaf* bezeichnet) und es soll auch ein Blattknoten zurückgegeben werden. Diese Operation ist zwar vergleichbar mit dem Finden eines anliegenden Quadrates an ein schon bekanntes Quadrat bei der Quadtree-Teilung, jedoch ist hier der Algorithmus etwas anders.

Der gegebene Knoten *v* kann ein interner Knoten sein und wenn *v* und *v'* die gleiche Tiefe haben, versucht der Algorithmus den Knoten *v'* und das benachbarte Quadrat *σ(v')* von *σ(v)* in der gegebenen Richtung zu finden. Wenn dort kein passender Knoten gefunden wird, wird der tiefste Knoten, der an das Quadrat angrenzt, zurückgegeben. Es gibt die Möglichkeit, dass kein angrenzendes Quadrat in der gegebenen Richtung vorkommt, z.B. wenn eine Kante von *σ(v)* in einer Kante des initialisierten Quadrates enthalten ist . Dann wird NULL zurückgegeben.

Der Algorithmus läuft folgendermaßen ab (vgl. [Berg 97/1], S. 295):
Der nördliche Nachbar von *v* soll gefunden werden. Wenn *v* ein SE- oder SW-leaf seines Elternknotens ist, ist das Finden des nördlichen Nachbarn leicht. Es ist entweder das NE- oder NW-leaf des gleichen Elternknotens. Wenn *v* selbst ein NE- oder NW-leaf seines Elternknotens ist, geht man wie folgt vor. Rekursiv wird der nördliche Nachbar, μ, des Elternknotens von *v* gesucht. Wenn μ ein interner Knoten ist, dann ist der nördliche Nachbar von *v* ein leaf von μ. Wenn μ ein leaf ist, dann ist μ der gesuchte Nachbar selbst.

Dieser Algorithmus gibt nicht immer einen leaf zurück. Wenn ein leaf in jedem Fall zurückgegeben werden soll, muss man tiefer im Baum von dem mit dem Algorithmus gefundenen Knoten ausgehend suchen, immer in Hinsicht darauf, einen südlichen Blattknoten zu finden. Der Algorithmus benötigt bei jedem rekursiven Aufrufen eine Zeit von O(1). Bei jedem Aufruf vermindert sich die Tiefe des übergebenen Knotens um eins. Aus diesem Grund ist die Laufzeit linear in der Tiefe des Quadtree.

Theorem: *T* ist ein Quadtree der Tiefe *d*. Ein Nachbar eines gegebenen Knotens *v* in *T* mit gegebener Richtung, wie oben definiert, kann in der Zeit O*(d+1)* zurückgegeben werden.

Balancieren des Quadtrees

Ein Quadtree kann ziemlich unausgeglichen sein, beispielsweise, wenn die Punktmenge nicht homogen verteilt ist. Darum können große Quadrate benachbart von vielen kleinen Quadraten sein. Dies ist vor allem unerwünscht, wenn Anwendungen zur Netzgenerierung verwendet werden. Aus diesem Grund wird ein Quadtree ausbalanciert.

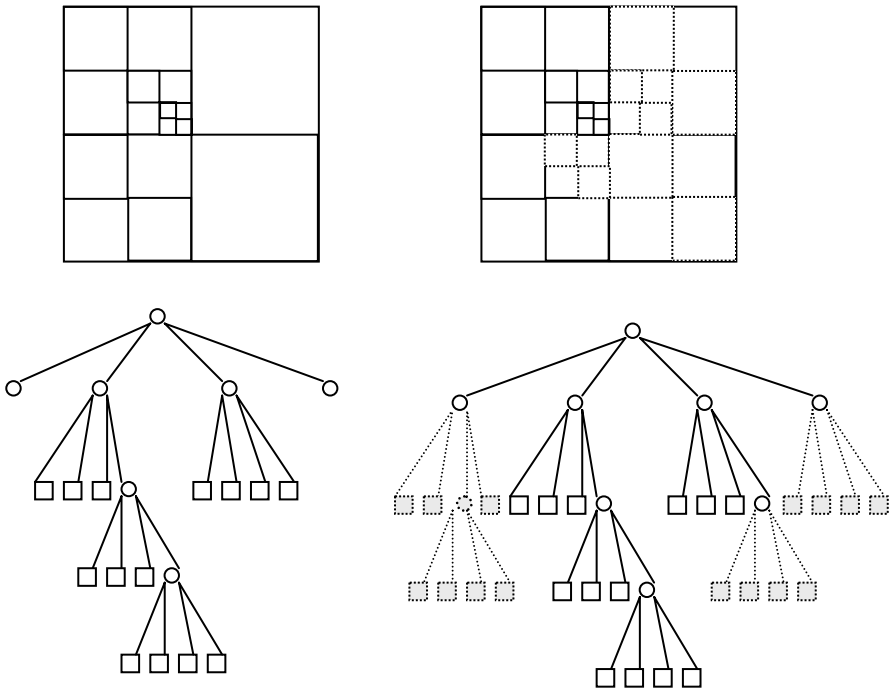


Abb. 5.7: Ein Quadtree und seine ausbalancierte Version (vgl. [Berg 97/1], S. 296)

Eine Quadtree-Teilung ist ausgeglichen, wenn sich zwei Nachbar-Quadrate nicht mehr als um den Faktor 2 unterscheiden.

Definition: Ein Quadtree wird als ausgeglichen (balanced) bezeichnet, wenn seine Teilung ausgeglichen (balanciert) ist. In einem ausgeglichenen Quadtree können sich zwei Blattknoten, deren Quadrate Nachbarn sind, höchstens um eine Ebene in der Tiefe unterscheiden.

Mit nachfolgend erklärtem Algorithmus kann ein Quadtree balanciert werden (vgl. [Berg 97/1], S. 295):

Zuerst muss festgestellt werden, welches Quadrat geteilt werden muss. Man muss also feststellen, welches Quadrat $\sigma(\mu)$ benachbart zu einem Quadrat kleiner der halben eigenen Größe ist. Dafür benutzt man den schon beschriebenen "neighbor-finding"-Algorithmus. Man nimmt an, dass nach einem nördlichen Nachbarn von $\sigma(\mu)$ gesucht wird, der kleiner als die Hälfte der eigenen Größe von $\sigma(\mu)$ ist. Wenn der Algorithmus einen Knoten zurück gibt, der einen SW- oder SE-Kindknoten hat, der nicht Blattknoten ist, gibt es ein solches Quadrat und $\sigma(\mu)$ muss geteilt werden

Danach wird festgestellt, ob $\sigma(\mu)$ Nachbarn hat, die nun geteilt werden müssen. Wieder kommt der "neighbor-finding"-Algorithmus zum Einsatz, wenn beispielsweise $\sigma(\mu)$ einen Nachbarn im Norden hat und die Funktion einen Knoten zurückgibt, dessen Quadrat größer ist als $\sigma(\mu)$.

In Abb. 5.7 erkennt man, dass die Komplexität eines balancierten Quadtrees stark zugenommen hat. Nicht nur, dass das Splitten größerer benachbarter Quadrate viel Zeit benötigt, das Splitten pflanzt sich auch fort. Manchmal sieht es so aus, als ob ein Quadrat nicht gesplittet werden muss, weil seine Nachbarn die richtige Größe haben, jedoch mussten die Nachbarn gesplittet werden, so daß am Ende auch σ geteilt werden muß. Trotzdem kann das Ausbalancieren effizient durchgeführt werden. Weitergehende Informationen und Algorithmen findet man in [Samet 95/1] und spezieller Literatur zum Thema "Computational Geometry".

5.3 Zusammenfassung

Die Strukturen vorliegender Daten und geplante weiterführende Operationen sind oft ausschlaggebend für die Wahl des Indizierungs-Algorithmus. In diesem Kapitel wurde sich mit der Indizierung von 2-dimensionalen Raumdaten beschäftigt. Es wurde gezeigt, dass einige Indizierungsverfahren, z.B. das Gitterverfahren, von der homogenen Verteilung der Daten abhängig sind. Manche Verfahren sind statisch, andere dynamisch. Einige Verfahren, wie KD-Bäume oder diverse Quadtree-Formen, eignen sich besonders für einzelne Datenstrukturen von Raumdaten, beispielsweise Punktdaten oder Linien.

Wird eine spätere Vernetzung angestrebt, benötigt man dafür ein Verfahren, welches beeinflussend auf eine homogene Verteilung der Punkte innerhalb der interessierenden Fläche einwirken kann.

Diese und weitere Aspekte müssen für eine funktionierende Indizierung in Betracht gezogen werden. Neben der Festlegung auf eine Methode besteht auch die Möglichkeit, einzelne Verfahren miteinander zu kombinieren. Im Fall des Netzgenerators des STDSS wurde das Gitterverfahren als Grundlage für den Quadtree-Algorithmus verwendet. Dies wird in Kapitel 7 eingehend erläutert.

6. Netzgenerierung

6.1 Grundbegriffe der Netzgenerierung

Der Netzbegriff (mesh) geht zumeist mit dem Begriff Gitter einher. Homann (vgl Hom 05/1]) beschreibt ein Gitter als Zerlegung eines Raumes in disjunkte[13] Subräume. Eine andere Beschreibung findet man von Burkhard (vgl. [Burg 05/1]):

Definition: Die Netzgenerierung ist ein Prozess, bei dem ein System in viele kleine Teilsysteme, die Elemente genannt werden und sich nicht überlappen dürfen, zerteilt wird. Ebene Systeme können in Dreieckelemente oder Viereckelemente, räumliche Gebilde in Tetraeder[14] oder Hexaeder [15]zerteilt werden.

Netze (Gitter) lassen sich in drei Arten kategorisieren: strukturierte Netze, unstrukturierte Netze und hybride Netze.

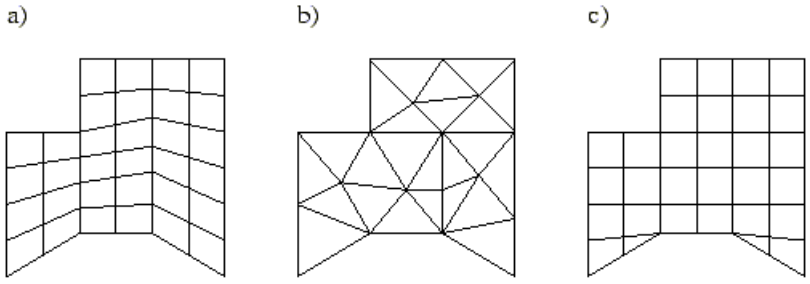


Abb. 6.1: a) strukturiertes, b) unstrukturiertes, c) hybrides Netz (vgl. [Burg 05/1]).

Strukturierte Netze

Bei strukturierten Netzen ist die Anzahl der an einen Knoten angrenzenden Elemente für jeden inneren Knoten gleich und entsprechend der optimalen Anzahl (vgl. [Burg 05/1]). Optimale Elementanzahl, beispielsweise für ein Viereck, ist die Vier. Strukturierte Netze können aus ein-, zwei- oder dreidimensionalen Elementen bestehen. Die Eckpunkte der Elemente liegen in einer strukturierten Ordnung vor. Eine Abbildung des Netzes auf ein Netz mit regelmäßigen Elementen ist möglich.

[13] disjunkt = getrennt
[14] Tetraeder = von vier gleichseitigen Dreiecken begrenzter Körper
[15] Hexaeder = Würfel (grch. hex ="sechs", hedra = "Grundfläche")

Unstrukturierte Netze

In unstrukturierten Netzen ist die Anzahl der Nachbarknoten bzw. Nachbarelemente für die einzelnen Knoten unterschiedlich (vgl. [Burg 05/1]). Eine wesentlich größere Flexibilität in Hinsicht auf die Anpassung an die Geometrie als auch bei Verfeinerungen zeichnen unstrukturierte Netze im Vergleich zu strukturierten Netzen aus. Wie schon in 4.2.5 erklärt, werden beliebig verteilte Punkte durch Linien verbunden, so dass ein Netz aus Dreiecken entsteht (siehe Abb. 6.1). Die beiden wichtigsten Methoden zur Erzeugung unstrukturierter Netze sind das Advancing-Front-Verfahren und die Delaunay-Triangulierung .

Hybride Netze

Zur Erzeugung dieser Netze werden Vorgehensweisen der strukturierten und der unstrukturierten Netzgenerierung kombiniert (vgl. [Burg 05/1]). Sie bestehen aus unterschiedlichen Elementen. Zweidimensionale Netze bestehen aus Dreiecken und Vierecken (siehe Abb. 6.1), dreidimensionale Netze aus Tetraedern und Hexaedern. Die resultierenden Netze sind daher unstrukturiert. Hybride Netze bieten im Vergleich zu strukturierten Netzen eine wesentlich größere geometrische Flexibilität. Weitergehende Informationen findet man in einschlägiger Literatur.

Für die Generierung unstrukturierter Gitter wird der Topologiebegriff erweitert. Hier unterscheidet man die Elementtopologie und die Gittertopologie:

Elementtopologie

Definition: Als *Elementtopologie* bezeichnet man die Beschreibung eines Gitterelementes durch seine Knoten und Kanten (vgl. [Hom 05/1]).

Abb. 6.2 zeigt ein Beispiel für elementtopologische Konventionen (vgl. [Hom 05/1]). Ein Dreieckselement wird mit den Knoten i, j, k und den drei Kanten I, J, K beschrieben. Den Kanten ist eine Orientierung so zugeschrieben, dass sie das Dreieck gegen den Uhrzeigersinn umlaufen. Die Kante I liegt gegenüber dem Knoten i usw. Dadurch können die Knoten der Kanten durch zyklischen Austausch bestimmt werden.

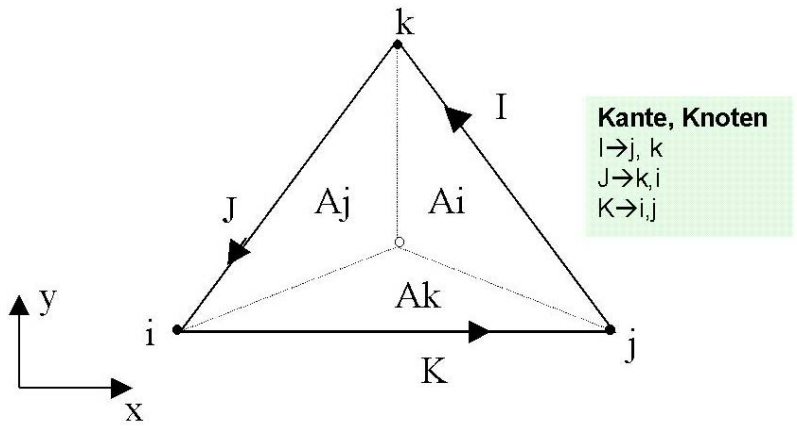


Abb. 6.2: Elementtopologische Konventionen (vgl. [Hom 05/1])

Gittertopologie

In unstrukturierten Gittern lassen sich die Nachbarschaftsbeziehungen nicht explizit angeben. Ihre Verwaltung erfordert daher einen hohen Aufwand. Nachbarschafts-beziehung sind entweder durch geeignete Verfahren zu bestimmen oder explizit zu speichern (vgl. [Hom 05/1]).

Definition: Die Beschreibung der Nachbarschaftsbeziehungen in einem Gitter wird als *Gittertopologie* bezeichnet.

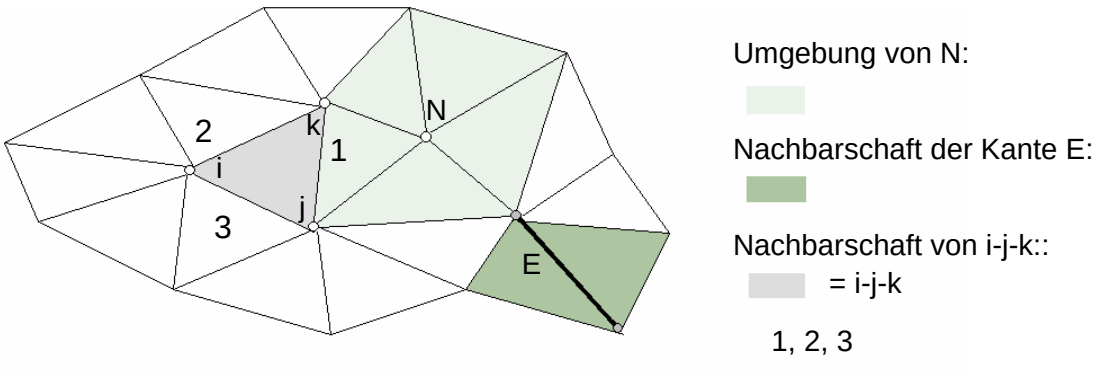


Abb. 6.3: Gittertopologie (vgl. [Hom 05/1])

Die Nachbarschaft einer Kante wird von maximal zwei Dreiecken gebildet (siehe Abb.6.3). Randkanten haben nur ein Nachbardreieck.

Definition: Die Menge der Dreiecke, deren Schnittmenge mit einem Dreieck i-j-k eine Kante von i-j-k ist, wird als die Nachbarschaft des Dreiecks i-j-k bezeichnet (vgl. [Hom 05/1]).

Demnach umfaßt die Nachbarschaft eines Dreiecks maximal drei Dreiecke.

Definition: Die Menge der Dreiecke, die einen Knoten *N* als Schnittmenge besitzen, wird als Umgebung (patch) von *N* bezeichnet [Hom 05/1].

Die Umgebung eines Knotens kann beliebig viele Dreiecke umfassen. Daraus ergibt sich der Begriff Umgebung eines Dreiecks [Hom 05/1]:

Definition: Besitzt ein Dreieck die Knoten i, j, k, so ist die Umgebung des Dreiecks die Vereinigung der Umgebungen von i, j und k ohne das Dreieck selbst.

Konvexität

Als *konvex* (lat.: *convexus* gewölbt, gerundet) bezeichnet man Formen (Flächenteile, Linien), die nach außen gewölbt sind.

In der Geometrie findet man den Begriff der konvexen Menge:

Definition: Eine Punktmenge S in der Ebene wird als konvex bezeichnet, wenn sich zwei Punkte $p,q \in S$ durch eine Strecke verbinden lassen, die ganz in S enthalten ist (vgl.[Berg 97/1], S. 2).

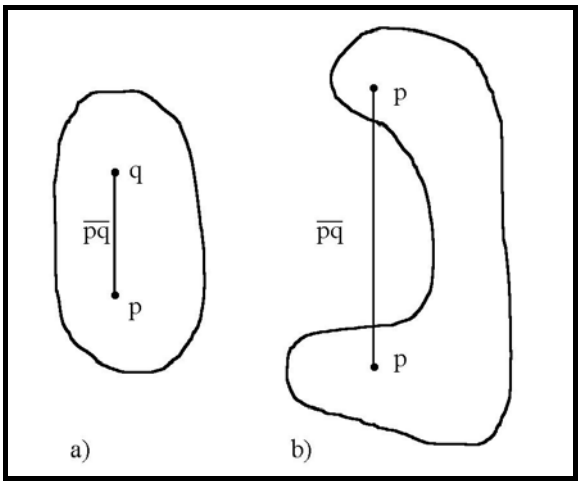


Abb. 6.4: Beispiel zur konvexen Menge: a) konvex, b) nicht konvex

Konvexe Hülle

Wenn *N* Punkte gegeben sind, so bilden einige von Ihnen ein konvexes Polygon, in dem die übrigen enthalten sind.

Definition: Die konvexe Hülle einer Menge von Punkten in der Ebene ist als das kleinste konvexe Polygon definiert, das alle diese Punkte enthält. Man nennt dies auch den kürzesten Pfad, der die Punkte umschließt (vgl. [Sed 92/01], S. 412).

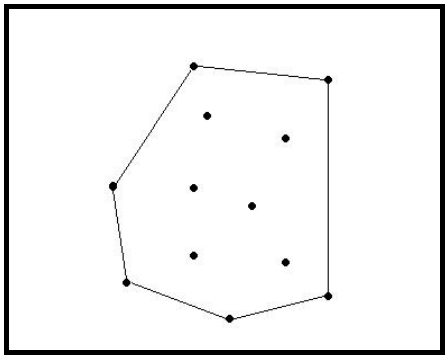


Abb. 6.5: Konvexe Hülle

Der Hülle in Abb 6.5 gehören sechs Punkte an. Es ist möglich, dass die konvexe Hülle nur drei Punkte enthält (großes Dreieck) oder ihr gehören alle Punkte an.

6.2 Triangulation

Ein Verfahren, um aus einer vorgegebenen Punktmenge und eventuell gegebenen Kanten ein Dreiecksnetz zu erzeugen, nennt man Triangulation. Die Dreiecke (Triangles) entstehen durch Verbinden der Punkte durch Kanten. Es sollten grundlegende Regeln bei einer Triangulation beachtet werden (vgl. [UniBerkl 05/1]):

- Jedes Dreieck der Triangulation *T* hat drei *Vertices* aus einer Punktmenge *V* für seine Ecken und schneidet keine anderen *Vertices* aus *V*.
- Das Innere (*Interior*) der Dreiecke schneidet sich nicht.
- Dreiecke dürfen sich nur an ihren *Kanten* und *Vertices* schneiden.
- Jeder Punkt der konvexen Hülle von *V* wird bei der Dreiecksbildung verarbeitet.

Daraus folgt, dass die Triangulation *T* eine Teilung der konvexen Hülle von *V* ist.

Eine simple Möglichkeit der Triangulation wäre ein willkürliches Verbinden der Punktmenge unter Berücksichtigung der eben genannten Grundregeln. Im Hinblick auf die Weiterverwendung des triangulierten Netzes zu numerischen Berechnungen und Simulationen spielt die Qualität der Vernetzung aber eine entscheidende Rolle. Dabei ist das Anstreben von möglichst gleichseitigen Dreiecken für eine stabile und möglichst genaue Lösung entscheidend. Außerdem muss das Netz eine größere Auflösung (kleinere Elemente) in Regionen besitzen, in welchen eine höhere Dichte an Informationen vorliegt und eine grobe Auflösung (größere Elemente) in Gebieten mit geringerer Informationsdichte. Es existieren zahlreiche Verfahren der Triangulation von Knotenmengen, die zu verschiedenen Dreiecksnetzen führen können.

6.2.1 Triangulierungskriterien

Bei einer Triangulation von 4 Punkten gibt es bereits mehr als ein Dreiecksnetz. Dabei ist bereits für diesen einfachen Fall eine Technik zur Auswahl der geeigneteren Triangulation notwendig. Ein einfaches Kriterium, um zwischen zwei triangulierten Netzen T und Ť zu entscheiden, ist gegeben durch das Kriterium der kürzeren Diagonalen (vgl. [Hoschek 92/1], S.379).

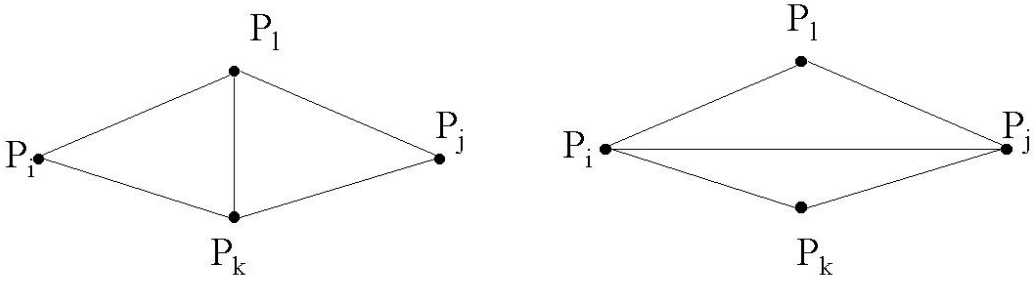


Abb. 6.6: Mögliche Triangulierung von vier Punkten

Definition: T ist eine bessere Triangulierung als Ť (bzgl. des Kriteriums der kürzeren Diagonalen) genau dann, wenn gilt d < d', wobei d die Länge der Diagonalen P_lP_k der Triangulierung T und d' die Länge der Diagonalen P_iP_j der Triangulierung Ť ist.

Dieses Kriterium vermeidet jedoch nicht die Erzeugung langer dünner Dreiecke, die man für die Interpolation vermeiden möchte. Laut Hoschek gibt es mehrere Entscheidungskriterien für eine optimale Triangulierung (vgl. [Hoschek 92/1], S. 380):

- *minimale Kantenlängensumme*: verhindert nicht die Erzeugung langer, dünner Dreiecke (ungünstig bei FEM u. Visualisierung)
- *Max-Min-Winkelkriteriu*m: der kleinste vorkommende Dreieckswinkel wird maximiert
- *Min-Max-Winkelkriteriu*m: der größte vorkommende Dreieckswinkel wird minimiert
- *Max-Min-Radiuskriteriu*m: der kleinste Radius der in die Dreiecke einbeschriebenen Kreise wird maximiert
- *Min-Max-Radiuskriterium:* der größte Radius der in die Dreiecke einbeschriebenen Kreise wird minimiert
- *Max-Min-Flächenkriteriu*m: der kleinste Flächeninhalt der Dreiecke wird maximiert
- *Max-Min-Höhenkriteriu*m: die kleinste Höhe der Dreiecke wird maximiert.

Obwohl die genannten Kriterien voneinander verschieden sind, können sie im speziellen Fall zur gleichen Triangulierung von vier Punkten führen.

6.2.2 Optimale Triangulierung

Mit Hilfe der für Vierecke eingeführten Entscheidungskriterien können auch größere Punktmengen trianguliert werden, z.B. indem, mit einem Viereck beginnend, durch Hinzunahme jeweils eines Punktes, eine optimale Triangulierung erzeugt wird. Aufgrund der aus den Kriterien folgenden nur lokalen Änderung soll auch ein globales Optimum erreicht werden(vgl. [Hoschek 92/1], S. 381):

Definition: Eine Triangulierung *T* heißt *lokal optimal* bzgl. eines Kriteriums K, wenn jedes Viereck, definiert durch je 2 entlang einer gemeinsamen Kante aneinandergrenzende Dreiecke von T, bzgl. *K* optimal trianguliert ist.

Wenn T eine Triangulation von Punktmenge P mit *m* Dreiecken ist, so erhält man 3m Winkel der Dreiecke von T der Größe nach aufsteigend sortiert. Dann ist die Ergebnisliste der Winkel $\alpha_1,\alpha_2,..., \alpha_{3m}$,wenn $\alpha_i \leq \alpha_j$, für i < j. Man nennt A(T) := $(\alpha_1,\alpha_2,..., \alpha_{3m})$ den Winkel-Vektor von T. Wenn T' eine andere Triangulierung der selben Punktmenge P ist und A(T') := $(\alpha'_1,\alpha'_2,..., \alpha'_{3m},)$ der zugehörige Winkel-Vektor, sagt man, dass der Winkel-Vektor von A(T) größer ist als der von T', wenn A(T) vergleichsweise größer ist als A(T') . Anders gesagt, es gilt z.B. A(T) > A(T') genau dann, wenn es einen Index *i* gibt, so dass gilt $\alpha_{i=}\alpha'_i$ für i = 1,....,3m–1 und $\alpha_{3m} > \alpha'_{3m.}$

Man sagt, die Triangulierung T ist winkel-optimal, wenn A(T) > A(T') für alle Triangulationen T' der Punktmenge P (vgl. [Berg 97/1], S. 184) gilt.

Definition: Eine Triangulierung *T* der Punktmenge *P* heißt *global optimal* bzgl. eines Kriteriums *K*, wenn jede andere Triangulierung von *P* ungünstiger als *T* bzgl. *K* ist (vgl. [Hoschek 92/1], S. 381).

Hoschek (vgl. [Hoschek 92/1], S. 381) zeigt, dass das Max-Min-Winkelkriterium von Lawson das einzige bekannte Kriterium ist, bei welchem stets lokale Optima auch globale Optima sind. Die mit dem Max-Min-Winkelkriterium erzeugte Triangulierung ist äquivalent zur Delaunay-Triangulierung.

6.2.3 Delaunay-Triangulation

Die Delaunay-Triangulation einer Punktemenge *P* wurde 1934 von Delaunay eingeführt. Dabei wird ein Netz mit Dreieckselementen erzeugt. Als Nebenbedingung wird das Maximieren des kleinsten Innenwinkels über alle Triangulationen der gegebenen Punktemenge angestrebt. Ist dieses Kriterium, auch Delaunay-Kriterium genannt, für alle Kanten und Dreiecke einer Vernetzung erfüllt, erhält man eine Delaunay-Triangulierung und damit ein hoch qualitatives Dreiecksnetz.

Die Delaunay-Triangulierung hängt geometrisch stark mit dem Voronoi-Diagramm, auch Dirichlet-Parkettierung oder Thiessen-Polygone genannt, zusammen:

Das Voronoi-Diagramm für eine Menge von Punkten in der Ebene teilt die Ebene in Gebiete gleicher nächster Nachbarn ein. Dies wird erreicht durch Antragen der Mittelsenkrechten auf den Verbindungsstrecken von einem Mittelpunkt ausgehend zu den umliegenden Punkten. Diese Mittelsenkrechten bilden so ein Polygon, das den Mittelpunkt umgibt (vgl. [Ottm 02/1], S.501):

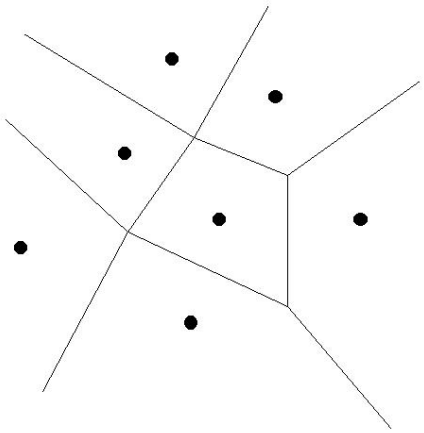


Abb. 6.7: Beispiel eines Voronoi-Diagramms

Definition: Für eine gegebene Menge *P* von Punkten und einen Punkt $p \in P$ ist der geometrischen Ort aller Punkte der Ebene, die näher bei *p* liegen als bei irgendeinem anderen Punkt, die *Voronoi-Region* von *p*. Sie ist stets der Durchschnitt aller Halbebenen von *p*. Die Vereinigung aller Voronoi-Regionen für eine Punktmenge wird *Voronoi-Diagramm* genannt.

Die Delaunay-Triangulierung der Punktmenge P ergibt sich sodann als duale Struktur des Voronoi-Diagramms und erzeugt ein globales Optimum, da alle $p \in P$ erfasst werden:

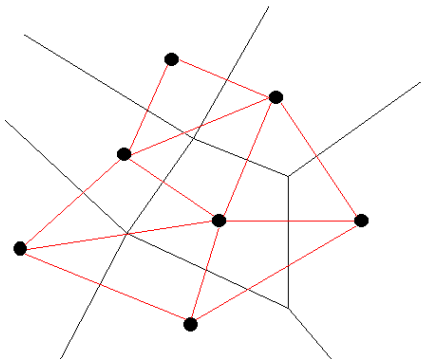


Abb. 6.8: Beispiel Voronoi-Diagramm mit Delaunay-Triangulierung

Nach Lawson (vgl. [Hoschek 92/1], S. 385) kann eine Delaunay-Triangulierung auch mit Hilfe eines *"Kreiskriteriums"* erzeugt werden. Das *lokale Umkreiskriterium* besagt, dass der Umkreis des Dreiecks T_{ikl} mit den Ecken P_i , P_k , P_l bei einer Triangulierung von vier Punkten nicht den gegenüberliegenden Punkt P_j des entlang der Seite P_k P_l anschließenden Dreiecks T_{kjl} beinhalten darf. Bei Nichterfüllung dieses lokalen Kriteriums ist die Teilungskante illegal, was zu einer Retriangulierung der vier Punkte durch Diagonalentausch (*edge flip)* führt.

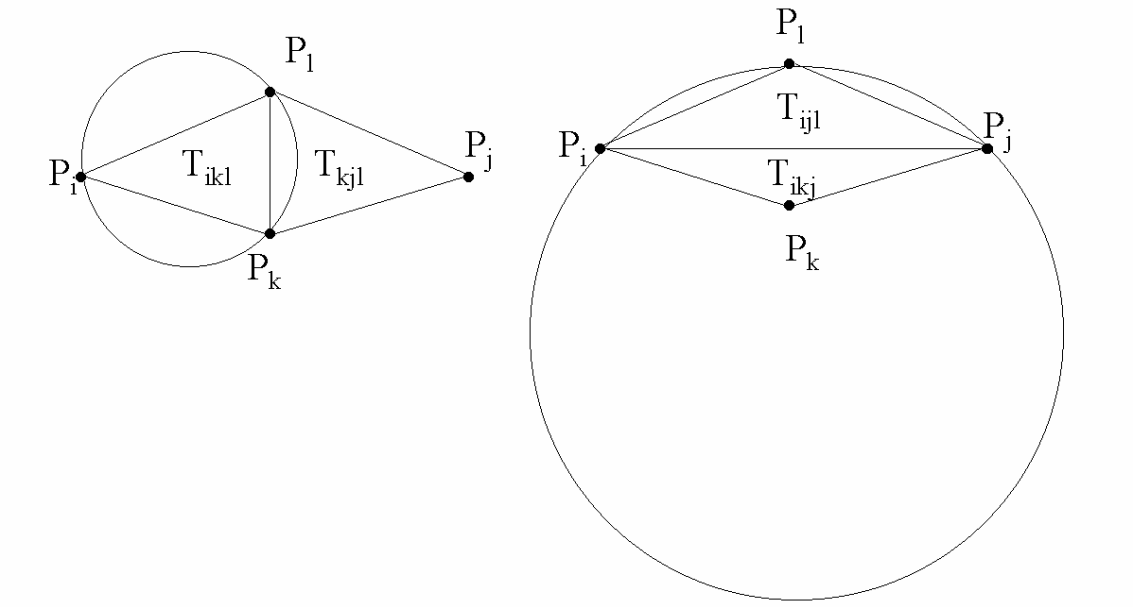


Abb. 6.9: Lokales Umkreiskriterium erfüllt (li), nicht erfüllt (re)

Das lokale Umkreiskriterium impliziert das globale Umkreiskriterium, nach dem für sämtliche Dreiecke der Triangulierung gelten muss, dass ihr jeweiliger Umkreis keine weiteren Datenpunkte beinhalten darf.

Die mit dem Umkreiskriterium erzeugte Triangulation ist dual zum Voronoi-Diagramm und damit äquivalent zur Delaunay-Triangulierung. Das lokale Umkreiskriterium erzeugt durch lokale Änderungen – wie auch das Max-Min-Winkelkriterium - ein globales Optimum.

Laut Hoschek (vgl. [Hoschek 92/1], S. 385) gibt es eine Vielzahl von Algorithmen, die auf der Delaunay-Technik beruhen, wichtig ist, darauf zu achten, dass ein guter Algorithmus immer eine global optimale Triangulierung erzeugt.

Nachfolgend werden der *Inkrementelle Einfügealgorithmus* und der Schritt-für-Schritt-Algorithmus, auch *Frontfortschrittverfahren* genannt, erklärt (vgl. [Hom 05/1]):

Inkrementeller Einfügealgorithmus

Grundidee dieses Algorithmus ist, dass ein freier Punkt in ein bereits bestehendes Delaunay-Netzwerk eingefügt werden soll. Das Netz wird dabei so umgeformt, dass wieder ein Delaunay-Netzwerk entsteht. Die Vorgehensweise wird durch Abb. 6.10 weiter erläutert:

1. Erzeugen des Anfangsnetzwerkes, dass dem Kreiskriterium entspricht (z.B. konvexe Hülle um alle Punkte).
2. Feststellen der Menge der Dreiecke aus der bereits bestehenden Triangulation, in deren Umkreis der nun in das Netzwerk einzufügende Punkt liegt.
3. Entfernen dieser Dreiecke aus der bestehenden Triangulation.
4. Verbinden der Randpunkte der Umgebung mit dem einzufügenden Punkt durch Kanten.
5. Überprüfen aller Vierecke, die diese Kanten als Diagonale haben. Bei Nicht-Erfüllung des Min-Max-Winkelkriterium wird die Diagonale vertauscht .

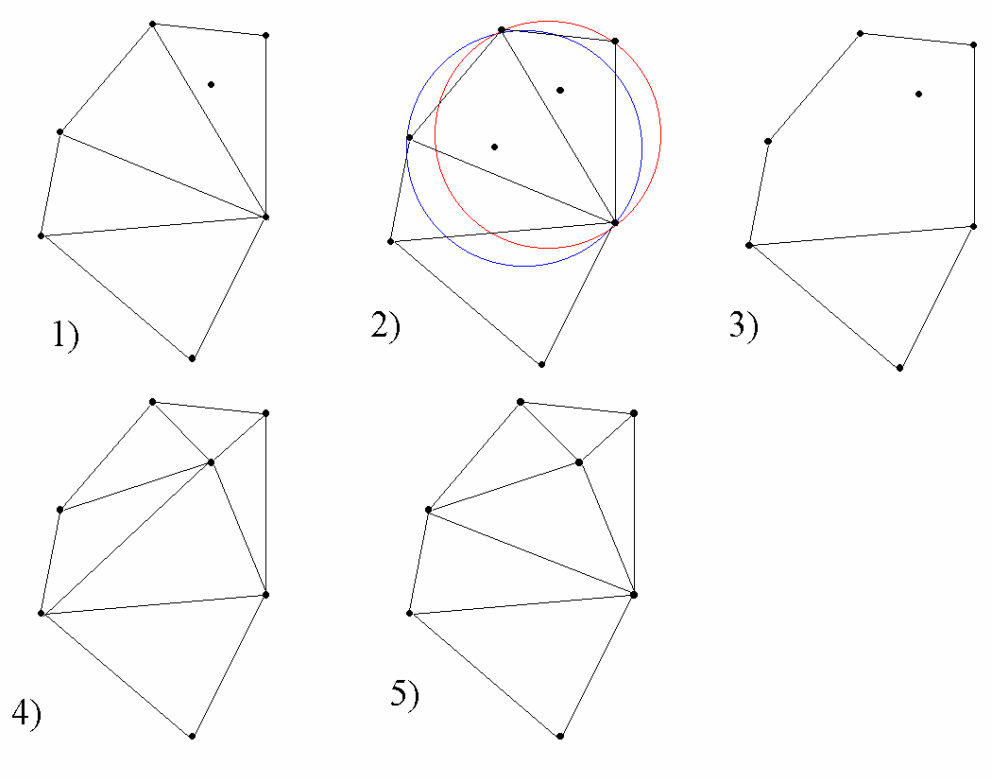


Abb. 6.10: Inkrementelles Einfügen eines Punktes in eine Delaunay-Triangulation

Frontfortschrittverfahren

Das Frontfortschrittverfahren ist auch als Schritt-für-Schritt-Algorithmus bekannt. Dieser Algorithmus (siehe Abb 6.11) nutzt das Umkreiskriterium direkt aus:

6. Erzeugen einer Basislinie zwischen zwei benachbarten Randpunkten einer Punktmenge.
7. Bilden eines Kreises jeweils durch beide Endpunkte der Kante und einen beliebigen weiteren Knoten der näheren Umgebung 1).
8. Bilden eines Delaunay-Dreiecks mit dem dritten Punkt des Umkreises, wenn dieser Umkreis keinen weiteren Punkt enthält (Delaunay-Umkreis).
9. Die beiden gewonnenen neuen Kanten bilden nun die neuen Basislinien 2).
10. Beginnend wieder bei 1). Ab diesem Zeitpunkt sind nur solche freien Knoten als Kandidaten zu betrachten, die auf der dem bereits existierenden Dreieck abgewandten "Seite" der soeben untersuchten Kante liegen.

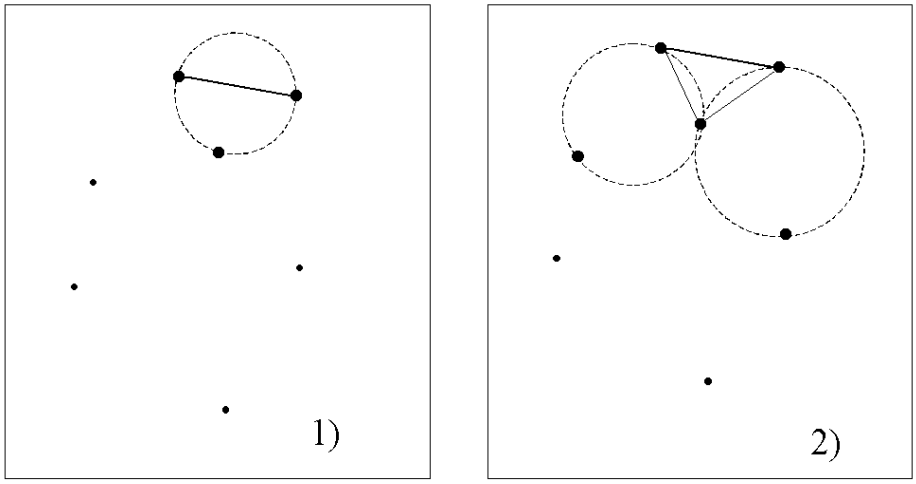


Abb. 6.11: Frontfortschrittverfahren

Wenn die Delaunay-Triangulation aus einer Punktmenge entsteht, ist ihre Berandung die konvexe Hülle der Punktmenge. Ist aber bei einer Triangulation neben der Knotenmenge eine feste Kantenmenge gegeben, die in der Triangulation enthalten sein soll, kann man nicht mehr von einer Delaunay-Triangulation ausgehen. Eine fest vorgegebene Kantenmenge kann z.B. eine feste Berandung sein, innere Zwangskanten (*breaklines*) oder innere Ringe. Zwangkanten kommen häufig vor. Deshalb wird dafür die beschränkte Delaunay-Triangulation definiert (vgl. [Hom 05/1]):

Definition: Eine beschränkte Delaunay-Triangulation liegt dann vor, wenn der Umkreis eines Dreiecks keine Knoten enthält, die von den Knoten des Dreiecks aus sichtbar sind. Zwei Knoten sind voneinander sichtbar, wenn sie geradlinig verbunden werden können, ohne eine Zwangskante zu schneiden.

Es gibt verschiedene Ansätze, solchen Problemen zu entgehen. Beispielsweise ist die gegebene Definition direkt auf den *Inkrementellen Einfügealgorithmus* anwendbar, in dem man beim Einfügen eines Punktes nur noch Dreiecke berücksichtigt, die von dem Punkt aus sichtbar sind. Der *Advancing-Front-Algorithmus*, der nachfolgend erklärt wird, läßt sich ebenfalls so entwickeln, dass Zwangskanten berücksichtigt werden.

6.2.4 Advancing-Front-Algorithmus

Nachfolgend wird der Advancing-Front-Algorithmus nach [Burg 05/1] erläutert. Er eignet sich hervorragend zur Vernetzung von komplexen zwei- und dreidimensionalen Oberflächen. Die Advancing-Front-Methode ist eine Netzgenerierungstechnik, mit der gleichzeitig Knoten und Elemente erzeugt werden können.

Der Rand (*boundary*) einer Geometrie wird in einem initialen Schritt in Verbindungsstrecken (Kanten) unterteilt. Die maximale Kantenlänge wird mit einer Parameterübergabe vom Benutzer definiert. Dadurch wird gleichfalls die Dichte der Gitterknoten entsprechend spezifiziert. Als *Advancing-Front* wird die dadurch entstandene Menge von Kanten bezeichnet, die sich nur in ihren Endpunkten berühren. Für den Rand einer Geometrie ergibt sich so ein Polygonzug. Sind alle Geometrien, die zu Zwangskanten führen (*boundary, breaklines*) diskretisiert, so bezeichnet man diese Menge *als aktive Front.* Die Elemente werden nun entlang der aktiven Front – ins Innere der Vernetzung weisend – eingefügt, wobei darauf zu achten ist, dass mindestens eine Kante des Elements auf der aktiven Front liegt. Nach der Erzeugung eines Elements wird die aktive Front aktualisiert und enthält nur noch die Kanten, die zur weiteren Vernetzung benötigt werden.

Wenn die aktive Front leer ist, ist der Vorgang der Vernetzung beendet.

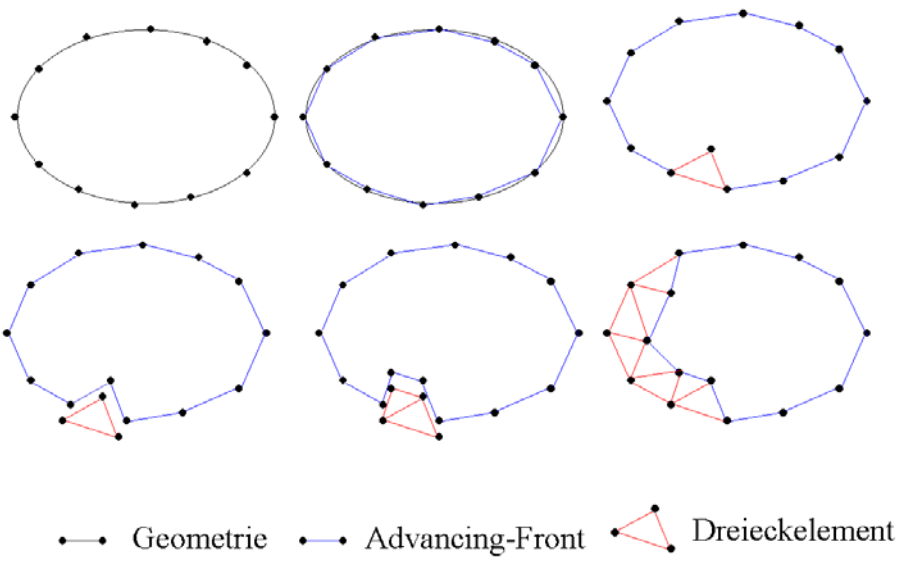


Abb. 6.12: Prinzip des Advancing-Front-Verfahrens

Zur Dreieckskonstruktion wird eine Kante der aktiven Front benötigt sowie ein Punkt des Inneren der Fläche. Dieser Punkt kann nach vereinbarten Regeln für die Elemente, z.B. der Gleichseitigkeit, neu erzeugt werden oder er ist ein bereits bestehender Punkt der aktiven Front. Für die Erzeugung der benötigten Punkte im Inneren besteht die Möglichkeit, diese sukzessiv nach vorgegebenen Qualitätsregeln einzufügen. Bern u.a. empfehlen die Dichte der Punkte im Inneren der Fläche durch ein Hintergrundnetz in Abhängigkeit zur Randdiskretisierung zu steuern (vgl. [Bern 05/1]):

Das Hintergrundnetz überdeckt das ganze Rechengebiet und wird zur Interpolation der Kantenlängen benutzt. Bern u.a. verwenden die Datenstruktur eines Quadtrees. Durch die Vorgabe der Randdiskretisierung werden die Quadranten in Randgebiete in entsprechender Tiefe erzeugt, die zur Mitte des Gebietes hin stufenweise abnimmt. Die nicht belegten Quadranten, die komplett innerhalb der Front liegen, werden mit einem Knoten in ihrem Zentrum belegt. Diese Punkte werden dann für die Dreiecksvernetzung benutzt.

Nach der Vernetzung kann noch eine Qualitätsverbesserung durch die sogenannte Netzglättung erzielt werden. Ist das Gebiet konvex und liegen keine inneren Zwangskanten bzw. –knoten vor, kann die Advancing-Front-Methode sofort geglättete Netze liefern. Da jedoch häufig der andere Fall vorliegt, sind Nachglättungsschritte durchzuführen.

Bei der Triangulierung ist folgendes zu beachten:

- Die Parameter für die Diskretisierung des Randes sollten geeignet gewählt werden, da durch die Parameterübergabe eine geeignete Dichtefunktion vorgegeben wird. Bei zu groß gewählten Parametern kann der Algorithmus scheitern – es wird kein Netz erzeugt.
- Bei der Erzeugung eines Elementes sind umliegende Frontteile oder Fronten auf Überschneidungen zu überprüfen. Überschneidungen sind zu vermeiden, da sie ein fehlerhaftes Netz erzeugen. Für diese Überprüfungen ist ein effizienter Algorithmus auszuwählen, da sonst die Geschwindigkeit der Netzgenerierung stark beeinträchtigt wird.

Teil III

Implementierung eines Netzgenerators

7. Implementierung eines Netzgenerators

In diesem Kapitel geht es um die Umsetzung der gewonnenen Erkenntnisse aus den vorangegangenen Kapiteln zur Realisierung eines Netzgenerators für zweidimensionale Oberflächen. Dieser Netzgenerator ist im STDSS zu integrieren und soll TINs erzeugen (vgl. [OpenGis 01/1]). Dazu werden zuerst Gebiete vorgestellt, die zu vernetzen sind. Dann wird die Funktionsweise des zu implementierenden Netzgenerators in einer Prozessübersicht dargestellt. Weiter werden die wichtigsten Methoden eingehender beschrieben. Dazu gehören unter anderem die Implementierung und Erstellung der Advancing-Front. Dies ist ein sehr wichtiger Prozess, denn ohne korrekt erstellte aktive Front wird auch kein Netz erzeugt. Weitere zu beschreibende Prozesse sind die Umsetzung des Delaunay-Kriteriums und die Netzglättung.

7.1 Gebietsdefinitionen

Zu vernetzen sind Gebiete in polygonaler Darstellung. Diese polygonalen Gebiete werden aus Shapefiles in die Anwendung DSSITT geladen und liegen in Form von Oberflächen-Objekten (Surface-Feature [OpenGis 01/1]) vor, deren Geometrie zumeist nicht konvex ist (siehe Abb. 7.1).

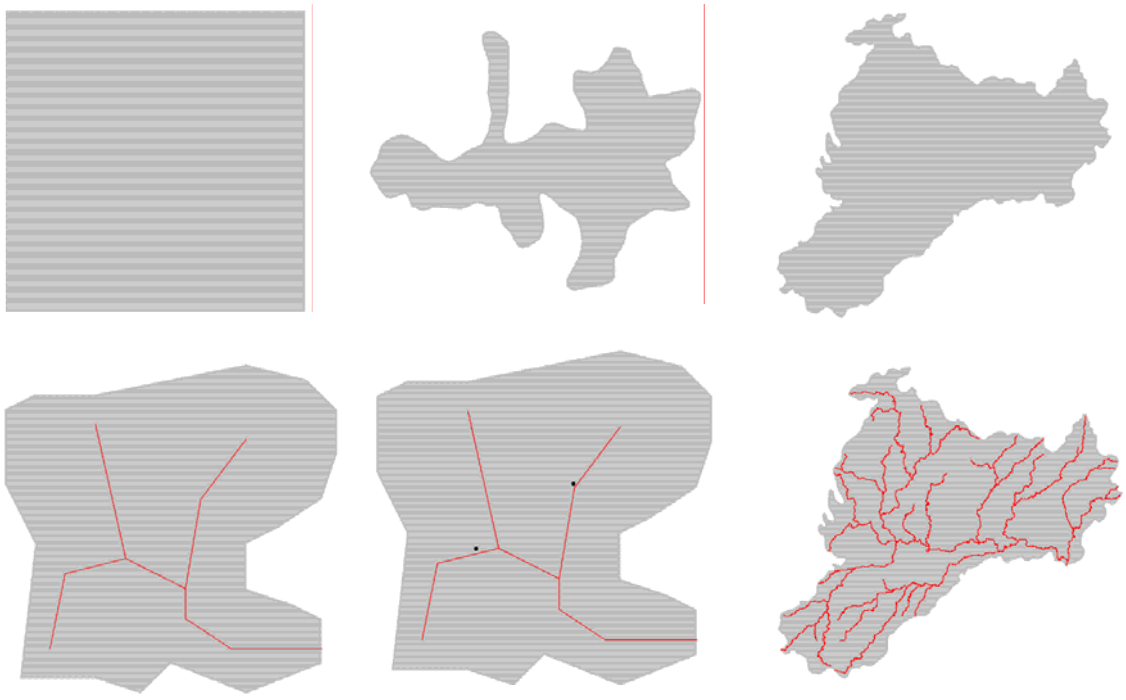


Abb. 7.1: Beispiele zu vernetzender Gebiete

Gleichfalls besteht die Möglichkeit, Gebiete mit inneren Kanten (*breaklines*) oder Zwangspunkten (*posts*) zu vernetzen. Wie in Abb. 7.1 rot angedeutet, können dies Einzugsgebiete von Flüssen oder Abbildungen von Brunnen oder Messstationen sein.

7.2 Datenstrukturen Advancing-Front

Die in [OpenGis 01/1] spezifizierten Datenstrukturen für *TIN* und *Triangle* sind Grundlage für die gebildeten Datenstrukturen.

Der Konstruktor der Klasse GeoAdvancingFront2D:

```
GeoAdvancingFront2D::GeoAdvancingFront2D (
surface_ area:<GeoSurface2D>,
post : ArrayList<GeoPoint2D>,
breakLines : ArrayList<GeoLine2D>,
maxLength : double,
minLength : double)
```

GeoSurface2D, GeoPoint2D, GeoLine2D sind abgeleitete Klassen der Java-Bibliothek *java.awt.geom*. Somit können die Funktionen der Basisklassen aus dieser Bibliothek benutzt werden. *Surface* ermöglicht den Zugriff auf den Rand (boundary) der Oberfläche. *BreakLines* enthält alle Zwangskanten, die nicht verändert (verschoben) werden dürfen. *StopLines* sind Kanten, die weggelassen werden dürfen – diese bleiben zur Zeit unberücksichtigt. *Posts* enthalten die Zwangspunkte. *MaxLength* ist die maximale vom Benutzer wählbare Verbindungsstrecke zur Randdiskretisierung. *MinLength* ist die Mindestlänge der Verbindungsstrecke der Randdiskretisierung. Auch sie ist ein frei wählbarer Parameter. Eine weitere Eigenschaft der Klasse Advancing-Front ist die Liste in der die Verbindungsstrecken als aktive Front gehalten werden:

```
private ArrayList<GeoLine2D> front;
```

Dreiecke werden, wie schon in 4.2.5 erklärt, durch ihre drei Ecken und die zugehörigen drei Kanten definiert:

```
GeoTriangle2D:: GeoTriangle2D (new GeoPoint2D[] {p0,p1,p2})
```

Die drei Seiten des Dreiecks werden durch bitweise Verschiebung (a<<b), als Byte-Array initialisiert:

```
private static byte SIDE[] = {
  (1<<0) | (1<<1),              // v[0] && v[1]
  (1<<1) | (1<<2),              // v[1] && v[2]
  (1<<2) | (1<<0)               // v[2] && v[0]
};
```

Der **Left-Shift**-Operator << schiebt alle Bits des linken Operanden um die Anzahl von Bits nach links, die der rechte Operand angibt, wobei die (rechten) niederwertigen Bits mit Nullen aufgefüllt werden (vgl. [Ullenboom 02/1]).

Beispiel für eine bitweise Verschiebung :
(a | b) = Or

1 << 0	00000001
1 << 1	00000010
1 << 0 | 1 << 1	00000011

1 << 1	00000010
1 << 2	00000100
1 << 1 | 1 << 1	00000110

1 << 2	00000100
1 << 0	00000001
1 << 2 | 1 << 0	00000101

Durch diese Art der Speicherung wäre es auch möglich, ein Flächengebilde mit *n* Ecken zu speichern.

7.3 Advancing-Front-Methode des STDSS

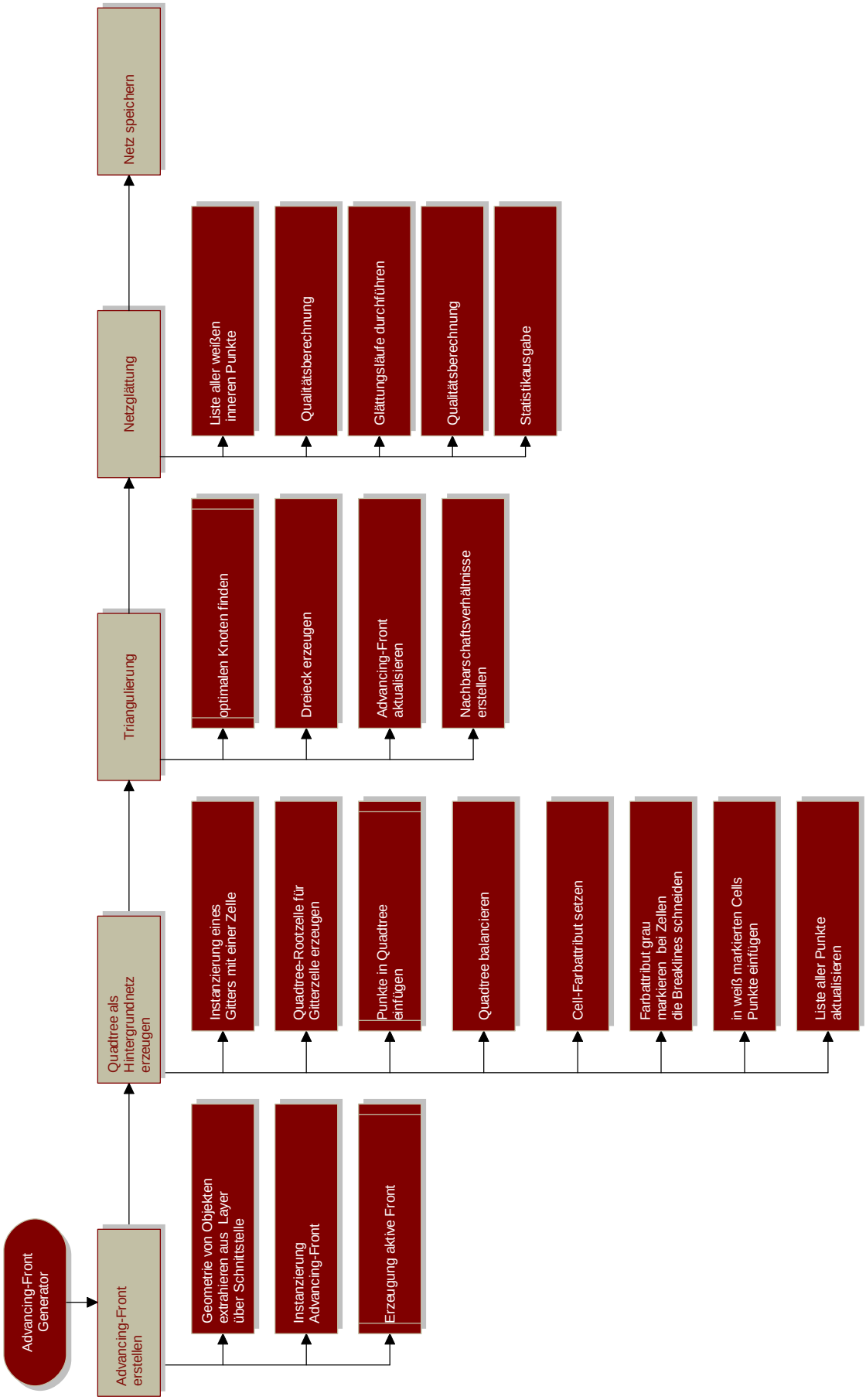


Abb. 7.2: Prozessübersicht des Netzgenerators

7.3.1 Erstellen Advancing-Front

Eine Liste von sinnvoll diskretisierten Rand- und Zwangskanten muss zur Triangulierung vorliegen. Diese Liste nennt man *erste aktive Front*.

Schnittstellen zur Geometrie der Layer

Die zu vernetzenden Oberflächen liegen in Layer-Format als Shapefile vor und sind in der Anwendung geladen.

Um eine brauchbare aktive Front zu erstellen, benötigt man die Geometrie der Feature eines Layers. Zur Übernahme der entsprechenden Geometrien benötigt man jeweils Schnittstellen für

- Oberflächen mit der Geometrie des Randes (boundary) von Surface-Feature,
- Flussläufe mit der Geometrie von Curve-Feature (segments),
- Brunnen bzw. Messstationen mit der Geometrie von Point-Feature (points).

Erstellen der aktiven Front

Nach ihrer Instanzierung wird die zu vernetzende aktive Front erstellt. Bei der Erzeugung der aktiven Front geht man folgendermaßen vor:

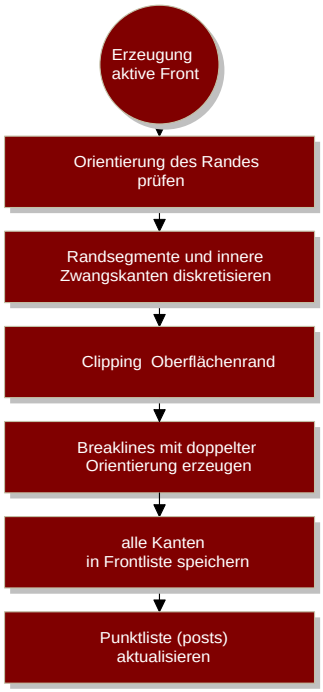


Abb. 7.3: Prozessübersicht zur Erzeugung der *aktiven Front*

Zuerst wird ein allgemeiner Überblick über die in der Prozessübersicht gezeigten Funktionen gegeben. Danach werden die Prozesse "*Orientierung des Randes*" und "*Clipping Oberflächenrand*" ausführlicher erklärt.

Erster Schritt bei der *Erzeugung der aktiven Front* ist die Überprüfung der Randorientierung des Flächenpolygons. Die Oberfläche wird nach innen vernetzt, darum muss die *Orientierung der Randkanten überprüft* werden. Es wird rechts von der Kante vernetzt, darum müssen die Kanten des Polygonrandes im Uhrzeigersinn sortiert vorliegen.

Der *Diskretisierungsvorgang* besteht aus Zusammenfügen (*merge*) und Teilen (*splitten*) von Kanten. Kanten, die kleiner sind als die vorgegebene minimale Länge, werden zusammengefügt. Kanten, welche die maximale Länge überschreiten, werden geteilt. Problematisch wird es, wenn sich zwei innere Zwangskanten (*breaklines*) schneiden.

Normalerweise schneiden sich nur zwei Kanten in ihren Endpunkten. Wenn es jedoch mehr als zwei Kanten sind, z.B. Fluss mit Seitenarmen, muss dies bereits vor dem Beginn des Vernetzungsvorganges berücksichtigt werden. Die Daten sollten in konstistenter Weise ohne Redundanzen vorliegen. Dies sollte möglichst schon bei der Digitalisierung berücksichtigt werden. Beispielsweise kann man festlegen, ob bei der Erzeugung eines Segmentes einer Curve der zuletzt erzeugte Punkt auch gleichzeigtig für das neue Segment benutzt wird. Bei manueller Erzeugung eines Startpunktes für ein neues Segment auf dem Display ist es schwierig, genau den vorher erzeugten Endpunkt des Vorgängersegmentes zu treffen.

Bei nicht korrekter Digitalisierung müssen die Daten einer sogenannten Vorverarbeitung (Preprocessing) unterzogen werden. Dieses sind meist sehr aufwendige und arbeitsintensive, teils manuelle Vorgänge, die besondere Programmunterstützung durch verschiedene Werkzeuge (Tools) benötigen.

Ist die Digitalisierung nicht korrekt und besteht keine Möglichkeit eines Preprocessing muss für eine korrekte Diskretisierung ein Vergleich aller Kanten untereinander durchgeführt werden, da in Shapefiles keine Topologie gespeichert ist. Ein dadurch erheblich erhöhter Geschwindigkeitsverlust bei der Vernetzung wäre die Folge und die Anwendung würde stark an Effizienz verlieren.

Im STDSS sind noch keine entsprechenden Werkzeuge eines Preprocessing installiert. Aus diesem Grund wurde beim Laden der Geometrie für Curven eine automatiserte Vorverarbeitung eingebaut. Diese Vorverarbeitung beinnhaltet folgende Schritte:

- Erstellen einer Topologie zwischen Kanten und Punkten, dadurch werden Punkte mit drei angrenzenden Kanten als Zwangspunkte deklariert.
- Splitten und Zusammenfügen der Kanten.
- Prüfen von Überschneidungen des Inneren von Kanten, bei Verzweigungen und nötigenfalls Erstellung eines neuen Zwangspunktes.

Mit diesen Schritten sind die Zwankskanten schon bereits diskretisiert bei der Instanzierung der aktiven Front und stehen weiteren Verarbeitungen zur Verfügung.

Das *Clipping des Oberflächenrandes* bedeutet, dass sich keine Geometrie außerhalb der zu vernetzenden Fläche befinden darf.

Breaklines müssen in zweierlei *Orientierung* (rechts und links) vorhanden sein, weil sie beidseitig vernetzt werden. Existieren die Breaklines mit einer Orientierung, z.B. nach rechts, ist es direkt bei der Vernetzung einer Kante möglich, eine gleiche Kante mit einer anderen Orientierung, nach links, zu erzeugen, indem man diese einfach umdreht, d.h., der Startpunkt wird zum Endpunkt und der Enpunkt zum Startpunkt. Die Kante mit der anderen Orientierung wird in die Frontliste gespeichert. Entweder müssen dafür die inneren Zwangskanten auch als solche gekennzeichnet werden, damit sie als solche von den Randkanten unterschieden werden können, oder sie werden in einer separaten Front-Liste geführt.

Für diesen Netzgenerator wird eine einfachere Methode verwendet. Nach der Diskretisierung wird ein zweiter Satz Breaklines mit entgegengesetzer Orientierung in der Front-Liste gespeichert. Man erspart sich dadurch das ständige Vergleichen der Kanten und das Verwalten von separaten Listen.

Nach diesen Schritten werden *alle Kanten* – die diskretisierten Randkanten und die rechts- sowie linksorientierten Breaklines – in einer einzigen Frontliste *gespeichert.*

Der letzte Schritt ist die *Aktualisierung der Liste der Punkte* (*posts*). Zu den evtl. vorgegebenen Zwangspunkten werden alle vorkommenden Punkte der Geometrien in die Liste eingefügt.

Orientierung des Randes

Für eine Vernetzung ins Innere der Fläche muss die Orientierung der Randkanten vereinbart werden. Für den Netzgenerator des STDSS wurde vereinbart, dass rechts von der zu vernetzenden Kante trianguliert wird, darum müssen die Kanten im Uhrzeigersinn (clockwise) geordnet sein. Dies ist vor der Diskretisierung zu testen.

Alle vorkommenden Punkte aus dem Polygon werden miteinander verglichen. Dabei bedient man sich einer Methode aus der algorithmischen Geometrie (siehe [Schwen 04/1]), erklärt an einem Dreieck:

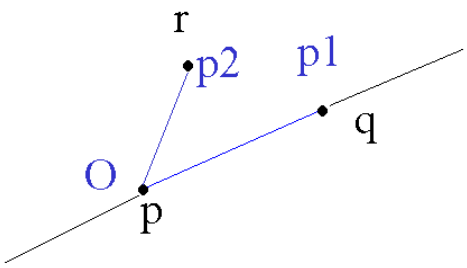


Abb. 7.4: Beispiel für Orientierung eines Dreiecks

Zu prüfen ist, ob *r* links oder rechts von der von *p* nach *q* führenden Geraden liegt. Dazu werden die beiden Vektoren *p1*, *p2* ermittelt. Durch das Vektorprodukt von *p1 x p2* wird ein repräsentativer Wert ermittelt, der die Orientierung angibt:

$$p1 := q - p \tag{7.1}$$

$$p2 := r - p \tag{7.2}$$

$$\text{Berechne Vektorprodukt: } p1 \times p2 = \begin{pmatrix} p1_x & p2_x \\ p1_y & p2_y \end{pmatrix} \tag{7.3}$$

p1 x *p2* > 0 → *p2* links von $\overrightarrow{Op1}$

p1 x *p2* < 0 → *p2* rechts von $\overrightarrow{Op1}$

p1 x *p2* = 0 → *O, p1, p2* kollinear

→ *r* links von $\overrightarrow{pq}$, falls (q - p) x (r - p) > 0

Zum Vergleich aller Punkte eines Polygons werden alle Startpunkte der Verbindungsstrecken in einer Liste der Reihe nach gespeichert. Der Vektor *p1* wird zwischen den ersten beiden Punkten der Liste ermittelt (siehe Abb. 7.5) und als Basis für die Vergleiche verwendet. So werden die Vektoren *p2* bis *p7* nach (7.2) und (7.3) berechnet und mit Vektor *p1* nach oben beschriebenem Algorithmus verglichen. Die ermittelten repräsentativen Werte für die Orientierung werden summiert.

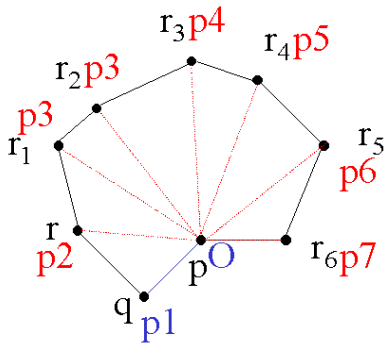


Abb. 7.5: Beispiel für die Orientierung der Verbindungsstrecken eines Polygons

Liegen die Punkte im Uhrzeigersinn (clockwise) vor, ist der Gesamtwert größer Null. Ist der Gesamtwert kleiner Null sind die Verbindungsstrecken des Randes entgegen dem Uhrzeigersinn orientert (counterclockwise) und müssen für die Vernetzung umgedreht werden.

Clipping Oberflächenrand

Das Oberflächen-Clipping ist zu vergleichen mit einer Bereichssuche. Anstelle eines Suchrechtecks grenzt der Rand der zu vernetzenden Fläche den Suchbereich ab. Liegen Geometrien komplett außerhalb des Suchrechtecks, bleiben sie beim Erstellen der aktiven Front unberücksichtigt. Geometrien, die den Rand schneiden, müssen bearbeitet werden.

Alle Punkte der bereits diskretisierten Randkanten werden berücksichtigt bei der Bildung eines Polygons mit *n* Kanten für den Suchbereich.

Danach werden die übrigen Geometrien, wie Zwangskanten und –punkte mit dem Suchbereich verglichen. Alle Zwangskanten werden mit Hilfe ihrer Bounding Box zunächst auf Überschneidung mit dem Suchbereich überprüft. Dabei gibt es drei Möglichkeiten:

- die Bounding Box der Zwangskante liegt komplett außerhalb des Suchbereichs, Kante wird aus Liste gelöscht.
- die Bounding Box der Zwangskante liegt komplett innerhalb des Suchbereichs, Kante wird nicht weiter behandelt.
- die Bounding Box der Zwangskante schneidet den Rand des Suchbereichs, Kante wird untersucht.

Ähnlich werden auch die Zwangspunkte bearbeitet.

Die Kanten aus der separaten Liste werden daraufhin spezieller auf Möglichkeiten von Überschneidungen untersucht:

(a) innere Zwangskante schneidet mit ihrem Rand den Rand einer Kante des Randes;
(b) innere Zwangskante schneidet mit ihrem Inneren den Rand einer Kante des Randes;
(c) innere Zwangskante schneidet mit ihrem Rand das Innere einer Kante des Randes;
(d) innere Zwangskante und Randkante schneiden sich mit ihrem Inneren.

Der Fall a) wird nicht weiter berücksichtigt, da er keine Auswirkung auf die Vernetzung hat.

Bei b) wird die Zwangskante bearbeitet, d. h. der außen liegende Bereich wird abgetrennt und nicht mehr weiter berücksichtigt. Der Rand der Kante des Suchbereiches, der geschnitten wird, wird für die Zwangskante zum Rand der Zwangskante.

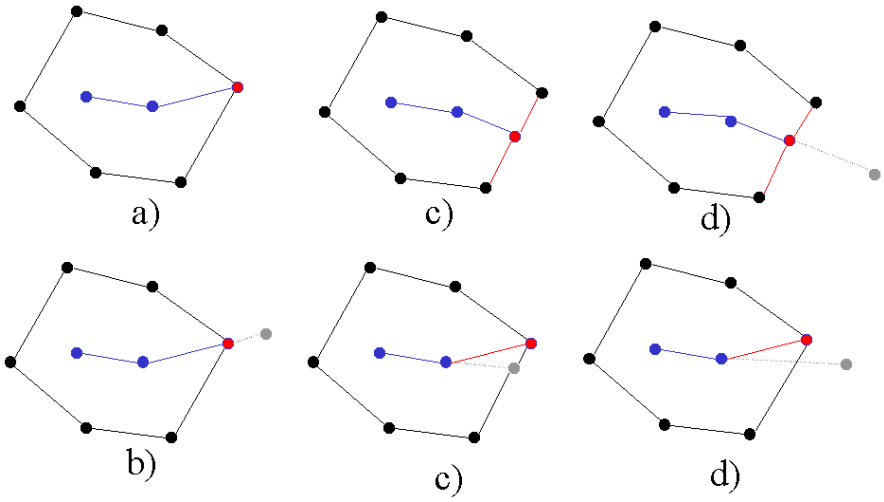


Abb. 7.6: Behandlung möglicher Randschneidungen beim Oberflächen-Clipping

Tritt c) auf, wird zuerst untersucht, ob der Rand (Start- oder Endpunkt) der Zwangskante in der Nähe eines der Randpunkte der geschnittenen Kante des Suchbereiches liegt. Ist der Abstand kleiner von einem der Randpunkte als *die minimale zu diskretisierende Länge der Randkanten* (minLength), wird der schneidende Rand der Zwangskante ersetzt durch den nahen Randpunkt des Suchbereichs.

Ist der Abstand des Randes der Zwangskante zu den Randpunkten der geschnittenen Kante des Suchbereiches größer als minLength wird die schneidende Kante des Suchbereichs im schneidenden Randpunkt der Zwangskante geteilt.

Für d) wird zunächst der Punkt ermittelt, in dem sich die Kanten schneiden. Ist dieser Punkt in der Nähe eines der Randpunkte der geschnittenen Randkante, verfährt man mit dieser nach c) und mit der Zwangskante nach b). Andernfalls wird wie in c) die Randkante im Schnittpunkt in zwei Kanten zerlegt. Der Schnittpunkt wird zum Rand der Zwangskante. Der außerhalb liegende Teil wird wie in b) gelöscht.

Die Listen der Zwangskanten und Randkanten werden aktualisiert im Hinblick auf die Erstellung der aktuellen Front.

Wurden die Prozesse zur Erstellung der aktiven Front erfolgreich durchgeführt, erhält man eine Liste von Kanten und eine Liste von Punkten. Beide Listen repräsentieren zu diesem Zeitpunkt den Rand und die inneren Zwangskanten nach der Diskretisierung. Visualisierung der Punktliste einer erfolgreich erstellten *ersten aktiven Front*:

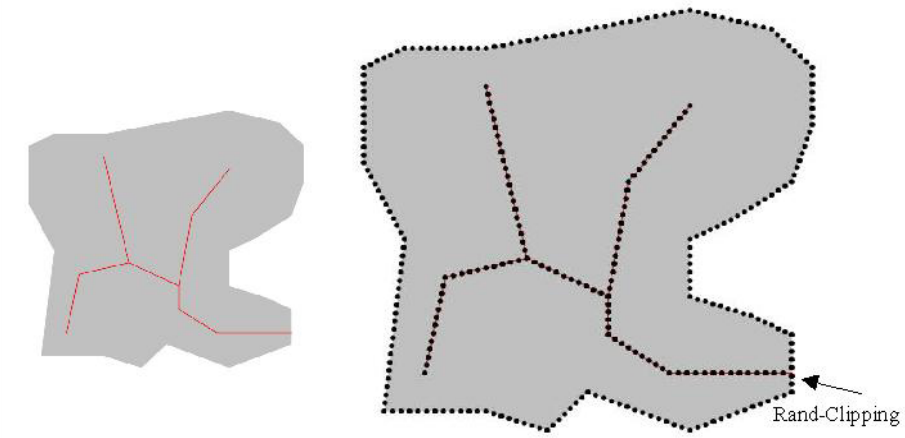


Abb. 7.7: Beispiel für eine erfolgreich mit Oberflächen-Clipping erstellte aktive Front

7.3.2 Quadtree als Hintergrundnetz

Für die Steuerung und Erzeugung weiterer Punkte im Inneren der zu vernetzenden Fläche wird ein schon in der Anwendung implementiertes Quadtree benutzt. Hier wird nur kurz auf die Funktionsweise dieses Quadtrees eingegangen, weil dies auch für die weitere Vernetzung wichtig ist.

In der Anwendung STDSS werden für die Indizierung des Raumes zwei Verfahren der in Kapitel 5 erklärten Verfahren kombiniert. Als Grundlage dient ein Gitter (Grid), auf welchem das Quadtree aufbaut. Nachfolgend die Klassenhierarchie der Implementierung in UML:

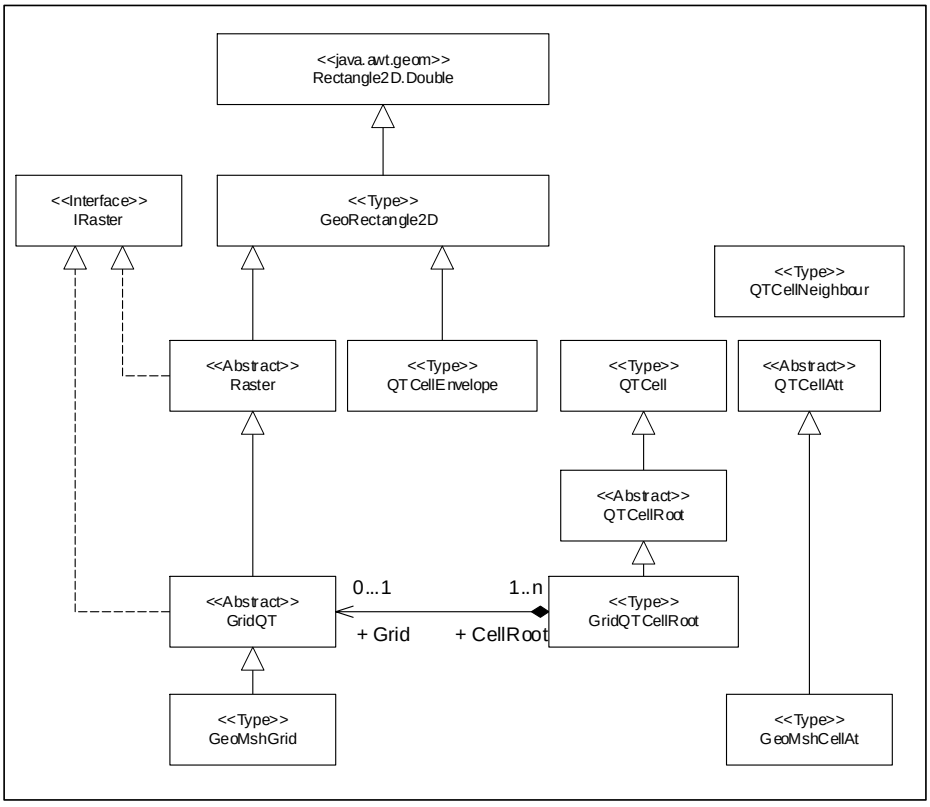


Abb. 7.8 : UML-Klassenhierarchie der Quadtree-Implementation in Java

Um den aktuellen Raum zu indizieren, wird zuerst eine Instanz der Klasse *GeoMshGrid* als Basis initialisiert. *GeoMshGrid* erbt von der Klasse *GridQT*, welche wiederum von der Klasse *Raster* erbt. *GeoMshGrid* kann damit ebenfalls die Methoden der beiden Klassen nutzen, sowie die Methoden des Interface *IRaster*, welches in den Klassen *GridQT* und *Raster* implementiert ist.

Zur Erzeugung der Instanz und Bestimmung der Gittergröße wird ein die Fläche umschließendes Rechteck (*envelope*) und die maximale Zellengröße an den Konstruktor übergeben.

Jede Gitterzelle stellt eine Instanz der Klasse *GridQTCellRoot* dar. Aus diesem Grund kann man das Gitter auch als "Grid von Objekten" bezeichnen. *GridQTCellRoot* besitzen als Eigenschaften einen Zeilen- und Spaltenindex, so dass man, wie in 5.1 erklärt, über diesen sofort auf die Zelle zugreifen kann. Ist die Datenverteilung – eine Punktemenge – inhomogen, ist es möglich, für jede Instanz von *GridQTCellRoot* ein Quadtree zu erzeugen. Das Quadrat der Wurzel (root) des Quadtrees entspricht einer *GridQTCellRoot*.

Wenn die Quadtree Root-Cell erzeugt ist, werden nun sukzessive die Punkte der ersten erfolgreich erzeugten aktiven Front in das Quadtree eingefügt (siehe Kapitel 5). Es wird solange eine rekursive Quadtree-Teilung vorgenommen, bis jedes Quadrat lediglich einen Punkt der Punktemenge enthält. Diese Knoten (*cells*), die einen Punkt enthalten, sind Blattknoten (*leaves*). Der Punkt wird als Attribut des entsprechenden *leaf* gespeichert. Zusätzlich werden die *leaves* durch ein zweites Attribut klassifiziert.

Jeder *leaf* erhält ein zusätzliches Farbattribut mit folgenden Eigenschaften:

- NONE = innerer *Cells*;
- BLACK = *leaves*, deren Quadrate außerhalb der konvexen Hülle, der gegebenen Punktmenge liegen ;
- GREY = *leaves*, deren Quadrate einen Punkt der vorgegebenen Punktmenge enthalten oder deren Rand schneiden, ohne einen Punkt zu enthalten und *leaves*, deren Quadrate eine innere Linie, z.B. einen Fluss, schneiden, aber keinen Punkt enthalten;
- WHITE = *leaves*, deren Quadrate ganz im Inneren der konvexen Hülle der gegebenen Punktmenge liegen und keinen dieser Punkte enthalten und auch keine Linie schneiden.

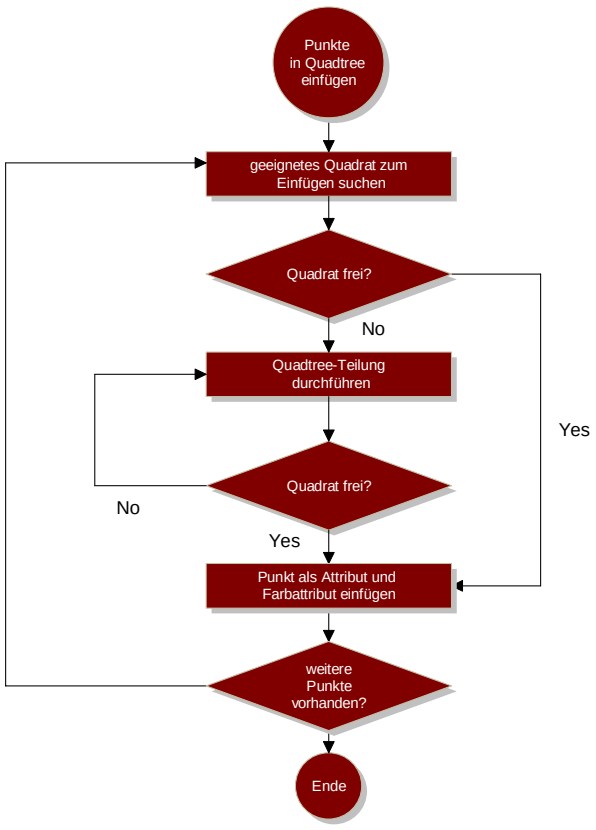


Abb. 7.9: Flow-Chart - Knoten einfügen in ein Quadtree

Für das Einfügen eines Punktes wird für diesen das entsprechende kleinste Quadrat gesucht, in dem er enthalten ist. Für den ersten Punkt ist dies beispielsweise das Quadrat der Wurzel (root). Wird sofort ein passender Knoten gefunden, dessen Punktattribut gleich *null* ist, wird der Punkt als Attribut des Knotens gespeichert. Gleichfalls wird das Farbattribut auf GREY gesetzt.

Wenn jedoch der Knoten bereits ein Blattknoten ist, muss eine Teilung vorgenommen werden. Das Quadrat dieses Blattknotens wird zerlegt. Es werden vier *leaves* mit ihren Quadraten erzeugt und dem aktuellen Knoten als Quadranten zugewiesen. Gleichfalls wird jedem neuen *leaf* der aktuelle Knoten als Elternknoten (parent) zugewiesen. Somit ist der aktuelle Knoten kein *leaf* mehr und ist zu einem inneren Knoten geworden. Das bedeutet auch eine Attributänderung für den aktuellen Knoten und seine Kinder.

Der Attributwert des aktuellen Knotens, der jetzt ein innerer Knoten ist, wird auf *null* gesetzt. Daraufhin wird ermittelt, an welchen der vier Quadranten das Attribut übertragen wird. Das Prinzip wurde bereits in Kapitel 5 erläutert. Liegt beispielsweise der Punkt im Süd-West-Quadranten wird das aktuelle Punktattribut an den Süd-West-Knoten übergeben. Für die drei anderen Knoten werden als Attributwerte Initialwerte erzeugt.

Dieser Vorgang wird solange wiederholt, bis alle Punkte in das Quadtree eingefügt sind. Danach wird der Quadtree-Baum noch ausgeglichen, damit sich angrenzende Quadrate höchstens um eine Stufe unterscheiden. Informationen über die Funktionsweise findet man in Kapitel 5 sowie in einschlägiger Fachliteratur, wie [Bil 91/1], [Samet 95/1].

Ist der Quadtree-Baum ausgeglichen, ist das Quadtree erzeugt. Ab diesem Punkt kann man direkt mit den Blattknoten arbeiten. Alle *leaves* erhalten ein Farbattribut (BLACK, GREY, WHITE) für ihre Zugehörigkeit. Knoten mit dem Farbattribut "WHITE" erhalten zusätzlich einen Punkt als Attribut im Zentrum ihres Quadrates. Dadurch entsteht eine zusätzliche Menge an Punkten im Inneren der zu vernetzenden Fläche.

Die Liste der Punkte, die bisher nur die Punkte der aktiven Front enthielt, wird um die zusätzlichen Punkte erweitert und bildet nun eine Liste aller zu vernetzender Punkte.

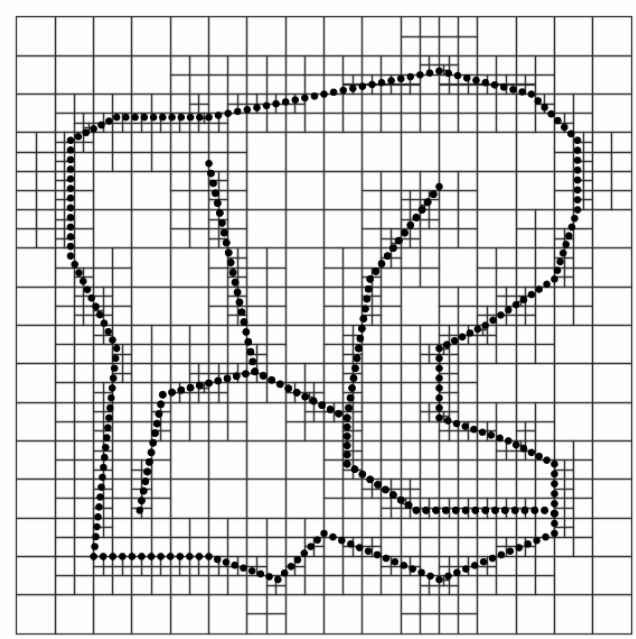


Abb. 7.10: Erzeugtes Quadtree mit Zwangspunkten der aktiven Front

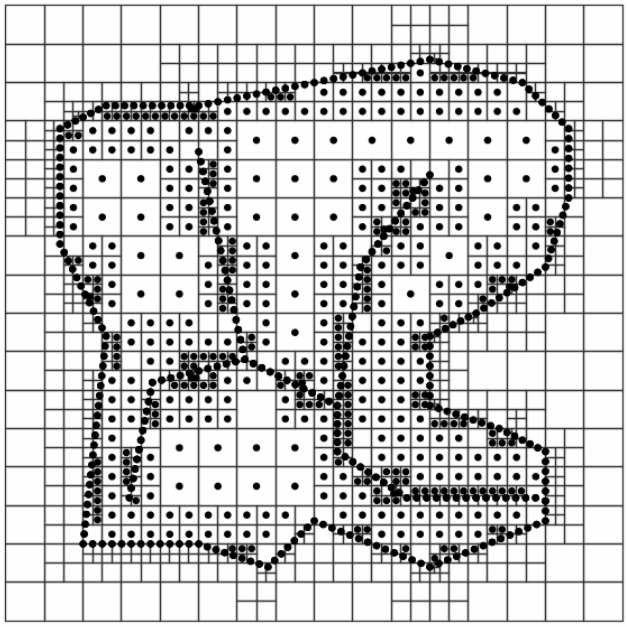


Abb. 7.11: Erzeugtes Quadtree mit zwangs- und flächenverteilten Punkten

Abbildung 7.10 und 7.11 zeigen erzeugte Quadtrees für die aktuelle Front. Dort, wo Zwangspunkte vorgegeben sind, erkennt man eine höhere Dichte von zu vernetzenden Knoten. Dies läßt auf ein höheres Aufkommen von Daten bzw. Informationen für diese Bereiche schließen.

7.3.3 Triangulierung

Nach der Erzeugung des Hintergrundnetzes kann mit der Triangulierung begonnen werden. Die Liste der aktiven Front wird als Stapel (Stack) benutzt. Die zu triangulierenden Kanten werden vom Listenende entnommen. Gleichzeitig werden durch die Dreiecksbildung neue Kanten erzeugt, die wiederum bei jeder Aktualisierung der aktiven Front ans Ende der Liste angefügt werden. Beim Erzeugen neuer Kanten wird die Orientierung nach rechts sofort berücksichtigt. Die Reihenfolge der zu vernetzenden Kanten spielt keine Rolle.

Die Regeln zur Dreiecksbildung sind bekannt. Der Teilprozess "optimalen Punkt finden" aus dem Prozessablaufplan in Abb. 7.2 besteht aus mehreren Methoden und wird nachfolgend näher erklärt:

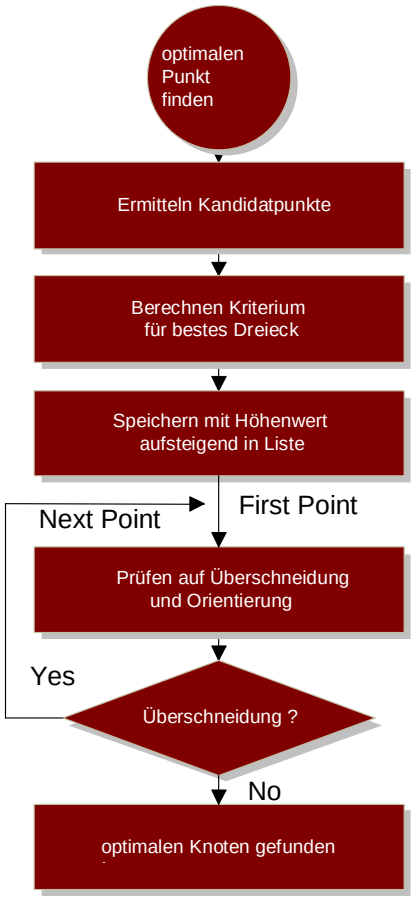


Abb. 7.12: Finden des optimalen Kanditatpunktes für die Triangulierung

Um ein Dreieck nach einem Delaunay-Kriterium zu erzeugen, werden dafür potentielle Punkte im Inneren der zu vernetzenden Fläche ermittelt. Abb. 7.12. zeigt den dazugehörigen Prozessablauf.

Mit Hilfe einer Bereichssuche unter Ausnutzung des Hintergrundnetzes werden Kanditatpunkte für eine Triangulierung der aktiven Kante ermittelt:

Ermitteln Kandidatpunkte

Um nicht alle inneren Punkte für ein Delaunay-Kriterium prüfen zu müssen, gibt es eine Methode zur Ermittlung potenieller Kanditatpunkte für eine Triangulierung. Für diese Bereichssuche wird zuerst ein Suchrechteck über der zu triangulierenden Kante erzeugt.

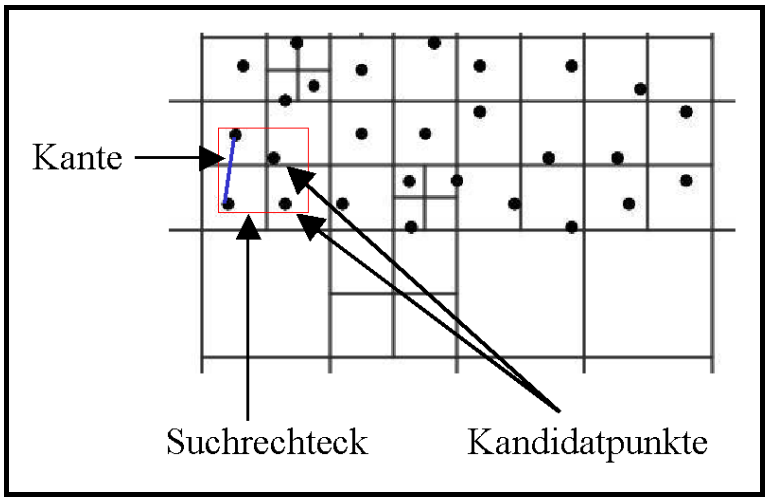


Abb. 7.13: Bereichssuche über Ausgangskante für Kandidatpunkte der Dreiecksbildung

Über die Quadranten des Quadtrees, beginnend mit der Rootzelle, wird der Quadtree-Baum sukzessive top-down rekursiv durchsucht (*traversiert*) und die Punktattribute der zu den sich mit dem Suchrechteck schneidenden Quadranten gehörenden *leaves* werden in eine Listes gespeichert.

Wenn das Ergebnis dieser Traversierung eine leere Liste ist, wird das Suchrechteck vergrößert. Dieser Vorgang wird so lange wiederholt, bis mindestens ein Kandidatpunkt gefunden wurde, der rechts von der zu triangulierenden Kante liegt.

Diese Vorauswahl der Kandidatpunkte verkleinert bereits die Anzahl der infrage kommenden Punkte für die weitere Auswahl des optimalen Kandidaten für die Dreiecksbildung.

Kriterium für das beste Dreieck berechnen

In Kapitel 6 wurden die Qualitätskriterien für die Dreiecksbildung erklärt. Die Berechnung erfolgt nach dem Umkreiskriterium von Delaunay. Für alle gefundenen Kandidatpunkte wird ein Umkreistest durchgeführt. Der Kandidatpunkt, der mit den Endpunkten der Kante den kleinsten Umkreis erzeugt, ohne dass ein weiterer Punkt im Umkreis liegt, ist der geeignete Punkt für eine Dreiecksbildung.

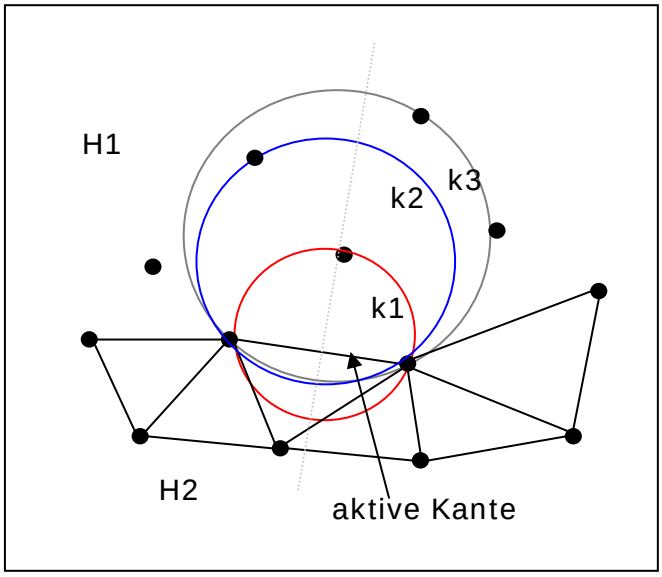


Abb. 7.14: Umkreistest für Kandidatpunkte

Da die "linke" Seite (H2 = Halbebene 2) der aktiven Front bereits trianguliert ist, wie Abbildung 7.14 zeigt, wird sich nur nach der von der Kante rechten Seite (H1) orientiert. Eine programmtechnisch leichter zu realisierende Variante als das Finden des kleinsten Umkreises, in welchem kein weiter Punkt liegt, ist die Suche nach der kleinsten Höhe des Umkreises über der Ausgangskante.

Wie in der Abbildung oben dargestellt, ist k1 nicht nur der kleinste Umkreis, sondern hat auch die kleinste Höhe des Umkreises über der aktiven Kante. Dies ist auch ein Indikator dafür, dass auf der zu triangulierenden Seite H1 kein weiterer Punkt enthalten ist. Wird auf diese Weise uneingeschränkt vernetzt, wird das Delaunay-Kriterium für jedes Dreieck erfüllt und somit für das gesamte Netz.

Problemfälle bringen Flächen, die nicht konvex sind und/oder innere Zwangskanten oder Zwangspunkte besitzen. Dann ist das Delaunay-Kriterium nicht für alle Dreiecke zu erfüllen.

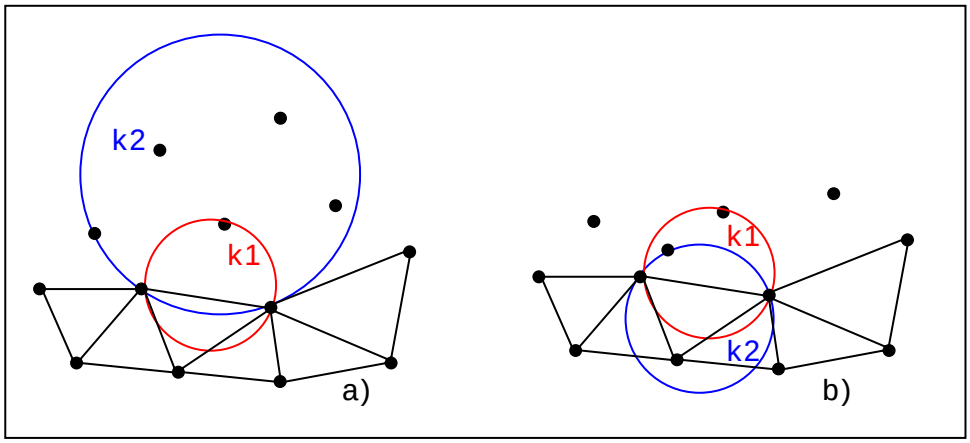


Abb. 7.15: Problematische Umkreise – Delaunay-Kriterium nicht erfüllt

Der in Abbildung 7.15 a) gezeigte Umkreis *k2* macht deutlich, dass die potentiellen Punkte für eine Dreiecksbildung genau auf der rechten Seite der aktiven Kante zu finden sein müssen. Im Beispiel a) ist der Punkt, der mit den Endpunkten der aktiven Kante den Kreis *k2* bildet, zwar rechts davon, aber nicht genau gegenüber der Kante. Dadurch wird der Umkreis *k2* sehr groß und beinhaltet somit weitere Punkte. Er wird nicht für eine Triangulierung herangezogen.

Der in Abbildung 7.15 b) gezeigte Umkreis *k2* zeigt, dass in manchen Fällen die minimale Höhe über der Ausgangskante andere Punkte der schon triangulierten Seite *H2* beinhalten kann. Dies tritt jedoch nur auf, wenn der optimale Punkt für eine Triangulierung nicht verwendet werden kann. Würde beispielsweise eine vorhandene Zwangskante bei der Dreiecksbildung geschnitten, wird ein schlechter Kandidat gewählt.

Für alle ermittelten Kandidatpunkte wird eine Berechnung der Umkreishöhe über der Ausgangskante durchgeführt. Für die Berechnung des besten Kandidaten wird wie folgt verfahren:

- Umkreisberechnung (Mittelpunktes)
- Umkreisberechnung (Radius)
- Berechnung des Abstandes vom Mittelpunkt zur Mitte der aktiven Kante
- Ergebnisberechnung in Abhängigkeit der Orientierung des Mittelpunktes zur Ausgangskante

Es wurden zwei Methoden für die Berechnung entwickelt, die beide nachfolgend vorgestellt werden. Die erste Methode ist eine mathematische Lösung - gut geeignet für den zweidimensionalen Raum. Sie ist leicht zu realisieren und ein sehr effizientes Verfahren. Die zweite Methode ist eine geometrische Lösung. Sie ist kompliziert in der Implementierung, aber ohne weiteres auf weitere Dimensionen übertragbar.

Mathematische Lösung

Die Ermittlung von Mittelpunkt und Radius erfolgt mit Hilfe des Satzes von Pythagoras.

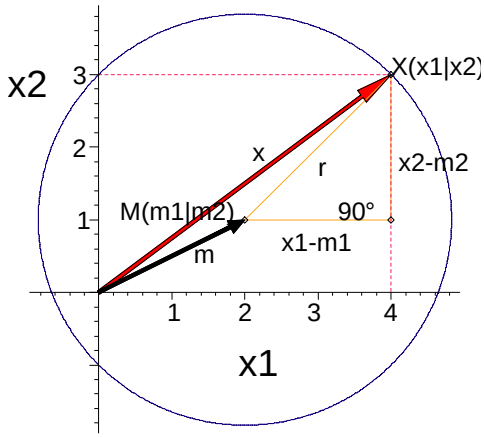


Abb. 7.16: Radiusermittlung durch Satz von Pythagoras (vgl. [Schweiz 99/01], S.171)

Aus Abb. 7.16 entnimmt man aufgrund des Satzes von Pythagoras, dass durch

$$r^2 = \left(x_1 - m_1\right)^2 + \left(x_2 - m_2\right)^2 \qquad (7.4)$$

die Punkte auf einem Kreis um $M(m_1 | m_2)$ mit dem Radius r bestimmt sind.

Satz: In der Ebene hat der Kreis um $M(m_1|\ m_2)$ mit dem Radius r > 0 die Gleichung $r^2 = (x_1 - m_1)^2 + (x_2 - m_2)^2$. Ist M der Ursprung, so lautet die Kreisgleichung: $x_1^2 + x_2^2 = r^2$ (vgl. [Schweiz 99/01], S.171).

Aus diesem Satz werden drei Gleichungen für die Berechnung des Mittelpunktes gebildet:

Es sind die Punkte $A(a_1|a_2)$, $B(b_1|b_2)$, $C(c_1|c_2)$. $M(m_1|m_2)$ ist zu ermitteln.

1. $(a_1-m_1)^2 + (a_2-m_2)^2 = r^2$
2. $(b_1-m_1)^2 + (b_2-m_2)^2 = r^2$
3. $(c_1-m_1)^2 + (c_2-m_2)^2 = r^2$

Die Auflösung der Gleichungen erfolgt durch Eliminationsverfahren. Von je zwei dieser Gleichungen werden Differenzen gebildet, dadurch fällt r^2 weg. Nach geeigneter Umformung der dadurch entstandenen zwei neuen Gleichungen wird erneut eine Differenz der beiden gebildet, um entweder m_1 oder m_2 zu eliminieren. Die Gleichung wird nach der verbleibenden Variablen aufgelöst. Damit kann auch die zweite Variable nach gängigen Verfahren berechnet werden. Die berechneten Werte m_1 und m_2 ergeben den Mittelpunkt $M(m_1|m_2)$.

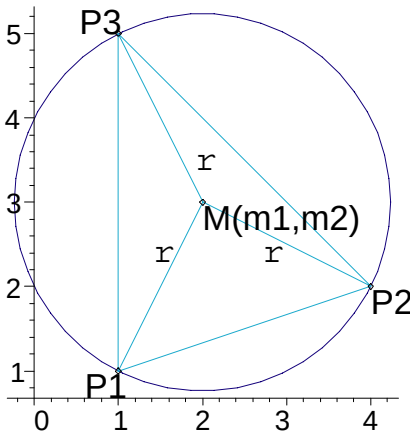


Abb. 7.17: Grafisches Beispiel zum Gleichungssystem

Wenn der Mittelpunkt vorliegt, ist es leicht, den Radius zu berechnen. Dafür fügt man lediglich den Mittelpunkt in einer der oben angegebenen Gleichungen ein und löst diese nach *r* auf.

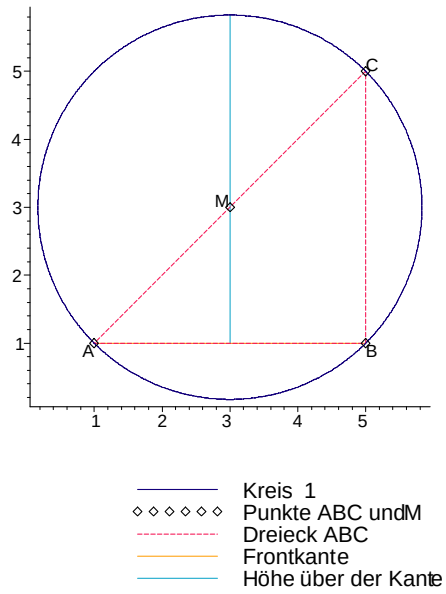


Abb. 7.18: Beispiel zur Höhenberechnung über der Ausgangskante

Für die Höhenberechnung über der Ausgangskante benötigt man neben dem Mittelpunkt und dem Radius auch noch die Länge der zu triangulierenden Kante, denn der Mittelpunkt liegt immer auf der Senkrechten der Seitenhalbierenden. Daraus resultiert auch die unten aufgeführte Formel zur Berechnung der Umkreishöhe über der Ausgangskante. Liegt der Mittelpunkt rechts, also über der Ausgangskante, berechnet sich die Höhe des Umreises durch Addieren der Werte des Radius und dem Abstand zwischen der Mitte der zu triangulierenden Kante und dem Mittelpunkt:

$$h = r + \sqrt{\left(\frac{1}{2}(b_1 + a_1) - m_1\right)^2 + \left(\frac{1}{2}(b_2 + a_2) - m_2\right)^2} \qquad (7.5)$$

Ein links von der Ausgangskante liegender Mittelpunkt ist ein Ausnahmefall, kann aber durch die schon beschriebenen Fälle von Zwangskanten vorkommen. Dann wird die Höhe des Umkreises über der Ausgangskante durch Subtraktion von Radius und Abstand zwischen Mittelpunkt und Mitte der Ausgangskante errechnet:

$$h = r - \sqrt{\left(\frac{1}{2}(b_1 + a_1) - m_1\right)^2 + \left(\frac{1}{2}(b_2 + a_2) - m_2\right)^2} \qquad (7.6)$$

Die berechnete Höhe des Umkreises über der Kante wird als Schlüssel mit dem Kandidatpunkt gespeichert.

Geometrische Lösung:

Basis für das hier verwendete Verfahren sind die Ausführungen der Arbeit [Littw 96/1] und die dort erarbeiteten mathematischen Formeln. Ausgehend davon, wird die Höhe des Umkreises über der Ausgangskante durch Verschieben und Drehen der Frontkante und des Kanditatpunktes ermittelt (vgl. [Littw 96/1], S. 43). Wie dort erklärt, wird für eine vereinfachte Ermittlung der Höhe *h* die Ausgangskante so in ein lokales Koordinatensystem verschoben, dass die Mitte der Kante im Ursprung liegt. Durch Drehung wird eine parallele Lage zur x-Achse erreicht.

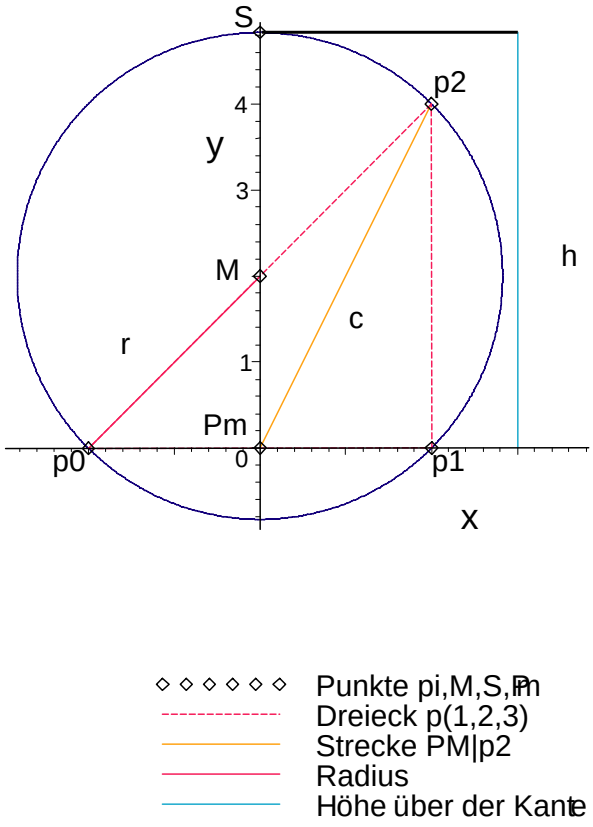


Abb. 7.19: Berechnung der Höhe *h* nach [Littw 96/1].

Wie man in Abb 7.19 sehen kann, schneidet die Mittelsenkrechte der Sehne AB den Kreis im Punkt *S*. Der Abstand ist die Höhe *h* über der Ausgangskante. Nachfolgend die Berechnung von *h* :

Es sind die Punkte p0($p0_x|p0_y$), p1($p1_x|p1_y$), p2($p2_x|p2_y$) gegeben.

Ermittlung von pm($pm_x|pm_y$)

$$\mathrm{pm}(\mathrm{pm_x}|\mathrm{pm_y}) = \left(\frac{1}{2}\left(p1_x + p0_x\right), \frac{1}{2}\left(p1_y + p0_y\right)\right) \tag{7.7}$$

Die absoluten Strecken c und l werden ermittelt

$$c = \sqrt{\left(pm_x - p2_x\right)^2 + \left(pm_y - p2_y\right)^2} \tag{7.8}$$

$$l = 2\sqrt{\left(pm_x - p0_x\right)^2 + \left(pm_y - p0_y\right)^2} \tag{7.9}$$

p0, p1, p2 werden im Verhältnis zu pm (liegt im Ursprung) in den Koordinatenursprung verschoben:

$$\begin{aligned} p0\_1 &= \left[p0_x - pm_x, p0_y - pm_y\right] \\ p1\_1 &= \left[p1_x - pm_x, p1_y - pm_y\right] \\ p2\_1 &= \left[p2_x - pm_x, p2_y - pm_y\right] \end{aligned} \tag{7.10}$$

Nach der Verschiebung wird p0_1 und p1_1 gedreht zu p0_2 und p1_2, die auf der x-Achse mit dem Abstand $-\frac{l}{2}, \frac{l}{2}$ liegen.

$$p0\_2 = \left[-\frac{1}{2}l\ ,\ 0\right] \tag{7.11}$$

$$p1\_2 = \left[\frac{1}{2}l\ ,\ 0\right] \tag{7.12}$$

Für die Drehung vom Kandidaten (p2_1 zu p2_2) wird der Winkel φ von p0_1 zu p0_2 ermittelt

$$\varphi = \arccos\left(\frac{\left(p0_x - pm_x\right) p0\_2_x}{\sqrt{\left(p0_x - pm_x\right)^2 + \left(p0_y - pm_y\right)^2}\ \sqrt{p0\_2_x^2}}\right) \tag{7.13}$$

Falls der Punkt p1_1 nach dem Verschieben oberhalb der x-Achse liegt, muss der Winkel von 360 Grad (2*Pi) abgezogen werden,

$$\varphi = 2\pi - \varphi \tag{7.14}$$

andernfalls ist der Winkel korrekt.

Jetzt kann auch der Kandidat p2_1 zu p2_2 gedreht werden. Für eine Drehung entgegengesetzt dem Uhrzeigersinn um den Ursprung wird eine Drehmatrix definiert:

$$Drehmatrix = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \tag{7.15}$$

$$p2\_2 = \begin{bmatrix} \cos(\varphi)(p2_x - pm_x) - \sin(\varphi)(p2_y - pm_y), \\ \sin(\varphi)(p2_x - pm_x) + \cos(\varphi)(p2_y - pm_y) \end{bmatrix} \tag{7.16}$$

Berechnung Strecke *b* von *pm* zum Umkreismittelpunkt *M* nach [Littw 96/1], Seite 43, (5.18):

$$b = \frac{c^2 - \frac{1}{4}l^2}{2p2\_2_y} \tag{7.17}$$

Berechnung Radius *r* nach [Littw 96/1], Seite 42, (5.14):

$$r = \sqrt{b^2 + \frac{1}{4}l^2} \tag{7.18}$$

Die Höhe *h* des Umkreises ergibt sich demzufolge durch:

$$h = \sqrt{b^2 + \frac{1}{4}l^2} - b \tag{7.19}$$

Durch Koordinatentransformation und Rotation wird die Berechnung der Umkreishöhe weniger rechenaufwendig. Wirklich transformiert wird nur *p2* – für die weiteren Berechnungen ab (7.17) wird jedoch lediglich die Transformation von $p2_y$ benötigt.

Für die Berechnung der Umkreishöhe wurde die geometrische Lösung implementiert. Auch wenn die mathematische Lösung durch simplere Berechnung schneller ist, kann sie definitiv nur für Berechnungen in der Ebene verwendet werden. Die geometrische Lösung ist auch auf andere Dimensionen übertragbar und findet darum hier Anwendung.

Auf Überschneidung prüfen

Die Kandidatpunkte sind nach der Festellung ihrer Höhe aufsteigend nach dieser in einer Liste sortiert. Der am Index 0 gespeicherte Punkt ist am besten für eine Dreiecksbildung mit der aktiven Kante aus der Front geeignet. Es kommt vor, dass zwei Punkte den gleichen Höhenwert haben, dann entscheiden die weiteren Prüfungen, welcher von beiden trianguliert wird. Erfüllen sie die gleichen Kriterien, ist es unerheblich, welcher Punkt benutzt wird. In der Reihenfolge wird dann der mit dem kleineren Index bevorzugt.

Liegen Zwangskanten oder nicht konvexe Flächen vor, ist es möglich, dass der Punkt zur nicht sichtbaren Seite der aktiven Frontkante gehört. Überschneidungen mit anderen Kanten der aktiven Front sind nicht erlaubt. Daraufhin werden die beiden neu entstehenden Verbindungsstrecken zum Kandidaten geprüft.

Hier werden die schon in der Anwendung STDSS existierenden implementierten Funktionen benutzt, welche folgende Möglichkeiten der Überschneidung untersuchen:

- Zwei Kanten schneiden sich in ihren Rändern
- Der Rand einer Kante schneidet das Innere einer Kante
- Zwei Kanten schneiden sich in ihrem Inneren

Zusätzlich werden die Verbindungsstrecken noch auf Parallelität geprüft. Trifft die Parallelität nicht zu und sind keine Schneidungen festzustellen, wird geprüft, ob alle anderen Kandidatpunkte außerhalb des neu zu bildenden Dreiecks liegen.

Falls ein Kandidat diese Prüfung nicht besteht, wird er verworfen und der nächste der Liste wird untersucht. Dies wird so lange durchgeführt, bis ein Punkt diese Prüfungen bestanden hat.

Erst dann wird ein neues Dreieck aus Start- und Endpunkt der aktiven Kante sowie dem für optimal befundenen Kandidatpunkt in der schon beschriebenen Struktur erzeugt und die Front-Liste aktualisiert.

Für weitere Bearbeitungsschritte des fertig gestellten Netzes, die unten erklärt werden, ist es von Vorteil, zuerst ein topologisches Nachbarschaftsverhältnis zwischen den Dreieckselementen herzustellen. Dies kann bereits während der Vernetzung geschehen oder nach der Fertigstellung in einer Nachbearbeitungsmethode für alle Dreiecke. Hier wird dies am Nachbearbeitungsschritt des fertigen Netzes erklärt.

Nachbarschaftsbeziehungen

Es wurde vereinbart, dass ein Dreieck immer drei Nachbarn besitzt – für jede Kante einen. Diese Nachbarn können Dreiecke oder selbst Kanten, z.B. Randkante, sein. Zuerst wird die Nachbarschaft zu den Rand- und Zwangskanten hergestellt.

Zuerst werden die Randkanten mit den Dreiecken verglichen. Ist eine Randkante gleich einer Dreiecksseite, wird diese zum Nachbarn dieser Seite und das Dreieck wird Nachbar der Randkante.

Danach werden die inneren Zwangskanten auf ihre Nachbarschaft hin zu den Dreiecken überprüft. Hier hat eine Zwangskante zwei Dreiecke zum Nachbarn. Über diese Nachbarschaftsbeziehungen ist es möglich, einen Dreiecksnachbarn auf der nicht sichtbaren Seite einer Zwangskante zu ermitteln.

Zum Schluss werden die Dreiecke an den noch nicht besetzten Seiten auf die direkte Nachbarschaft zu einem anderen Dreieck hin überprüft. Die Nachbarschaft für beide Dreiecke wird hergestellt, wenn sich zwei Dreiecke eine Kante teilen.

7.3.4 Netzoptimierung

Die Qualität der Netze entspricht nicht immer den erwarteten Anforderungen. Darum versucht man nachträglich, die Netze zu optimieren, indem man die Eigenschaften von Gittern hinsichtlich bestimmter Qualitätskriterien verbessert. Dabei wird auf eine Verbesserung der Elementform und des Überganges der Elementgrößen hingewirkt. Beispielsweise kann ein zu grober Übergang zwischen verschieden großen Elementen bei einem Finite-Elemente-Netz numerische Diffusionen erzeugen, was zu schlechten Ergebnissen oder Instabilität führt.

Es gibt verschiedene Kriterien zur Netzelementbewertung. Hier taucht wieder der Begriff "Wohlgeformtheit" des Elementes auf, was laut diverser Literatur gleichbedeutend mit einem gleichseitigen Dreieck ist.

Qualitätsbewertung

Nicht nur beim Vernetzungsprozess selbst, auch bei der anschließenden Verbesserung ist die geometrische Bewertung der Qualität einzelner Netzelemente notwendig. Die Beschreibungen für die Qualität sind topologischer oder geometrischer Natur. Allgemein gesehen kann das Netz durch die Summe des gewählten Indikators q über alle Netzelemente i beurteilt werden:

$$Q_{gesamt} = \frac{1}{n}\sum_{i=1}^{n} q_i \qquad (7.20)$$

In der Literatur (vgl.[Burg 05/1], [Hom 05/1]) findet man diverse Qualitätskennwerte für Elemente:

Kriterium	Gleichseitiges Dreieck
β = Umkreisradius / Inkreisradius	β = 3.0
σ = Maximale Kantenlänge / Inkreisradius	σ = 4.8989
ω = Umkreisradius/Maximale Kantenlänge	ω = 0.6125
ζ = Maximale Kantenlänge / Minimale Kantenlänge	ζ = 1.0
s = Innkreisradius/Umkreisradius (Sliver)	s = 0.5
a = (4* $\sqrt{3}$ * A)/ Summe(li^2) Formkriterium	a = 1.0
b = Umkreisradius/minimale Kantenlänge	b = 0

Da man durch vorangegangene Berechnungen schon den Umkreisradius der Dreiecke ermittelt hat, ist es sinnvoll, einen Indikator zu verwenden, in dem dieser Wert eingeht.

Littwin (vgl. [Littw 96/1], S. 65) testete bereits das Umkreis-/Inkreis-Kriterium und erzielte damit gute Ergebnisse. Ein Nachteil ist, dass es keine obere Schranke gibt.

Ein Verfahren, welches vergleichbare Ergebnisse erzielt, ist das Innkreis-/Umkreis-radius-Kriterium. Es wird zur Zeit für die Qualitätsberechnung verwendet. Dieses Kriterium wurde lt. Burkhardt (vgl.[Burg 05/1]) von Schimada u.a. 1997 als Geometriekriterium eingeführt.

Für gleichseitige Dreiecke ist das Verhältnis 0.5 und für stumpfwinklige Dreiecke strebt es gegen Null.

Die Gesamtqualität des Netzes wird definiert durch:

$$E(g) = \frac{1}{n}\sum_{i=1}^{n} 0.5 - \frac{r_i}{R_i} \tag{7.21}$$

Je kleiner E(g) desto regulärer die Netzqualität. Als Obergrenze kann der Wert 1 angenommen werden.

Formeln für die oben beschriebene Berechnung:

$$r_u = \frac{a}{2\sin\alpha} = \frac{b}{2\sin\beta} = \frac{c}{2\sin\gamma} \tag{7.22}$$

$$\begin{aligned} U(E) &= a + b + c \\ a &= \left(p0_x - p1_x\right)^2 + \left(p0_y - p1_y\right)^2 \\ b &= \left(p1_x - p2_x\right)^2 + \left(p1_y - p2_y\right)^2 \\ c &= \left(p2_x - p0_x\right)^2 + \left(p2_y - p0_y\right)^2 \end{aligned} \tag{7.23}$$

$$s = \frac{(a+b+c)}{2} = \frac{U}{2} \tag{7.24}$$

$$A = \sum_{i=1}^{2} \left| \frac{1}{2} \left(\left(p0_x - p_{i_x} \right) \left(p0_y + p_{i_y} \right) + \left(p_{i_x} - p_{i+1_x} \right) \left(p_{i_y} + p_{i+1_y} \right) + \left(p_{i+1_x} - p0_x \right) \left(p_{i+1_y} + p0_y \right) \right) \right| \tag{7.25}$$

$$r_i = \frac{A}{s} = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}} \tag{7.26}$$

Netzverbesserung

Um die Qualität eines Netzes zu verbessern, versucht man durch Nachbearbeitung des fertigen Netzes möglichst viele "gute" Dreieckselemente zu erhalten.
Um eine Verbessung des Netzes zu erzielen, können sich zwei Verfahren ergänzen:

- Topologisches Tauschen

Das Formkriterium ist nach Burkhardt (vgl.[Burg 05/1]) ein guter Indikator für den topologischen Kantentausch. Dabei wird die Diagonale zwischen zwei benachbarten Dreiecken gewechselt um das Delaunay-Umkreiskriterium zu erfüllen.

Das topologische Tauschen wurde im Zusammenhang mit dem Formkriterium schon erwähnt. Hierbei werden die gemeinsamen Kanten zweier benachbarter Dreiecke vertauscht. Dies wird häufig durchgeführt, wenn die Elementform nicht dem Delaunay-Kriterium entspricht oder die Anzahl derer an einem Knoten angeschlossener Elemente verringert werden soll. Das nachfolgende Beispiel zeigt, dass ein Kantentausch nicht immer realisierbar ist.

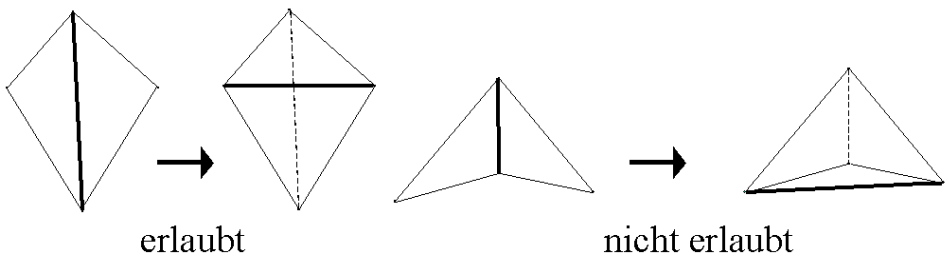


Abb. 7.20: Topologisches Tauschen(vgl. [Burg 05/1])

- Geometrisches Verfahren

Der hier verfolgte Ansatz ist die sogenannte Netzentspannung oder auch Netzglättung (smoothing) genannt. Dieses Verfahren basiert auf einem kontrollierten Verschieben von Knoten der Triangulation, ohne etwas an der Gittertopologie zu verändern.

Ein bekanntes und häufig verwendetes Verfahren geht auf Laplace im Jahre 1960 zurück und wird als Laplace-Glättung bezeichnet.

Bei der Laplace-Glättung wird beispielsweise ein innerer Netzknoten p in die arithmethische Mittelpositition seiner Nachbarn verschoben:

$$p' = \frac{1}{n}\sum_{i=1}^{n} p_i \qquad (7.27)$$

Es kann vorkommen, dass ein Schwerpunkt außerhalb des Polygons liegt, welches seine umliegenden Nachbarn bilden. Diese Fälle müssen gesondert behandelt werden.

Die Verschiebung verbessert die Größe und Form der umliegenden Dreiecke von p, aber dies ist nicht garantiert.

Bessere Ergebnisse erzielt die Variation der Laplace-Glättung, die baryzentrische Glättung. Sie stellt eine gewichtete Verschiebung dar, die durch zwei Methoden realisiert werden kann.

Baryzentrische Glättung durch Flächengewichtung:
Der innere Netzknoten p wird in den Flächenschwerpunkt seiner umliegenden Dreiecke verschoben.

$$p_s' = \frac{\sum_{i=1}^{n} A_i p_{s_i}}{\sum_{i=1}^{n} A} \qquad (7.28)$$

Baryzentrische Glättung durch Längengewichtung :

Die Verbindungsstrecken der umliegenden Knoten von *p* werden für eine Verschiebung des Mittelknotens herangezogen. Durch Verkürzen oder Verlängern der Verbindungsstrecken wird für *p* eine neue Mittelposition gefunden:

$$p' = \frac{\sum_{i=1}^{n} l_i p_i}{\sum_{i=1}^{n} l} \qquad (7.29)$$

Zum Testen wurden alle drei Glättungsmethoden implementiert. Bei der Netzerstellung kann man die Glättungsmethode und die Anzahl der Durchgänge auswählen. Die Qualitätsermittlung für das gesamte Netz und der einzelnen Elemente wird in einer Statistik ausgegeben.

Die Tabelle zeigt Effektivität und Effizienz der Glättungsmethoden:

	Einheitsquadrat (4779 Dreiecke)	Watershed mit Fluss (3664 Dreiecke)	Watershed mit Fluss 199 Dreiecke)
Zeit (in Sekunden) ohne Glättung:	12.06	12.141	0.297
Knotengewichtung			
Zeit (in Sekunden):	53.391	37.0	0.469
Netzqualität:	0.4558	0.4470	0.4230
höchste Kategorie 0.46 - 0.5:	2795 Dreiecke	1899 Dreiecke	102 Dreiecke
Bestes Ergebnis nach Durchgängen:	5	3	5
höchste Kategorie 0.46 - 0.5:	2795 Dreiecke	1902 Dreiecke	102 Dreiecke
Flächengewichtung			
Zeit (in Sekunden)	51.906	36.359	0.469
Netzqualität:	0.3321	0.4002	0.4193
höchste Kategorie 0.46 - 0.5:	233 Dreiecke	915 Dreiecke	83 Dreiecke
Bestes Ergebnis nach Durchgängen:	0	1	1
höchste Kategorie 0.46 - 0.5:	1500 Dreiecke	1354 Dreiecke	84 Dreiecke
Längengewichtung			
Zeit:	32.593	25.469	0.375
Netzqualität:	0.4063	0.4231	0.4130
höchste Kategorie 0.46 - 0.5:	1288 Dreiecke	1274 Dreiecke	82 Dreiecke
Bestes Ergebnis nachDurchgängen:	0	3	2
höchste Kategorie 0.46 - 0.5:	1500 Dreiecke	1274 Dreiecke	82 Dreiecke

Tabelle 4: Übersicht Glättungsmethoden

Aus der Tabelle kann man entnehmen, dass von diesen drei Verfahren für die Netzglättung das knotengewichtete am effektivsten ist und die höchsten Qualitätswerte erreicht. Am effizientesten ist das längengewichtete Verfahren mit etwas geringerer Qualität. Das flächengewichtete Verfahren weist bei einer verfeinerten Diskretisierung des Randes und der Zwangskanten schlechtere Ergebnisse auf. Bei relativ homogenen Elementen erzeugt dies Verfahren vergleichbare Ergebnisse zu den anderen Verfahren. Die Erstellungszeit des Netzes mit diesem Verfahren ist mit dem knotengewichteten Verfahren vergleichbar. Wie man ebenfalls sehen kann, sollte man diese Glättungsdurchgänge nicht mehr als 5-mal durchführen. Bessere Ergebnisse erhält man oft nach drei Durchgängen. Es kommt sogar vor, dass das Netz auch ohne Glättung qualitätsmäßig gute Ergebnisse erzielt, dies ist jedoch abhängig von der Form und der Diskretisierung der zu vernetzenden Fläche.

In der Literatur, wie [Boro 05/1] und [Chen 05/1] finden sich weitergehende Konzepte, die eine Erweiterung der oben geschilderten klassischen Verfahrensweisen darstellen.

7.3.5 Speicherung TIN

Das Netz, welches bis dorthin nur als Liste von Dreiecken besteht, wird über eine Schnittstelle als Surface-Patches permanent als TIN gespeichert. Dazu benutzt man wieder die Geometrie sowie evtl. die topologischen Verhältnisse, wie die Nachbarschaft. Die Punkte der Dreiecke erhalten zusätzlich zu ihrer x- und y-Koordinate ein oder mehrere Attribute, z.B. Höhe oder Niederschlagsmessungen. Attributwerte für einen Punkt an beliebiger Stelle der Oberfläche können dann interpoliert werden.

7.3.6 Visualisierung

Nachfolgend werden grafische dargestellte Beispiele der Vernetzung mit dem Netzgenerator des STDSS gezeigt:

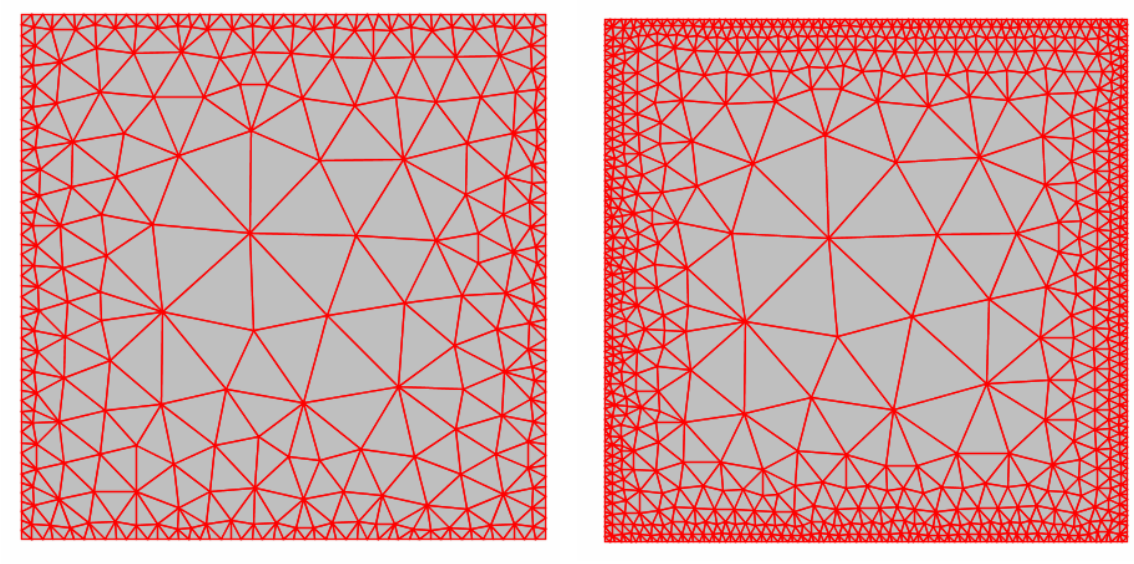


Abb. 7.21: Einheitsquadrat: (li) 591 Dreiecke und (re) 1655 Dreiecke

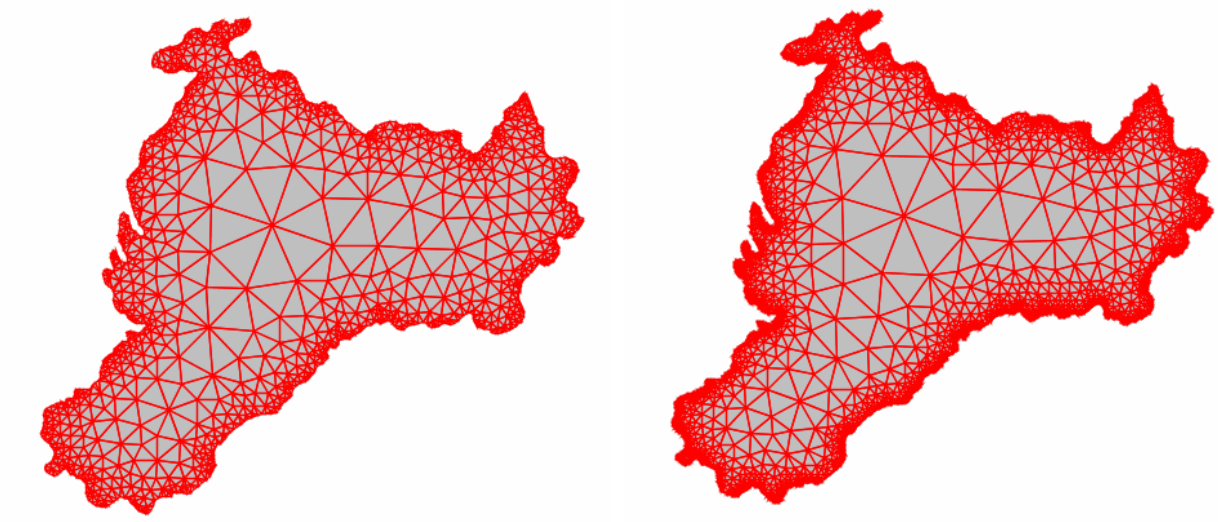


Abb. 7.22 : Beispiel Watershed: (li) 2166 Dreiecke und (re) 6776 Dreiecke

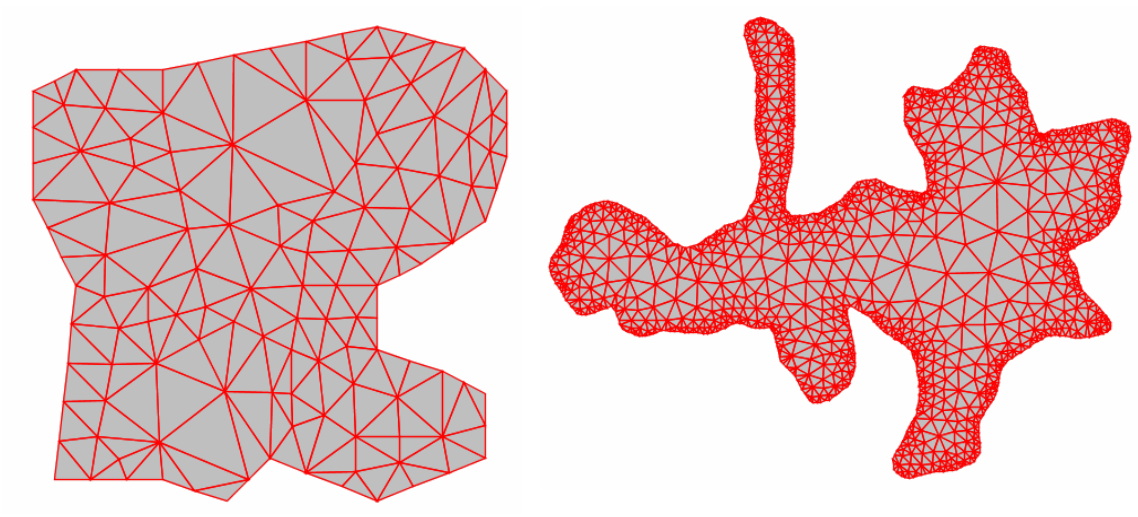


Abb. 7.23: Watershed mit Fluss 199 Dreiecke und Teresapolygon mit 2138 Dreiecke

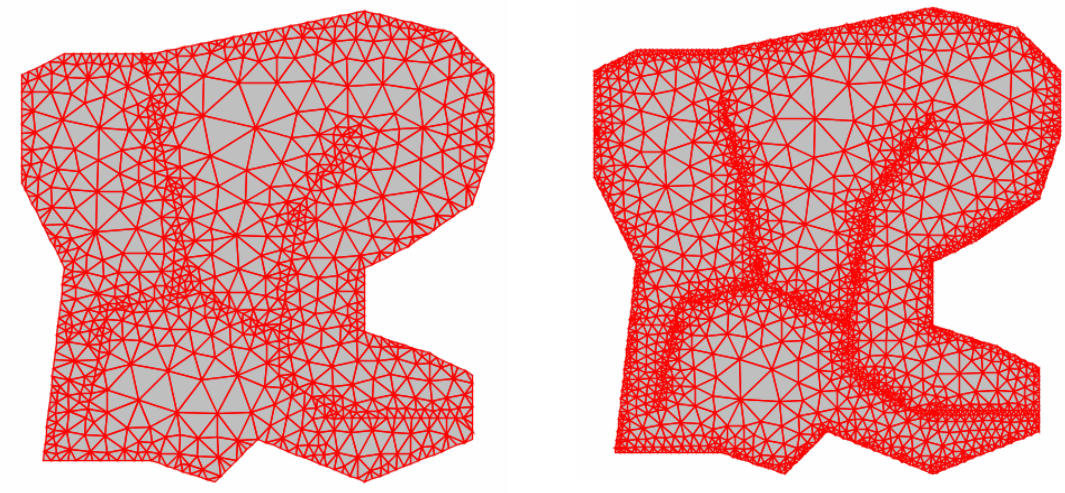


Abb. 7.24: Watershed mit Fluss: (li) 1580 Dreiecke und (re) 3664 Dreiecke

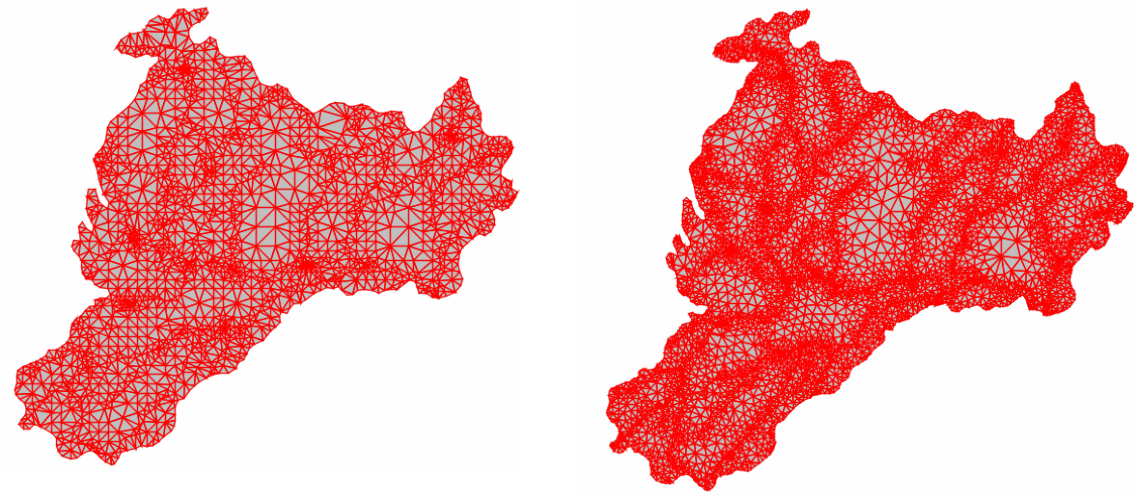


Abb. 7.25: Watershed Rio do Peixe: (li) 3839 Dreiecke und (re) 8906 Dreiecke

8. Zusammenfassung und Ausblick

Eine erfolgreiche Implementation eines Netzgenerators ist von verschiedenen Faktoren abhängig. Zum einen ist die richtige Wahl aus der Vielzahl der Datenstrukturen und Algorithmen zur Umsetzung der gestellten Anforderungen entscheidend. Zum anderen die Wahl der Programmierumgebung. Auch hier müssen die Anforderungen erfüllbar sein. Ein Kriterium wäre hier die Objektorientiertheit. Manche Programmiersprachen arbeiten noch zum Teil prozedural und haben Probleme bei der Vererbung. Zweites Kriterium wäre eine Plattformunabhänigikeit. Die Anforderungen an die Programmierumgebung müssen jedoch nicht immer einen programm-technischen Hintergrund haben. Auch wirtschaftliche Aspekte können eine Rolle spielen, z. B. Kosten für Standardbibliotheken und Standardsoftware.

Des weiteren stehen die Anforderungen des Anwenders im Vordergrund. Diese sollten genau spezifiziert sein, z.B. durch Angabe der zu verwendenden Datenart und die anschließende Weiterverarbeitung. Denn es bestätigt sich die Aussage der Bundesregierung in ihrer Stellungnahme: Es gibt kein System, das sämtliche Aufgaben, die allgemein im GIS-Umfeld auftauchen, lösen kann (vgl. [BuReg 00/1], Seite 5).

Durch diese speziellen Anforderungen des Anwenders an den Netzgenerator unterliegt dieser einer ständigen Weiterentwicklung. Beispielsweise berücksichtigt die eingefügte Vorverarbeitungsprozedur für die Zwangskanten nicht alle Funktionen ausgereifter Preprocessing-Tools. Solche sollten an anderer Stelle im STDSS als umfangreiches Toolset zur Verfügung gestellt werden.

Der Netzgenerator ist zur Zeit auf die Triangulation von einer Oberfläche ausgelegt. Zukünftig sollen auch verbundene Oberflächen, sogenannte *Surface-Patches,* trianguliert werden können. Weiterhin ist es durch die Berücksichtigung geometrischer Algorithmen auch möglich, eine 3D-Erweiterung zu implementieren.

Literaturverzeichnis

[Bal 99/1]
Balzert, Heide: Lehrbuch der Objektmodellierung. Analyse und Entwurf. Heidelberg: Spektrum 1999

[Bar 95/1]
Bartelme, Norbert: Geoinformatik. Modelle, Strukturen, Funktionen. Berlin: Springer-Verlag 1995

[Berg 97/1]
Berg, M. de / Kreveld / Overmars / Schwarzkopf: Computational Geometry. Algorithms and Applications. Berlin: Springer-Verlag 1997

[Bil 91/1]
Bill, R./ Fritsch, D.: Grundlagen der Geo-Informationssysteme. Band 1: Hardware, Software und Daten. Karlsruhe: Wichmann-Verlag GmbH 1991

[Boro 05/1]
George, H. Borouchaki: Delaunay Triangulation and Meshing, Application to Finite Elements, Edition Hermes, Paris, 1998

[Endl 85/1]
Endl, Kurt (Prof. Dr.), Mathem. Institut Gießen: Analytische Geometrie und lineare Algebra, Düsseldorf: VDI-Verlag GmbH 1985

[ESRI 98/01]
ESRI: Shapefile Technical Description. ESRI White Paper, July 1998 (PDF-Dokument)

[Hoschek 92/1]
Hoschek (Prof. Dr.) / Lasser (Dr.): Grundlagen der geometrischen Datenverarbeitung, 2. Auflage. Stuttgart: B.G. Teubner 1992

[Littw 96/1]
Littwin, Roland: Automatische Netzgenerierung für Finite Elemente Berechnungen, Diplomarbeit 1996

[OpenGis 01/1]
OpenGisConsortioums: OpenGIS Feature Geometry des ISO/TC 211 19107 Geographic information – Spatial schema , Version 5. Editor: Dr. John R. Herring, 2001

[Ottm 02/1]
Ottmann, T. / Widmayer, P.: Algorithmen und Datenstrukturen, 4.Auflage. Berlin: Spektrum Akademischer Verlag 2002

[Roehrig 03/1]
Roehrig, Jackson (Prof. Dr.-Ing.): GIS – Geografisches Informationssystem, Teil 2. Vorlesung SS 2003 (PP Präsentation)

[Samet 95/1]
Samet, Hanan: Applications of spatial data structures: computer graphics, image processing and GIS. Verlag: Addison-Wesley 1995

[Schweiz 99/01]
Schweizer, W. / Schmidt, A.: Analytische Geometrie, 1. Auflage. Tübingen: LS 1999

[Sed 92/01]
Sedgewick, Robert: Algorithmen in C++, 3. Auflage. Verlag: Addison-Wesley 1997

[Ullenboom 02/1]
Ullenboom, Christian: Java ist auch eine Insel. Programmieren für die Java-2-Plattform in der Version 1.4, 2. Auflage, Online Version, Galileo Computing, 2002

[West 03/1]
Westenberger, Hartmut, Prof. Dr.: Betriebliche Anwendungen II. Data Warehousing. Fachhochschule Köln, Vorlesung 2003

[Zeiler 99/1]
Zeiler, Michael: Modelling our world. The ESRI's Guide to Geodatabase Design. ESRI Version 1999

Internetdokumente

[Bern 05/1]
Bern, Marshall: Mesh Generation and optimal Triangulation.
http://www.devdept.com/fem/docs/BernEppstein.pdf (02.04.2005)

[BuReg 00/1]
Bundesregierung: Geografische Informationssysteme (GIS). Stellungnahme: 11.10.2000
dip.bundestag.de/btd/14/041/1404139.pdf
dip.bundestag.de/btd/14 /032/1403214.pdf(11.05.2004)

[Burg 05/1]
Burghardt, Michael: Parallele Netzgenerierung für ebene und räumliche Problemstellungen aus dem Bauwesen, Dissertation
www.michael-burghardt.de/diss/index.html (17.01.2005)

[Chen 05/1]
Zhijian Chen: Combined Laplacian and Optimization-Based Smoothing for Quadratic mixed Surface Meshes, www.imr.sandia.gov/papers/imr12/chen03.pdf (02.04.2005)

[Com 05/1]
Computerbase: Lexikon
www.computerbase.de/lexikon/Geoinformationssystem (02.04.2005)

[fh-mainz 05/01]
Wozu GIS ? www2.geoinform.fh-mainz.de/~zipf/arcInfoBuch/x236.html (11.05.2004)

[GIS 04/1]
GIS-Tutor Version 3.0., www.gis-tutor.de (11.05.2004)

[Good 04/1]
Goodchild, M.F. und K.K. Kemp, eds. (1990), National Center for Geographic Information Analysis (NCGIA): Unit 59- Spatial - Descision Support Systems
http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/toc.html
http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u59.html#UNIT59 (20.05.04)

[Hom 05/1]
Homann: Grundlagen der Gittermodellierung
http://www.bauinf.tu-cottbus.de/mitarbeiter/homann/DISS/Kap3.html (21.02.2005)

[Kind 03/1]
Kindernothilfe, www.kindernothilfe.de/downloads/g11.swf (25.03.2003)

[Kind 03/2]
Kindernothilfe, ww.kindernothilfe.de/schwerpunkte/unterseiten/02052/index_main.html (25.03.2003)

[Schwen 04/1]
Schwentick, T.: Effiziente Algorithmen 2004 17. Algorithmische Geometrie: Einleitung
http://www.mathematik.uni-marburg.de/~tick/Pages/Lehre/EA1/geomd.pdf
(02.04.2005)

[UniBerkl 05/1]
Meshing and Triangulation Lecture 1
www.cs.berkeley.edu/~jrs/mesh/1/lecture.html (02.04.2005)

Abbildungsverzeichnis

Tabellenverzeichnis

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht.

Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, den 28. Juni 2005

Liane Ziethen-Aksoy