UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Prozedural erstellte digitale Landschaftsmodelle auf der Basis von ATKIS-DLM und DGM Daten

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers
im Studiengang Computervisualistik

vorgelegt von
Stefan Rilling

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dipl. Inf. Thorsten Grosch Institut für Computervisualistik, AG Computergraphik

Koblenz, im Dezember 2006

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Danksagung

Allen sei gedankt, die mich während meines Studiums durch ihre Tatkraft, ihr Engagement und ihre Leistungen unterstützt haben. Besonderer Dank gilt hier meinen Eltern, die mir dieses Studium ermöglicht haben. Weiterhin gilt mein besonderer Dank all denen, die mir bei der Anfertigung dieser Diplomarbeit hilfreich zur Seite standen, allen voran meinem Betreuer Prof. Müller, der sich für mich stets die nötige Zeit genommen hat und mich mit kompetentem Rat hilfreich unterstüzte. Ferner möchte ich mich bei all den Menschen bedanken, die ihre Zeit für die Korrektur dieser Ausarbeitung geopfert haben: Meine Mutter, welche den Großteil der äußeren Form dieser Arbeit korrigierte, sowie Fabian Scheer, Armin Meyer und Andreas Mosig, die mir in allen inhaltlichen Fragen stets mit Rat und Tat zur Seite standen. Bedanken möchte ich mich auch bei den Mitarbeitern des Landesamtes für Vermessung und Geobasisinformation Rheinland Pfalz, die mir die Datenbasis für diese Arbeit schnell und unbürokratisch zur Verfügung gestellt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

Digitale Landschaftsmodelle haben in den letzten Jahren verstärkt Einzug in das tägliche Leben gehalten. Wenn raumbezogene Information veranschaulicht werden soll, wird immer häufiger auf dreidimensionale Visualisierungen zurückgegriffen. Die Verwendung von digitalen Kartensystemen und Routenplanern ist längst für einen Großteil der Bevölkerung selbstverständlich geworden. Spätestens seit der Entwicklung von Google Earth™haben sich dreidimensionale Landschaftsvisualisierungen im kollektiven Gedächtnis manifestiert.

1.1 Motivation

Ein Standardverfahren zur Visualisierung dreidimensionaler Landschaften ist sicherlich die Texturierung eines Höhenmodells der Landschaft mit Satellitenbildern bzw. Luftbildaufnahmen. Dies ist der Ansatz welcher z.B. auch in Google Earth™verfolgt wird. Bei Bedarf bzw. vorhandener Datenlage werden dann Gebäude und Straßen über das so texturierte Terrainmodell gelegt. Der Vorteil dieses Verfahrens ist, dass das so erstellte Modell bei Betrachtung aus großer Entfernung sehr realistisch wirkt. Gleichzeitig wird bei der Betrachtung aus der Nähe schnell deutlich, dass die durch die Textur dargestellten Geländemerkmale letztendlich keine räumliche Ausdehnung besitzen, das Geländemodell wirkt dadurch relativ statisch und unrealistisch. Die eingefügten dreidimensionalen Objekte wirken, in Verbindung mit dem flachen Eindruck der Textur, aufgesetzt und aus dem Gesamtbild herausgerissen. Ein deutlich plausiblerer Eindruck des Landschaftsbildes kann durch die vollständige dreidimensionale Modellierung der Landschaft erreicht werden.

1.2 Zielsetzung

In dieser Arbeit soll ein dreidimensionales, echtzeitfähiges Landschaftsmodell des Mittelrheintals erstellt werden. Dabei soll die Modellerstellung so weit wie möglich automatisiert werden. Als Datengrundlage dienen das digitale Landschaftsmodell *ATKIS-Basis DLM* sowie das *digitale Geländemodell (DGM)*, welches die notwendigen Höheninformationen zur Erzeugung des dreidimensionalen Modells enthält. Insbesondere soll dabei untersucht werden, wie sich die Generierung von Landschaftsmerkmalen wie Infrastruktur und Vegetation durch ein parametrisierbares Modell automatisieren lässt, und in wie weit sich die verwendeten Daten für einen solchen Automatisierungsprozess eignen.

1.3 Aufbau der Arbeit

Diese Arbeit gliedert sich thematisch in acht Teile. Zunächst wird in Kapitel 2 auf die vermessungstechnischen Grundlagen eingegangen, welche zur Benutzung der verwendeten Daten nötig sind. Kapitel 3 beschreibt die verwendeten Datenquellen ATKIS-Basis DLM und DGM im Detail. In Kapitel 4 wird die Erstellung eines dreidimensionalen Oberflächenreliefs aus den verwendeten Daten beschrieben. Hierbei wird insbesondere auf die Erstellung eines triangulierten Modells der Landschaft eingegangen. In Kapitel 5 werden die Möglichkeiten zur prozeduralen Texturerzeugung beleuchtet. Dabei werden als Basisverfahren Perlin Noise und der Cellular Algorithmus vorgestellt. Kapitel 6 zeigt die Möglichkeiten prozeduraler, automatisierter Generierung von Infrastruktur am Beispiel von Schienen- und Straßenverkehrswegen auf. In Kapitel 7 werden Ansätze für ein parametrisierbares Vegetationssystem aufgezeigt. In Kapitel 8 werden die in dieser Arbeit implementierten Verfahren zusammengefasst und die erzielten Ergebnisse einer kritischen Betrachtung unterzogen. Kapitel 9 soll einen abschließenden Ausblick auf ausstehende Aufgaben geben und Möglichkeiten für zukünftige Arbeiten aufzeigen.

1.4 Hinweis zu den Vermessungsdaten

Die Datensätze ATKIS-Basis DLM sowie das digitale Höhenmodell (DGM) wurden für diese und andere Arbeiten an der Universität Koblenz vom *Landesamt für Vermessung und Geobasisinformation Rheinland Pfalz*[1] kostenfrei zur Verfügung gestellt. Die Nutzung der Daten sowie die Präsentation der damit erzielten Ergebnisse erfolgt mit freundlicher Genehmigung der genannten Behörde. Für die gesamte Arbeit gilt folgender Erlaubnisvermerk:
"Geobasisdaten (DLM, DGM, DOP), ©Landesamt für Vermessung und Geobasisinformation Rheinland - Pfalz vom 12.05.2006; Az.: 26 722-1.11/uni)"

[1] http://www.lvermgeo.rlp.de/indexlvermgeo.html

2 Geodäsie

Die Ursprünge der Geodäsie, der „Wissenschaft von der Ausmessung und Abbildung der Erdoberfläche“ [2], lassen sich bis in das alte Ägypten zurückverfolgen. Schon zu damaliger Zeit war die Aufteilung von Land und die Definition von Grundstücks- und Landesgrenzen notwendig, und auch heutzutage bauen eine Vielzahl von Entwicklungsprozessen und planerischen Entscheidungen auf geodätischem Datenmaterial auf, Architektur- und Bauwesen sei hier als eines von vielen Beispielen zu nennen. Grundsätzlich lässt sich die Geodäsie in die Teilbereiche *Höhere Geodäsie* und *Niedere Geodäsie* aufteilen.
Die **Höhere Geodäsie** beschäftigt sich hierbei u.a. mit der Berechnung des Erdschwerefeldes, der Erstellung von präzisen Referenz- und Koordinatensystemen sowie der mathematischen Beschreibung der Erdfigur (vgl. Kapitel 2.1). Ferner wird auch die *Landesvermessung* in den Bereich der Höheren Geodäsie eingegliedert. Gerade die aus der Landesvermessung resultierenden Daten, hauptsächlich topographische Kartenwerke, sind für diese Arbeit von zentraler Bedeutung.
Die **Niedere Geodäsie** hat im Gegensatz dazu hauptsächlich die Erstellung von Liegenschaftskatastern, d.h. Lageplänen von Flurstücken und Gebäuden, zur Aufgabe.

2.1 Erdfigur - Referenzellipsoid

Um z.B. Längen- oder Flächenberechnungen auf der Erdoberfläche durchzuführen, ist es sinnvoll, die Gestalt der Erde durch eine sog. *Bezugsfläche* anzunähern. Dies ist eine geometrische Figur, welche mathematische Berechnungen im Vergleich zur unregelmäßigen physikalischen Erdfigur, dem *Geoid*, erheblich vereinfacht. Das in der Allgemeinheit bekannteste Beispiel dürfte mit hoher Wahrscheinlichkeit das Kugelmodell sein. Als Rechenfläche in der Geodäsie ist die Kugel als Erdfigur jedoch ungeeignet, da die Erde auf Grund der durch ihre Rotation bedingten wirkenden Fliehkräfte an den Polen eine Abplattung erfährt. Die daraus resultierenden Abweichungen betragen beim Kugelmodell ca. 10 km, diese Erdfigur ist somit nur für Karten sehr großen Maßstabs geeignet, wie etwa komplette Weltkarten [3]. Um die Genauigkeit der Messungen deutlich zu erhöhen, werden in der Geodäsie die sogenannten *Referenzellipsoide* verwendet. Hierbei handelt es sich um einen durch zwei Halbachsen a und b definierten Rotationsellipsoiden. In der Praxis wird der Referenzellipsoid durch seine *große Halbachse* a, der *Abplattung* f und dem sog. *Fundamentalpunkt* bestimmt. Der Fundamentalpunkt ist jener Punkt an dem der Referenzellipsoid das Geoid berührt. Durch die Zuordnung der astronomisch bestimmten geographischen Länge und Breite des Fundamentalpunktes

[2] nach F.R. Helmert
[3] vgl. http://de.wikipedia.org/wiki/Erdfigur

Name	große Halbachse a (m)	1/f
WGS1984 (IUGG)	6378137.000	298.257
WGS 1972 (IUGG)	6378135.000	298.260
Bessel 1841	6377397.155	299.153
Krassovsky 1940	6378245.000	298.300
Clarke 1880	6378249.145	293.465
International 1924 (Hayford 1909)	6378388.000	297.000

Tabelle 1: Erdellipsoide

zur entsprechenden ellipsoiden Länge und Breite sowie die parallele Ausrichtung der Rotationsachse des Ellipsoiden zur Erdachse wird letztendlich die Lagerung der Erdfigur bestimmt. Fundamentalpunkt und Referenzellipsoid zusammen werden als *geodätisches Datum* bezeichnet. Der für die deutsche Landesvermessung maßgebliche Fundamentalpunkt ist der *Trigonometrische Punkt (TP) Rauenberg* auf der Marienhöhe im Berliner Bezirk Tempelhof-Schöneberg[4]. Zusammen mit dem *Bessel-Ellipsoid* bildet er das in Deutschland verwendete Referenzsystem, das sog. *Potsdam Datum*. Tabelle 1 zeigt eine Auflistung verschiedener heute noch gebräuchlicher Erdellipsoide [5]

2.2 Kartenprojektionen

Zur Erstellung einer Karte muss die dreidimensionale Erdfigur in eine zweidimensionale Ebene projeziert werden. Dies bedeutet letztendlich, dass die die Erdfigur parametriesierenden Geographischen Koordinaten $(Laenge, Breite)$ in Kartesische Koordinaten $(X, Y$ bzw. $Rechts, Hoch)$ transformiert werden. Hierzu existieren unterschiedliche Projektionsverfahren, jedes dieser Verfahren hat Vor- und Nachteile und ist für einen bestimmten Kartentyp am geeignetsten. Grundsätzlich lassen sich die Kartenprojektionen in

- Zylinderprojektion
- Kegelprojektion
- Azimutale Projektion

unterteilen (vgl. Abb. 1). Die Abbildung in die Ebene erfolgt bei Zylinder- und Kegelprojektion durch „Abrollen“ des Projektionskörpers. Verwendet man als Erdfigur eine Kugel, so ist die Zylinderprojektion ein flächentreues Projektionsverfahren. Dabei kann die Abbildung der Erdfigur auf die Projektionsfläche auch durch speziell definierte Berechnungsvorschriften erfolgen, die Projektion ist dann eher mathematischer als geometrischer Natur. Man spricht in diesem Fall auch von

[4] http://de.wikipedia.org/wiki/Rauenberg_%28Trigonometrischer_Punkt%29
[5] http://www.olanis.de/knowhow/mapprj/mapprj2.shtml

einer *unechten Projektion*. Durch unechte Projektionen können Eigenschaften wie Mittelabstandstreue, d.h. die Abstände vom Kartenmittelpunkt werden unverzerrt wiedergegeben, und Flächentreue erreicht werden.

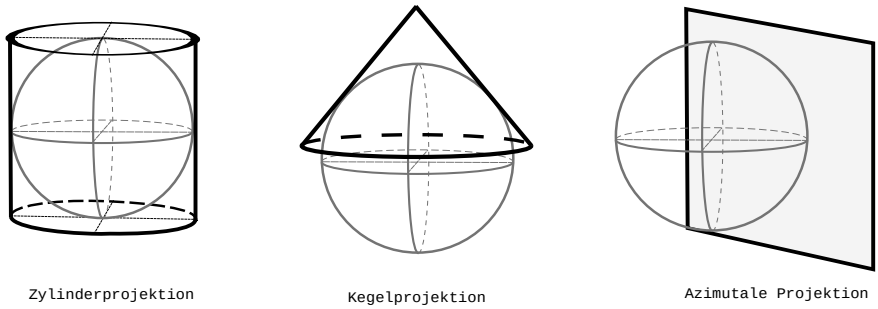


Abbildung 1: Projektionsarten

2.3 Koordinatensysteme

2.3.1 Geographische Koordinaten

Das geographische Koordinatensystem, welches einen Punkt auf der Erdoberfläche durch *Längen- und Breitengrad* beschreibt, dürfte das der Allgemeinheit bekannteste Koordinatensystem darstellen. Im Gradnetz der Erde führen die Längenkreise als Großkreise durch die beiden Pole, rechtwinklig dazu verlaufen die Breitenkreise. Dabei werden die Breitengrade vom Äquator aus nord- und südwärts gezählt, die beiden Pole liegen dementsprechend bei $90°N$ und $90°S$. Die Längengrade werden vom definierten Nullmeridian aus jeweils $180°$ in Ost- und Westrichting gezählt. Diese Aufteilung der Gradzahlen stimmt nicht mit der mathematischen Definition der Polarkoordinaten überein ($\theta \in [0, \pi], \phi \in [0, 2\pi]$). Zur Erhöhung der Präzision können die Gradangaben in *Bogenminute* und *Bogensekunde* unterteilt werden. Der Eingangsbereich der Universität Koblenz hat demnach die geographischen Koordinaten $Lat = 50°21'49,27''N, Lon = 7°33'32,88''E$[6]. Eine *Breitenminute* hat auf der Erdoberfläche eine Länge von $1\ NM$ (Nautical Mile, dt. Seemeile, entspricht $1852,01m$). Die Ausdehnung einer *Längenminute* auf der Erdoberfläche ist am Äquator gleich der Ausdehnung einer Breitenminute, nimmt jedoch in Abhängigkeit des Breitengrades ab und erreicht an den Polen die Länge Null. Innerhalb Europas beträgt die Länge der Strecke $1\ km$ bis $1,5\ km$[7].
Es bleibt hier festzuhalten, dass sich die Längen- und Breitengrade immer auf das zugrunde liegende Referenzellipsoid beziehen, insbesondere die Breitenangaben werden durch die Normalenrichtung des Punktes auf dem Ellipsoiden bestimmt,

[6]Quelle: Google Earth ™
[7]http://de.wikipedia.org/wiki/Abweitung

die Ellipsoidnormale verläuft daher nur im Falle der Kugel durch den Erdmittelpunkt (vgl. Abbildung 2).

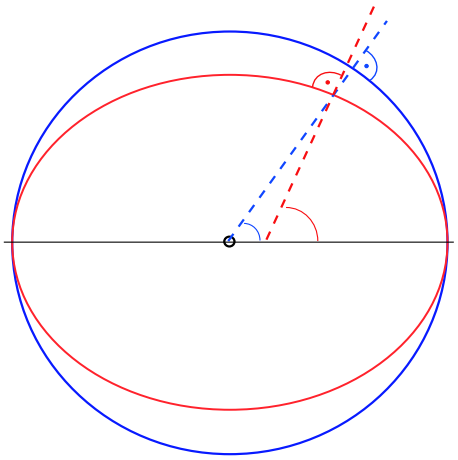


Abbildung 2: Breiten bei Ellipsoid und Kugel

2.3.2 Gauß-Krüger Koordinatensystem

Das von Carl Friederich Gauß und Johann Heinrich Louis Krüger entwickelte rechtwinklige Koordinatensystem basiert auf der sog. *transversalen* Mercatorprojektion, einer Zylinderprojektion, bei der die Zylinderachse in der Äquatorialebene der verwendeten Erdfigur liegt. Das Gauß-Krüger Koordinatensystem wird in Deutschland für die meisten topographischen Kartenwerke verwendet, als Referenzellipsoid wird hierbei auf den Bessel Ellipsoid zurückgegriffen. Das Gauß-Krüger Kordinatensystem stellt eine nach Gauß *konforme Abbildung* des Referenzellipsoides in die Ebene dar, dies bedeutet vereinfacht, dass die Verzerrungen bezüglich Länge, Fläche und Winkel für kleinste -im differenziellen Sinne- Teile des in die Ebene projezierten Ellipsoides sehr klein sind. Um dies zu erreichen, wird in dem von Gauß und Krüger entwickelten Projektionsverfahren nicht der komplette Referenzellipsoid auf einmal in die Ebene projeziert, sondern jeweils nur Streifen einer gewissen Breite (vgl. Abb. 3), die sog. *Meridianstreifen*. Dadurch werden die durch die transversale Zylinderprojektion verursachten Verzerrungen in „Ost-West“ Richtung in Grenzen gehalten. Um die gesamte Oberfläche des Referenzellipsoides abzubilden, wird der Projektionszylinder um die Breite jeweils eines Meridianstreifens „weitergedreht“. Im Gauß-Krüger Koordinatensystem werden in der Regel Meridianstreifen von $3°$ Breite verwendet. Dabei verläuft der *Mittelmeridian* eines Streifens durch die geographischen Längengrade $(0°, 3°, 6°, \ldots, 357°)$, die Gradzahl der Meridiane wird in östlicher Richtung aufsteigend gezählt.

2.3 Koordinatensysteme

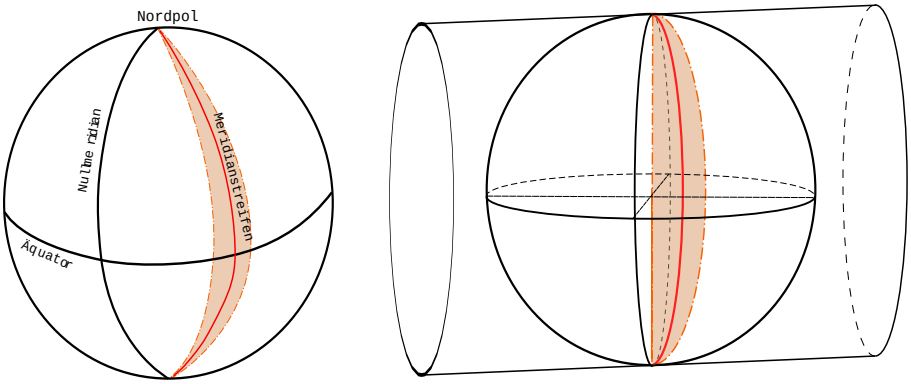


Abbildung 3: Gauss-Krüger Projektion

Somit stellt das Gauß-Krüger Koordinatensystem ein ebenes kartesisches Koordinatensystem dar, der Äquator entspricht hierbei der Abszisse, der Mittelmeridian der Ordinate. Ein Punkt innerhalb des Meridianstreifens lässt sich somit eindeutig durch ein Koordinatenpaar (x, y) bestimmen, im Gauß-Krüger System erhalten die Koordinaten die Bezeichnungen *Rechtswert* und *Hochwert*. Die Rechtswerte erhöhen sich in Richtung Ost, die Hochwerte in Richtung Nord. Der Rechtswert repräsentiert somit den Abstand eines Punktes zum Mittelmeridian, der Hochwert den Abstand eines Punktes zum Äquator entlang des längentreu abgebildeten Mittelmeridians. Die Abstände werden in Metern angegeben. Um negative Rechtswerte zu vermeiden, wird auf diese Koordinate ein Offset von $500000.0m$ addiert. Ferner wird dem Rechtswert noch die Kennziffer des entsprechenden Mittelmeridians vorangestellt, welche sich aus dem durch drei dividierten Wert der Gradzahl des Mittelmeridians berechnet. So lassen sich die in Abschnitt 2.3.1 angegebenen geographischen Koordinaten für den Eingangsbereich der Universitätsbibliothek durch die äquivalenten Gauß-Krüger Koordinaten $(3397597.0, 5579489.0)$ beschreiben. Hierbei gilt für den Rechtswert:

- Aus der Kennziffer des Mittelmeridians ($\underline{\mathbf{3}}397597.0$) ergibt sich dessen geographische Länge von $9°E$
- Der Punkt liegt $500000.0\ m - 397597.0\ m = 102403.0\ m$ westlich des Mittelmeridians
- Das Lot des Punktes auf den Mittelmeridian ist $5579489.0\ m$ vom Äquator entfernt

Für die Fläche der BRD sind vier Mittelmeridiane von Bedeutung: $6°, 9°, 12°, 15°$; somit wird Deutschland von drei Meridianstreifen überzogen. Die Grenzmeridiane werden dabei jeweils um 20 Bogenminuten erweitert, somit ergeben sich an den Streifengrenzen Überlappungszonen. Die vermessenden Behörden können

in diesen Gebieten entscheiden, welchem Meridianstreifen ein Punkt zugeordnet wird, dadurch können Umrechnungsprobleme vermieden werden. Abbildung 4 zeigt die Situation für Rheinland Pfalz: Die Fläche des Landes liegt in zwei Meridianstreifen, die Meridianstreifen mit der Kennziffer 2 und 3. Die in dieser Ar-

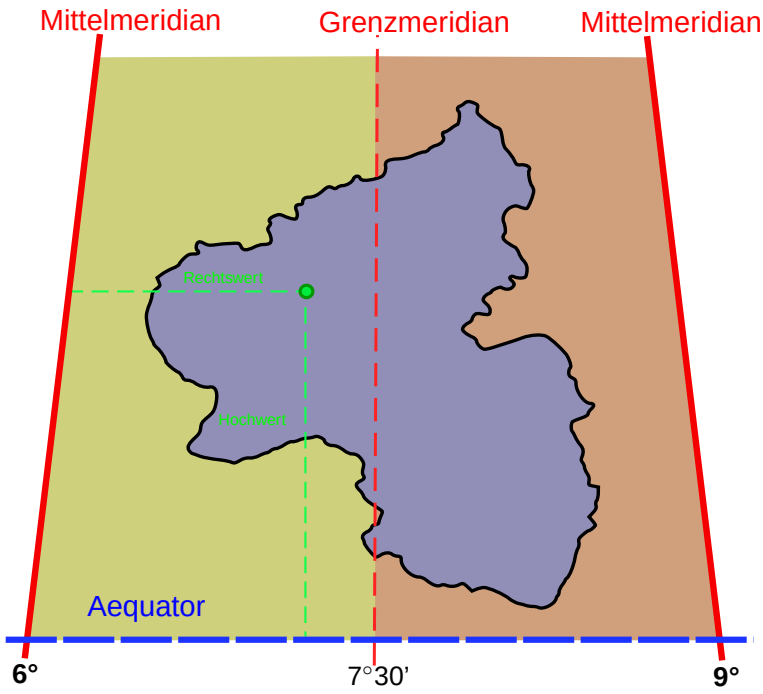


Abbildung 4: Gauss-Krüger Koordinatensystem

beit verwendeten Daten, insbesondere die digitalen topographischen Karten umfassten Koordinatenangaben im 2. und 3. Meridianstreifen. Damit ergibt sich das Problem, dass die Koordinatenursprünge der beiden Meridianstreifen verschieden sind. Da für die 3D-Visualisierung ein eindeutiges Weltkoordinatensystem notwendig ist, müssen die Daten auf einen Meridianstreifen umgerechnet werden. Aufgrund der Tatsache dass die in den verwendeten Datensätzen erfassten Gebiete hauptsächlich im dritten Meridianstreifen liegen, werden zur Vereinfachung und zur Minimierung von Verzerrungen, welche durch die Umrechnung entstehen, die Koordinaten vom zweiten in den dritten Meridianstreifen umgerechnet. Für die Umrechnung wurde das Werkzeug **Geographic Translator (GEOTRANS)**[8] verwendet. Hierbei handelt es sich um eine umfangreiche Programmbibliothek für die verschiedensten geographischen Berechnungen. Im Paket enthalten sind ein Anwendungsprogramm mit graphischer Benutzeroberfläche sowie der komplette C-Quellcode.

[8] http://earth-info.nima.mil/GandG/geotrans/

2.3.3 Rendering-Koordinatensystem

Für das Rendering der Landschaft ist es notwendig, dass das Landschaftsmodell durch Vertices in einem dreidimensionalen rechtwinkligen Koordinatensystem, dem sog. *Weltkoordinatensystem*, repräsentiert wird. Dazu müssen die in den Daten enthaltenen Gauss-Krüger Koordinaten in dieses Weltkoordinatensystem umgerechnet werden. In dieser Arbeit repräsentiert die X- Achse des Weltkoordinatensystems die Abszisse des Gauss-Krüger Koordinatensystems, die Z- Achse des Weltkoordinatensystems entspricht der „invertierten“ Ordinate des Gauss-Krüger Koordinatensystems, d.h. die Z-Werte steigen in südlicher Richtung. Die Höhe eines Punktes über NN.[9] wird durch die Y-Achse des Weltkoordinatensystems repräsentiert. Diese Umrechnung des linkshändigen Gauss-Krüger Koordinatensystems in das rechtshändige Weltkoordinatensystem wurde durch die Verwendung des OpenGL Graphik API zur Datenvisualisierung notwendig. Der Ursprung des Weltkoordinatensystems kann bezüglich Gauss-Krüger Koordinaten beliebig gewählt werden, zur Umrechnung von Gauss-Krüger Koordinaten aus verschiedenen Meridianstreifen in das Weltkoordinatensystem müssen diese Gauss-Krüger Koordinaten zuerst in den Meridianstreifen, welcher den Ursprung des Weltkoordinatensystems enthält, umgerechnet werden.

[9] **N**ormal **N**ull, über dem Meeresspiegel

3 Datenquellen

Das in dieser Arbeit entwickelte dreidimensionale digitale Landschaftsmodell wird im wesentlichen auf der Grundlage digitaler Landesvermessungsdaten erstellt. Das Datenmaterial umfasst:

- ATKIS Basis DLM - Eine digitale topographische Karte
- DGM - Ein digitales Höhenmodell
- Digitale Orthophotos - Luftbildaufnahmen

3.1 Digitales Geländemodell

Das digitale Geländemodell (DGM) beinhaltet Höhenangaben von Messpunkten im Gelände. Die Messpunkte verteilen sich hierbei in einem regelmäßigen Gitter über die Landschaft. Die Hauptachsen des Gitternetzes verlaufen in Richtung Nord-Süd bzw. Ost-West, der Abstand zwischen den Messpunkten beträgt $10.0\ m$, sowohl in Nord-Süd als auch in Ost-West Richtung. Die Daten sind in einer Textdatei (`uni_ko_la.xyz`) zeilenweise im Format *Rechtswert Hochwert Höhe ü. NN* gespeichert. Abbildung 5 zeigt einen Auszug von drei aufeinander folgenden Zeilen aus der Datei. Deutlich zu sehen ist, dass die Abtastpunkte auf jeweils ganz-

```
  ⋮
3392030.000   5588000.000   65.645
3392040.000   5588000.000   65.742
3392050.000   5588000.000   65.754
  ⋮
```

Abbildung 5: DGM Datei (Auszug)

zahlige Rechts- und Hochwerte fallen. In Richtung Nord-Süd wurde das Gelände mit 3401 Messpunkten abgetastet, in Richtung Ost-West sind es 2001 Messpunkte, es liegen daher Höhendaten für eine Fläche von $34\ km \times 20\ km$ vor. Die Datei hat eine Größe von ca. 250 MB. Auf Grund der Anordnung der Messpunkte in einem regelmäßigen Gitter sind die Rechts- und Hochwerte in jedem Messpunkt implizit enthalten. Für die Georeferenzierung genügt es daher, die Rechts- und Hochwerte des „ersten“ Messpunktes sowie die Auflösung des Gitters in Nord-Süd und Ost-West Richtung zu speichern. Dadurch reduziert sich die Datenmenge der Textdatei auf ca. 50 MB. Speichert man die Daten als Binärdatei mit einer Genauigkeit von 32 Bit pro Messpunkt, so fallen nur noch ca. 26 MB Daten an. Eine Reduktion der Genauigkeit auf 16 Bit sollte kein Problem darstellen, selbst bei einem angenommenen Höhenbereich von 0 m bis 1000 m über NN. würden die Messpunkte auf

ca. 1,5 cm genau dargestellt. Somit würde sich die Größe der Datei nochmals auf ca. 13 MB verringern lassen.

3.1.1 Interpolation der Höhenwerte

Um den Höhenwert eines beliebigen Punktes p im DGM zu erhalten, muss zwischen den diskreten Abtastpunkten des DGMs interpoliert werden. Da die Abtastpunkte in Form eines regelmäßigen Gitters angeordnet sind, kann auf das Verfahren der *bilinearen Interpolation* zurückgegriffen werden. Abbildung 6 veranschaulicht die Vorgehensweise. Interpoliert wird die Höhe p^y des Punktes p aus den

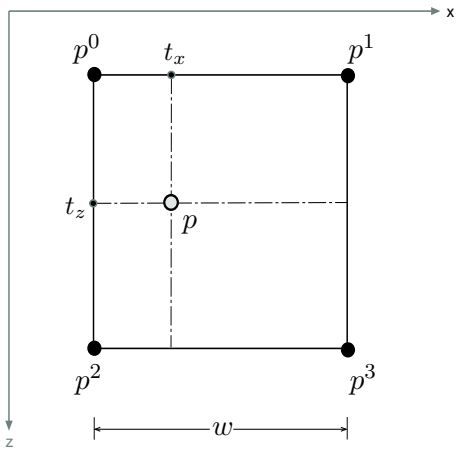


Abbildung 6: Bilineare Interpolation der Eckpunkte

gegebenen Höhenwerten der umgebenden Gittepunkte p_0, p_1, p_2, p_3. Um die Interpolationsfaktoren t_x und t_z zu berechnen, werden die x- und z-Koordinaten des Gitterpunktes p_0 sowie die Gitterweite w benötigt. Da die Gitterpunkte jeweils auf ganzzahligen x- und z-Koordinaten liegen ergibt, sich unter Zuhilfenahme der *floor* Funktion ($\lfloor\rfloor$) somit für p_0^x und p_0^z

$$\begin{aligned} p_0^x &= \left(\lfloor\frac{p^x}{w}\right) \cdot w \\ p_0^z &= \left(\lfloor\frac{p^z}{w}\right) \cdot w \end{aligned}$$

Somit lassen sich die Interpolationsfaktoren durch

$$\begin{aligned} t_x &= (p^x - p_0^x) \cdot \frac{1}{w} \\ t_z &= (p_z - p_0^z) \cdot \frac{1}{w} \end{aligned}$$

berechnen. Für die bilineare Interpolation werden nun im ersten Schritt die y-Werte der Gitterpunkte p_0 und p_1 bzw. die y-Werte der Gitterpunkte p_2 und p_3 mittels

$$\begin{aligned} p_{01}^y &= p_0^y + t_x \cdot (p_1^y - p_0^y) \\ p_{23}^y &= p_2^y + t_x \cdot (p_3^y - p_2^y) \end{aligned}$$

interpoliert. Aus den sich ergebenden Zwischenwerten p_{01}^y und p_{23}^y kann nun durch

$$p^y = p_{01}^y + t_z \cdot (p_{23}^y - p_{01}^y)$$

der gesuchte Höhenwert p^y errechnet werden.

3.2 ATKIS Basis DLM

Das ATKIS (*Amtliches Topographisch-Kartographische Informationssystem*) erfasst topographische Objekte einer Landschaft in Form eines *Digitalen Landschaftsmodelles (DLM)*. Grundlage für die Erfassung bilden die *Deutsche Grundkarte*, digitale *Orthofotos* (Maßstab 1:5000), sowie Informationen des Topographischen Informations-Managements (TIM).
Die erfassten Objekte sowie die Erfassungskriterien und das Regelwerk zur Objektbildung sind sehr ausführlich im *ATKIS Objektartenkatalog (ATKIS-OK)* [AdV03] beschrieben. Für jedes Objekt werden sowohl topographische als auch inhaltliche Informationen erfasst. Die topographische Information beschreibt hierbei die vereinfachte geometrische Form der Landschaftsobjekte. Die vorkommenden Geometrien werden in Punkte, Linien und Flächen unterteilt. Ferner existieren noch Raster- und komplexe Objekte. Das topologische Netz der Verkehrswege wie Straßen und Bahnlinien, Fluss- und Bachläufe und ähnliche Landschaftsobjekte teilen die Erdoberfläche gewissermaßen in „Parzellen“ auf, welche dann mit flächenförmigen Objekten gefüllt werden. Diejenigen Objekte, welche die „Grenzlinien“ der Flächen bilden, werden ihrerseits durch linienförmige Geometrien repräsentiert. Die Linie verläuft dann mittig des eingrenzenden Objektes, bei Straßen wird hierbei die Mittellinie des Straßenkörpers als Grundlage für die Modellierung verwendet. Masten, Schornsteine, Türme etc. können als punktförmige Objekte beschrieben werden. Komplexe Objekte besitzen keine eigene topographische Beschreibung, sondern setzen sich aus verschiedenen Objekten zusammen. Somit lassen sich Objekte modellieren, welche nach den im ATKIS-OK festgelegten Objektbildungsregeln getrennt modelliert werden müssen. So kann z.B. eine Eisenbahnbrücke als ein komplexes Schienenbahnobjekt modelliert werden, welches dann aus den Objekten „Schienenbahn“ und „Brücke“ besteht. Bei rasterförmigen Objekten werden die geometrischen Daten in Form einer $(m \times n)$ Matrix gespeichert. Dabei können in der Matrix die Datentypen *„Integer“*, *„Real“* und

„String“ gespeichert werden. Rasterobjekte kamen in den Daten nicht vor, daher wurde auf eine weitere Behandlung in dieser Arbeit verzichtet.
Inhaltlich werden die topographischen Objekte durch ihre Funktion klassifiziert. Funktional gleichartige topographische Objekte werden einer sog. *Objektart* zugeordnet. Die verschiedenen Objektarten sind nach sachlogischen Gesichtspunkten hierarchisch strukturiert, die Kodierung im ATKIS-DLM wird durch eine vierstellige Ganzzahl realisiert. Eine grundlegende Unterteilung der Objektart ist durch die *Objekbereiche* beschrieben. Tabelle 2 zeigt deren verschiedenen Ausprägungen. Die einzelnen Objektbereiche erfahren dann eine feinere Unterteilung in *Objekt-*

Nr.	Objektbereich
1000	Präsentation
2000	Siedlung
3000	Verkehr
4000	Vegetation
5000	Gewässer
6000	Relief
7000	Gebiete

Tabelle 2: Objektbereiche

gruppen, Tabelle 3 zeigt hier exemplarisch die Aufteilung des Objektbereichs *„Verkehr“* in die einzelnen Objektgruppen. Die Objektgruppen werden ihrerseits noch-

Nr.	Objektgruppe
3100	Straßenverkehr
3200	Schienenverkehr
3300	Flugverkehr
3400	Schiffsverkehr
3500	Anlagen und Bauwerke für Verkehr, Transport und Kommunikation

Tabelle 3: Objektbereich *„Verkehr“*

mals in *Objektarten* kategorisiert, welche dann die konkrete Ausprägung eines topographischen Objektes beschreibt. Tabelle 4 zeigt die Aufgliederung der Objektgruppe *„Schienenverkehr“* in ihre einzelnen Objektarten.
Eine feinere Klassifizierung der einzelnen Objektarten wird durch zusätzliche *Attribute* erreicht. Attribute ermöglichen eine genauere Beschreibung des modellierten Objektes. Jedes Attribut besitzt hierbei einen *Attributnamen*, durch drei Buchstaben codiert, und einen durch eine vierstellige Ganzzahl codierten *Attributwert*. Tabelle 5 zeigt beispielhaft das Attribut *„Bahnkatekorie“ (BKT)“* der Objektart 3201

Nr.	Objektart
3201	Schienenbahn
3202	Seilbahn, Schwebebahn
3203	Bahnkörper
3203	Bahnstrecke

Tabelle 4: Objektgruppe *„Schienenverkehr“*

(Schienenbahn). Die Objekte werden durch das ATKIS *DLM-Datenmodell* (vgl. [Ver93,

BKT Bahnkatekorie	
1100	Eisenbahn
1102	Güterzugbahn
1104	S-Bahn
1200	Stadtbahn
1201	Straßenbahn
1202	U-Bahn
1300	Bergbahn
1301	Zahnradbahn
1302	Standseilbahn
1400	Museumsbahn
1500	Bahn im Freizeitpark
1600	Magnetschwebebahn

Tabelle 5: Attribute für Objektart *3201*

Band 2, Punkt 1.2.1ff],) beschrieben. Ein ATKIS DLM-Objekt ist hierbei als geometrisch abgrenzbare konkrete Instanz einer Objektart zu sehen. Es besteht aus einem oder mehreren Objektteilen, welche jeweils eine punkt-, linien-, flächen- oder rasterförmige Repräsentation haben können. Eine eindeutige Auszeichnung eines DLM-Objektes erfolgt über eine aus sieben alphanumerischen Zeichen bestehende *Objek-ID*.
Die Bildung von Objektteilen erfolgt nach in [AdV03, Teil D0, S. 6] festgelegten Regeln. Ein Objekt wird z.B. bei der Änderung eines Attributes in mehrere Objektteile aufgegliedert. Dies kann beispielsweise bei einer Fahrbahnverengung einer Straße vorkommen, hier würde sich das Attribut „BRF“ (Breite der Fahrbahn) ändern. Einem Objektteil sind dann die eigentlichen geometrischen Informationen in Form sog. *Vektorelemente* zugeordnet. Eindeutig identifiziert wird der Objektteil über die aus zehn alphanumerischen Zeichen bestehende *Objektteil-ID*. Diese setzt sich aus der Objekt-ID und einer zusätzlichen dreistelligen Ziffernfolge zusammen, welche die Objektteile durchnummeriert.

Ein Vektorelement beschreibt einen Linienzug, welcher mindestens aus Startpunkt und Endpunkt besteht und mehrere Zwischenpunkte beinhalten kann. Abbildung 7 zeig ein Vektorelement, welches aus dem Startpunkt S, den Zwischenpunkten z_0, z_1, z_2 sowie dem Endpunkt E aufgebaut ist. Im ATKIS wird die Positionsanga-

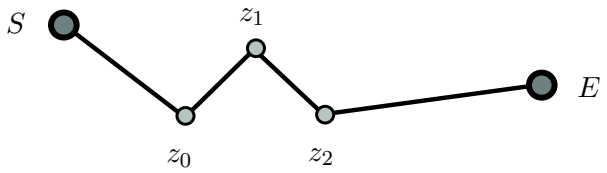


Abbildung 7: Vektorelement

be durch Gauss-Krüger Koordinaten im Bessel Ellipsoid beschrieben, eine Angabe über die tatsächliche Höhe eines Punktes über NN. ist in den ATKIS Basis-DLM Daten nicht enthalten.

3.3 ATKIS und DGM Daten im Kontext von 3D Rendering

3.3.1 Höheninformation

Die durch ATKIS modellierten Vermessungsdaten werden primär zur Erzeugung von zweidimensionalen topographischen Karten verwendet. Das ATKIS Datenmodell, insbesondere die Beschreibung der Objektgeometrien kann nicht direkt als Datengrundlage für ein Echtzeit-Rendering System verwendet werden. Zunächst müssen die zweidimensionalen ATKIS Daten um eine dritte Dimension, der Höhe über NN., erweitert werden. Dies kann durch die Verwendung eines DGMs und Interpolation der Höhenwerte aus dem DGM erreicht werden (vgl. 3.1). Die Qualität der Ergebnisse hängt damit eng mit der Auflösung des DGMs zusammen. In den zur Verfügung gestellten Daten beträgt die Auflösung des DGMs $10\ m$ in beiden Abtastrichtungen. Durch diese relativ grobe Abtastung des Geländes ergeben sich zwangsläufig Artefakte. Abbildung 8 zeigt beispielhaft hierfür den Verlauf einer Uferlinie im Gelände. Die Wasserfläche wird hier auf Grund der Abtastrate nicht komplett auf eine Ebene abgebildet, was in der Abbildung deutlich zu erkennen ist. Diese Artefakte würden durch eine höhere Abatstrate der Höheninformation reduziert werden, die höhere Präzision wird jedoch mit einem größeren Speicherbedarf der DGM-Daten erkauft. Eine Glättung der Höhendaten würde keinen wesentlichen Vorteil mit sich bringen, da diese Operationen auch mit der gegebenen Abtastrate durchgeführt werden müssten.
Ferner schränkt die Repräsentation der Höhendaten als sog. *Height Field* die Darstellbarkeit möglicher Geländeformen ein. Durch die beschränkte Abtastfrequenz ergibt sich nach dem Nyquist-Shannonschen Abtasttheorem eine *Unterabtastung*

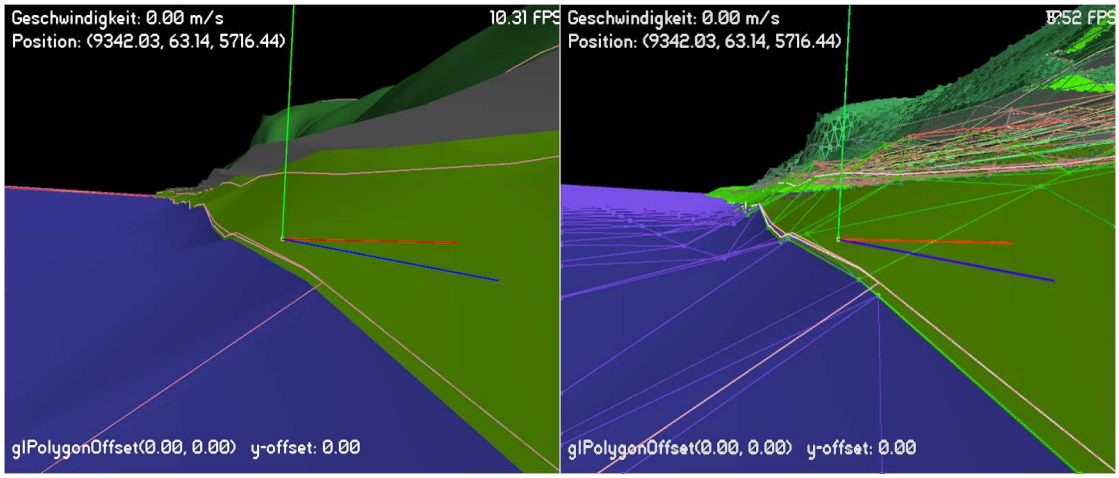


Abbildung 8: Auflösung DGM

des Geländeprofils. Als Konsequenz hieraus ergibt sich letztendlich, dass jegliche Höheninformation zwischen den Abtastpunkten verloren geht. Weiterhin können durch Height Fields keine Überhänge oder zur Bodenebene senkrecht verlaufende Kanten dargestellt werden (vgl. Abbildung 9). Dies ist ein generelles Problem, wel-

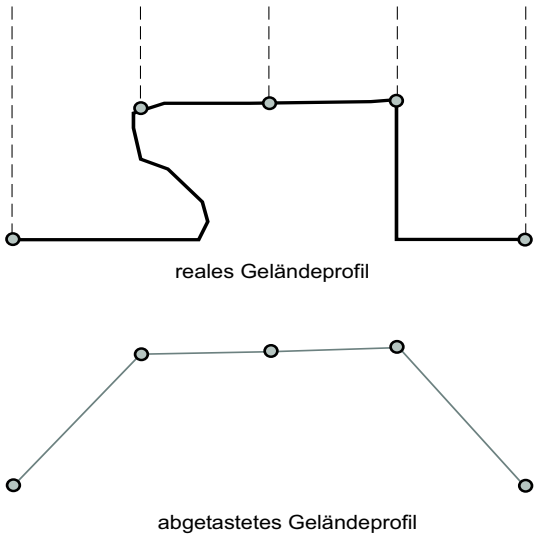


Abbildung 9: Abtastung des Höhenprofils

ches bei der Verwendung von Height Fields als Höheninformation auftritt. Durch eine „mehrschichtige“ Abtastung des Geländes ließen sich auch Überhänge und senkrechte Wände erfassen, jedoch ist eine Vermessung ganzer Landschaften auf diese Art -wenn überhaupt- nur mit sehr großem Aufwand denkbar. Weiterhin ist

die Erzeugung von Geometriedaten aus einem mehrschichtigen Geländemodell wesentlich aufwändiger, als dies bei Height Fields der Fall ist. Da in dieser Arbeit ausschließlich Datensätze in Form eines Height Fields zur Verfügung standen, konnte in diesem Punkt letztendlich kein alternativer Weg verfolgt werden.

3.3.2 Anordnung der Vektorelemente

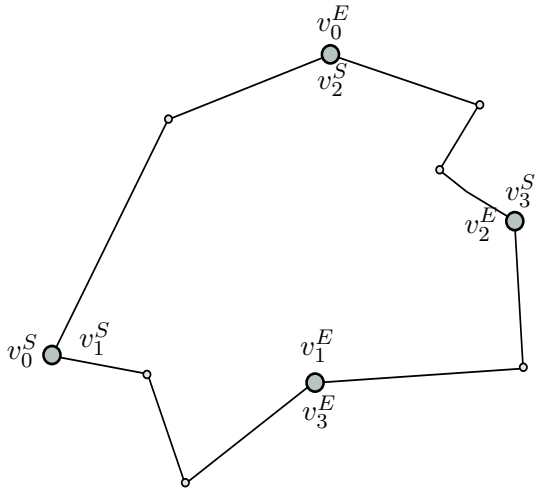


Abbildung 10: Anordnung von Vektorelementen

Ein weiteres Problem stellt die Anordnung der die Geometrie beschreibenden Vektorelemente, d.h. der Liniensegmente, dar. So sollte eine durch ein Polygon beschriebene Fläche idealerweise durch eine im Uhr- oder Gegenuhrzeigersinn geordnete Liste von Punkten beschrieben werden. Dies ist im ATKIS nicht zwingend vorgeschrieben. Abbildung 10 zeigt die Problematik an Hand eines flächenförmigen Objektes, welches durch eine Liste von Vektorelementen (v_0, v_1, v_2, v_3) beschrieben wird. Zur Vereinfachung wurden lediglich die Start- und Endpunkte der Vektorelemente eingezeichnet, im Schaubild mit v_i^S bzw. mit v_i^E bezeichnet. Da in jedem Vektorelement jeweils dessen Start- und Endpunkt gespeichert ist, kommen in einem durch Vektorelemente beschriebenen Objekt zwangsläufig Punkte doppelt vor, da bei geschlossenen Linienzügen jeder Punkt gleichzeitig als Start- und Endpunkt von zwei verschiedenen Vektorelementen fungiert und somit auch doppelt gespeichert wird. Bei offenen Linienzügen, d.h. bei linienförmigen Objekten, trifft dies auf alle Punke mit Ausnahme des Anfangs- und Endpunktes des Linienzuges zu. Weiterhin ist die Reihenfolge der Vektorelemente keinem festen Regelwerk unterworfen, im Schaubild fallen beispielsweise die Startpunkte der Vektorelemente v_0 und v_1 auf den gleichen Punkt. Demzufolge ist der Startpunkt von Vektorelement v_1 nicht der Endpunkt von Vektorelement v_0, wie es bei

einer Anordnung der Punkte als Linienzug der Fall wäre. Die Liste von Vektorelementen muss daher in die geforderte Form, als Liste von geordneten Eckpunkten, umgewandelt werden. Im Beispiel hätte eine mögliche Ausprägung der Eckpunktliste die Form $\left(v_0^S, v_1^E, v_2^E, v_2^S\right)$, zur Vereinfachung werden auch hier nur die Start und Endpunkte betrachtet. Der Algorithmus, welcher die Umformung realisiert, fügt in einem initialen Schritt den Startpunkt, die Zwischenpunkte und den Endpunkt des ersten Vektorelementes der Liste in die Eckpunktliste ein. Danach wird dieses Vektorelement gelöscht. Nun wird aus den verbleibenden Vektorelementen dasjenige, welches den Start- oder Endpunkt mit dem zuletzt bearbeiteten Vektorelement gemeinsam hat, herausgesucht. Ist der gemeinsame Punkt der Startpunkt des gefundenen Vektorelements, so werden dessen Zwischenpunkte und dessen Endpunkt in die Eckpunktliste eingefügt. Ist der gemeinsame Punkt der Endpunkt des gefundenen Vektorelements, so werden die Zwischenpunkte in umgekehrter Reihenfolge, gefolgt vom Startpunkt in die Eckpunktliste eingefügt. Danach wird das Vektorelement gelöscht. Dies wird iterativ so lange fortgesetzt, bis alle Vektorelemente verarbeitet wurden. Lisitng 1 zeigt den Ablauf des Algorithmuses in Pseudo-Code.

Listing 1: Vektorelemente in Eckpunktliste konvertieren

```
1  Eingabe: Liste von Vektorelementen vecElList
2  Ausgabe: Liste von Eckpunkten vertexList
3
4  vecElList_0.startPunkt in vertexList einfügen;
5  vecElList_0.zwischenPunkte in vertexList einfügen;
6  Vertex letzter = vecElList_0.endPunkt;
7
8  lösche vecElList_0 aus vecElList
9  WHILE(vecElList ∉ ∅){
10     FOR EACH(vEl ∈ vecElList){
11         IF(vEl.startPunkt == letzter){
12             vEl.zwischenPunkte in vertexList einfügen;
13             vEl.endPunkt in vertexList einfügen;
14             letzter = vEl.endPunkt;
15             break;
16         }
17         IF(vEl.endPunkt == letzter){
18             vEl.zwischenPunkte in vertexList umgekehrt einfügen;
19             vEl.startPunkt in vertexList einfügen;
20             letzter = vEl.startPunkt;
21             break;
22         }
```

```
23      }
24      lösche vEl aus vecElList
25  }
```

Ein ähnliches Problem wie bei der Anordnung der Vektorelemente zeigt sich, wenn ein Objekt aus mehreren Objektteilen besteht, d.h. wenn zum Beispiel eine Straße aus mehreren linienförmigen Objektteilen aufgebaut ist. Die einzelnen Objektteile können auch hier in ungeordneter Form vorkommen.

3.3.3 Flächenförmige Geometrie

Für die Visualisierung von flächenförmigen Objekten mittels eines 3D Renderingsystems ergeben sich eine Vielzahl von Problemstellungen. Das sicherlich aufwändigste Problem ergibt sich aus der Tatsache, dass eine Fläche lediglich durch ihre Umrisslinie modelliert wird. Somit ist eine *Triangulation* der Fläche vor ihrer Verwendung im Renderingsystem unumgänglich. Besonders erschwerend ist hierbei die große Zahl an möglichen Formen der Umrisslinie, wobei insbesondere das Vorkommen von konkaven Umrisslinien zur Problematik beiträgt. Ferner erlaubt der ATKIS Objektartenkatalog das Vorkommen von Flächen mit Aussparungen, wobei dies laut OK eher selten vorkommt und geschlossene Flächen den Regelfall darstellen. Weiterhin sind flächenförmige Objekte, welche aus mehreren geschlossenen Flächen ohne Punktidentität bestehen, sowie flächenförmige Objekte, welche aus Flächen mit Punktidentität bestehen erlaubt. Flächen mit Punktidentitäten resultieren dann in nicht-einfachen Polygonen. Im Verlauf der Entwicklung des Renderingsystems hat sich gezeigt dass Flächen mit Aussparungen in den Datensätzen nicht vorkommen. Somit konnte auf eine programmtechnische Umsetzung dieses Teilaspektes zugunsten anderer Problemstellungen verzichtet werden, ohne einen Verlust an visueller Qualität hinnehmen zu müssen.
Bei der Abbildung der Erdoberfläche in flächenförmige Geometrien können im ATKIS Redundanzen auftreten, d.h. es kommt zur Überlagerung von zwei oder mehreren Flächen. Im ATKIS Objektartenkatalog steht hierzu: „Auf Grund der Vielfalt der Erscheinungsformen der Landschaft ist die Erdoberfläche nicht immer redundanzfrei abzubilden. Deshalb dürfen die folgenden Grundflächen andere Grundflächen überlagern: *Sportanlage / Tagebau, Grube, Steinbruch / Halde, Aufschüttung / Absetzbecken, Schlammteich, Erdfaulbecken / Platz / Hafenbecken / Grünland / Wald, Forst / Gehölz / Binnensee, Stausee, Teich*. Die Objekte der Objektart *Ortslage, Hafen, Schleuse,* sowie des Objektbereiches *Gebiete* können alle Objektarten überlagern.“[10] Flächenförmige Objekte, welche aus mehreren Einzelflächen bestehen, müssen im ATKIS stets durch mehrere Objektteile repräsentiert werden. Da

[10] vgl [AdV03], Teil D0, S.3

die einzelnen Objektteile jeweils einzelne geschlossene Flächen darstellen, erhält man durch eine Weiterverarbeitung der Daten auf Objektteilebene stets die gewünschten geschlossenen, einfachen Polygone. Eine Verarbeitung der Daten auf Objektteilebene macht auch unter Berücksichtigung der bereits erwähnten nichtgeordneten Reihenfolge der Objektteile in den Objekten Sinn. In Kapitel 4.5 wird nochmals ausführlich auf die Triangulierung der Daten eingegangen.

3.3.4 Linienförmige Geometrie

Die Repräsentation linienförmiger Objekte im ATKIS ist für das 3D Rendering wesentlich günstiger, da hier nur Linienzüge mit genau zwei Endpunkten vorkommen dürfen. Nicht erlaubt sind:

- Zwei seperate Linienzüge innerhalb eines Objektes
- Sich kreuzende Linienzüge
- Verzweigungen
- Linienzug mit gleichem Anfangs- und Endpunkt
- Linienzug, der auf der eigenen Geometrie endet

3.4 Das EDBS Format

Das Datenformat *EDBS (einheitliche Datenbankschnittstelle)* dient als standardisiertes Austauschformat für Daten der ALK und des ATKIS. Die offizielle Dokumentation des Formates befindet sich in [Ver93], Anhang A. Mit dem EDBS Format werden die punkt- und vektororientierten Daten des ATKIS-Datenmodells in einer textbasierten Struktur beschrieben. Die einzelnen Datensätze sind hierbei zeilenweise als ASCII-Text gespeichert. Das EDBS Format selbst, sowie die Regeln zur Transformation des objektorientierten ATKIS-Datenmodells in das hierarchisch organisierte EDBS Datenmodell sind in [Ver93], Band 2 beschrieben.

3.4.1 Aufbau und Organisation

Die grundlegende Eigenschaft des EDBS Datenformates ist es, die Geometrieinformation losgelöst vom dazugehörigen Objekt zu speichern. Dies dient dem Zweck der redundanzfreien Datenhaltung der Geometriedaten. Als maßgebliches Geometrieprimitiv dient hierbei der Linienzug. Zu jedem Linienzug werden die Geometriedaten durch den Startpunkt, beliebig viele Zwischenpunkte und den Linienendpunkt gespeichert. Hier findet eine Abbildung der ATKIS-Vektorelemente (vgl. Abschnitt 3.2) in die EDBS Datenstruktur statt. Zu jedem Linienzug werden neben den eigentlichen Geometriedaten Referenzen auf diejenigen Objekte, welche diesen Linienzug beinhalten, gespeichert. Ein Linienzug kann demnach ein

Teil von Umrisslinien zweier benachbarter Flächen und gleichzeitig die Mittellinie eines linienförmigen Objektes sein (vgl. Abbildung 11). Im folgenden soll nun

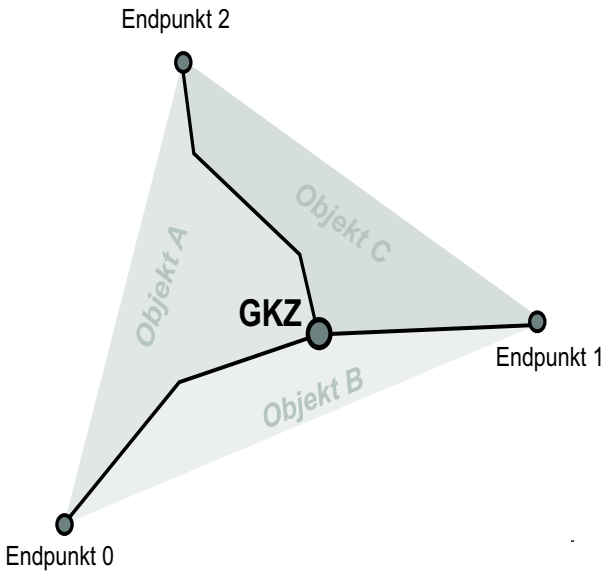
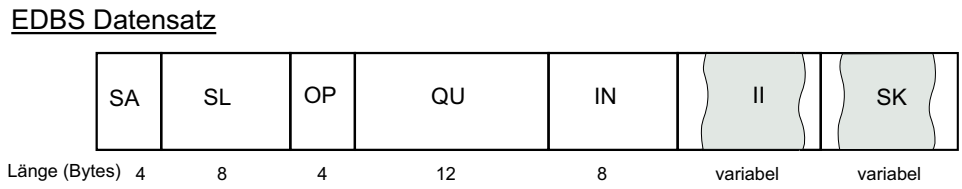


Abbildung 11: Linienzüge im EDBS-Format

genauer auf die einzelnen Aspekte des EDBS Formates und die Abbildung des ATKIS-Datenmodells in EDBS eingegangen werden.
Ein EDBS-Satz stellt die atomare Informationseinheit des EDBS-Formates dar. Er erstreckt sich über genau eine Zeile der Textdatei. Die maximale Länge einer Zeile beträgt hierbei 2000 Zeichen, das Ende einer Zeile wird durch ein Zeilenvorschubzeichen markiert (Windows: CR LF, Unix: LF). Ist die darzustellende Information größer als die maximale Zeilenlänge, so wird diese auf mehrere aufeinanderfolgende Datensätze verteilt. Der allgemeine Aufbau eines EDBS Datensatzes gestaltet sich wie folgt:



Parameter	Bedeutung
SA	Anfang des EDBS Satzes
SL	Länge des EDBS Satzes und Anfangsadresse des Suchkriteriums
OP	Operationsschlüssel
QU	Quittungs und Editierschlüssel
IN	Name der Information
II	Inhalt der Information
SK	Suchkriterium

Der Parameter SA ist immer vorhanden und stets mit dem Wert „EDBS“ belegt. Die eigentliche Information wird in den Parametern „IN“ und „II“ übermittelt, daher sei für eine genaue Erläuterung der übrigen Parameter an dieser Stelle auf die Dokumentation der AdV ([Ver93], Anhang A) verwiesen. Eine komplette Datenübermittlung aus einem entsprechenden Informationssystem wird als *EDBS-Auftrag* bezeichnet. Ein EDBS-Auftrag besteht aus:

- *Auftragskennsatz*
- Weiteren EDBS-Sätzen (welche die angeforderten Daten enthalten)
- *Auftragsendsatz*

Der Auftragskennsatz ist hierbei der erste Datensatz eines EDBS Auftrages. Er enthält verschiedene Steuer- und Kenndaten, welche jedoch für diese Arbeit nicht relevant waren. Der Auftragsendsatz enthält keine Daten und dient ausschließlich zur Markierung des Auftragsendes. Die weiteren EDBS-Sätze bestehen ihrerseits aus:

- Gebietskennzeichnung
- Grundrissdaten
- Attributdaten

Ein Grundrissdatensatz wird in der EDBS-Dokumentation als *Grundrissdatei* bezeichnet, ein Attributdatensatz analog dazu als *Attributdatei*. Tatsächlich handelt es sich hierbei um eine einzelne Zeile der Textdatei. Die Grundrissdaten stellen die Abbildung der im ATKIS definierten Objekte in das EDBS Format dar, sie können sowohl die fachliche als auch die geometrische Information beinhalten. Die Attributdaten beinhalten die in Abschnitt 3.2 aufgeführten Attribute. Die Gebietskennzeichnung beinhaltet die Information über das im EDBS Auftrag umfassende

Gebiet. Es werden hierbei die Rechts- und Hochwerte der nord-westlichsten und süd-östlichsten Koordinaten angegeben.

3.4.2 Die Parameter IN und II

Die Parameter IN und II in den Attribut- und Grundrissdateien beschreiben die eigentliche topographische Information, daher kommt ihnen besondere Bedeutung zu.
Der Parameter IN, der Name der Information, kennzeichnet letztendlich die Art der Information, welche der Parameter II enthält. Im Falle der Attribut- und Grundrissdateien lauten die Informationsnamen „ULOBNN␣␣“ für eine Grundrissdatei und „ULTANN␣␣“ für eine Attributdatei. Da für den Parameter IN 8 Bytes vorgesehen sind, werden Informationsnamen mit weniger als 8 Zeichen mit Leerzeichen aufgefüllt.
Der Inhalt der Information, in [Ver93] als *Dateneinheit* bezeichnet, ist in sog. *Datengruppen* untergliedert. Dabei kann eine Datengruppe beliebig oft, mindestens einmal, oder gar nicht in der Dateneinheit vorkommen. Ferner kann eine einzelne Datengruppe ihrerseits wieder aus Datengruppen bestehen, somit kann eine hierarchische Anordnung der Information erreicht werden. In der ATKIS Dokumentation wird die Datengruppe zuweilen auch als *Standardaggregat* bezeichnet. Eine Datengruppe umfasst eine Menge von Datenfeldern, welche die eigentlichen Informationsträger darstellen. Die Datenfelder werden als *Datenaggregate* bezeichnet. Im EDBS-Format wird die Häufigkeit des Vorkommens einer Datengruppe durch den *Wiederholfaktor (WHF)* angegeben, bei dem es sich um eine vierstellige Ganzzahl handelt. In Abbildung 12 werden die Datengruppen der Grundrissdatei und deren hierarchische Beziehung zueinander als UML-Klassendiagramm dargestellt. In diese Datenstruktur werden letztendlich die ATKIS-Objekte und die dazugehörigen ATKIS-Objektteile überführt. Dabei werden die topographischen Informationen wie Objektart, Objekttyp, Objektnummer etc. in den Datengruppen „Funktion des Objektes“, „Besondere Information“, Funktion der Linie“ und „Geometrieangaben“ gespeichert. Die in den ATKIS-Vektorelementen enthaltenen Geometrieinformationen werden in die Datengruppen „Grundrisskennzeichen“, „Lageparameter“ und „Endpunkt der Linie“ überführt. Enthält die Dateneinheit die Datengruppe „Endpunkt der Linie“, so enthält das Grundrisskennzeichen (GKZ) den Startpunkt eines oder mehrerer Vektorelemente, die Lageparameter enthalten dessen Mittelpunkt, die Endpunkte des Vektorelementes werden entsprechend im Endpunkt der Linie gespeichert. Vektorelemente mit gleichem Startpunkt werden so zusammengefasst, das der Startpunkt zur Redundanzvermeidung nur einmal im Grundrisskennzeichen gespeichert wird, die untereinander verschiedenen Mittelpunkte und Endpunkte der Vektorelemente werden dann in die entsprechend wiederholten Datengruppen „Lageparameter“ und „Endpunkt

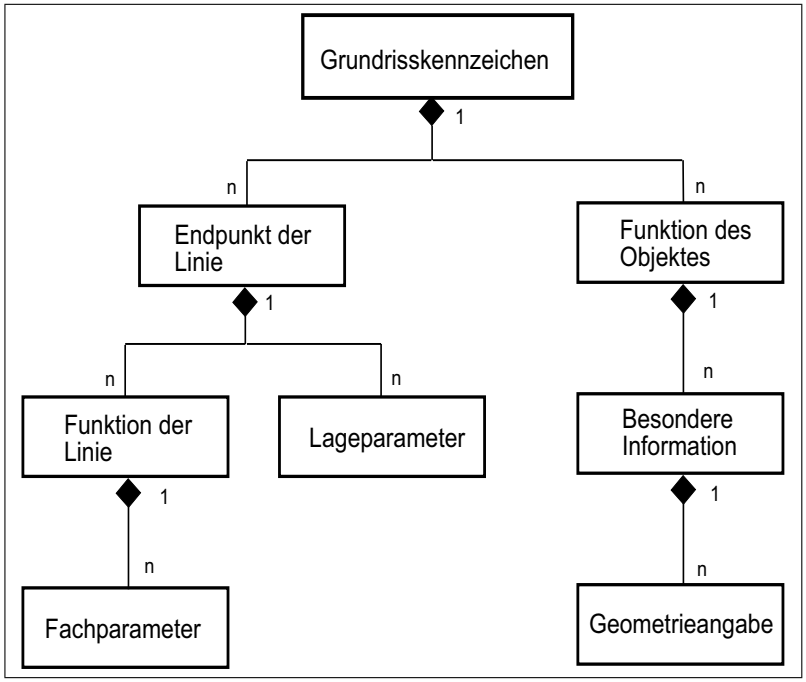


Abbildung 12: Datengruppen der Grundrissdatei

der Linie“ gespeichert. Eine solche Konstellation von Vektorelementen ist in Abbildung 11 zu sehen. Für die Datengruppe „Funktion des Objektes“ stellt das Grundrisskennzeichen die Objektkoordinate dar.
Die Abbildung der Attributdatei in EDBS-Datengruppen ist im Vergleich zur Grundrissdatei wesentlich einfacher aufgebaut, wie in Abbildung 13 zu sehen ist. Objekt-

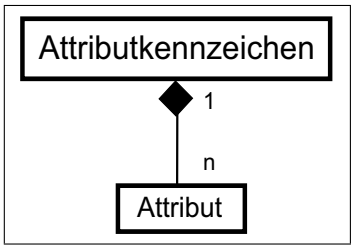


Abbildung 13: Datengruppen der Attributdatei

nummer sowie Objektteilnummer werden in der Datengruppe „Attributkennzeichen“ gespeichert, Attributtyp und Attributwert sind dann in der Datengruppe „Attribut“ untergebracht.
Eine vollständige Auflistung der den Datengruppen zugeordneten Datenaggreagten des ATKIS-DLMs findet sich in [Ver93, Band 2, S. 4-5].
Um die Struktur der Grundrissdatei in einen EDBS-Satz abzubilden, muss die

hierarchische Anordnung der Datengruppen in eine sequentielle Struktur festgelegt werden. In [Ver93], Anhang A werden die Abbildungsvorschriften der Datengruppen dokumentiert, daher sei an dieser Stelle nur ein Beispiel vermerkt (Abbildung 14). Die Abbildung zeigt die aus den Abbildungsregeln hergeleite-

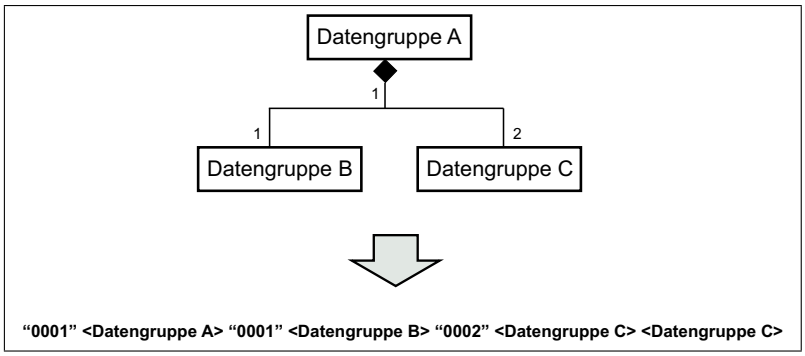


Abbildung 14: Sequentielle Anordnung der Datengruppen

te sequentielle Anordnung in EBNF Notation. Die Grammatik einer kompletten

EDBS-Grundrissdatei lässt sich somit in EBNF Notation wie folgt darstellen:

$$
\begin{aligned}
\langle \text{Grundrissdatei} \rangle &:= \text{“0001“ “0001“ } \langle \text{ULOB0000} \rangle \\
&\quad \text{WHF}_{\text{ULOB1000}} \{\text{ULOB1000}\}^{\text{WHF}_{\text{ULOB1000}}} \\
&\quad \text{WHF}_{\text{ULOB200F}} \{\text{ULOB200F}\}^{\text{WHF}_{\text{ULOB200F}}}; \\
\langle \text{ULOB0000} \rangle &:= \text{“DLOB0001“ “DLOB0002“ “DLOB0003“;} \\
\langle \text{ULOB1000} \rangle &:= \text{“DLOB1001“ “DLOB1002“ “DLOB1003“;} \\
&\quad \text{WHF}_{\text{ULOB110F}} \{\text{ULOB110F}\}^{\text{WHF}_{\text{ULOB110F}}} \\
&\quad \text{WHF}_{\text{ULOB1200}} \{\text{ULOB1200}\}^{\text{WHF}_{\text{ULOB1200}}}; \\
\langle \text{ULOB110F} \rangle &:= \text{WHF}_{\text{ULOB1100}} \{\text{ULOB1100}\}^{\text{WHF}_{\text{ULOB1100}}} \\
&\quad \text{WHF}_{\text{ULOB1110}} \{\text{ULOB1110}\}^{\text{WHF}_{\text{ULOB1110}}}; \\
\langle \text{ULOB1100} \rangle &:= \text{“DLOB1001“ } \ldots \text{“DLOB1008“;} \\
\langle \text{ULOB1100} \rangle &:= \text{“DLOB1111“ “DLOB1112“ “DLOB1113“;} \\
\langle \text{ULOB1200} \rangle &:= \text{“DLOB1201“;} \\
\langle \text{ULOB200F} \rangle &:= \text{WHF}_{\text{ULOB2000}} \{\text{ULOB2000}\}^{\text{WHF}_{\text{ULOB2000}}} \\
&\quad \text{WHF}_{\text{ULOB210F}} \{\text{ULOB210F}\}^{\text{WHF}_{\text{ULOB210F}}}; \\
\langle \text{ULOB2000} \rangle &:= \text{“DLOB2001“ } \ldots \text{“DLOB2008“;} \\
\langle \text{ULOB210F} \rangle &:= \text{WHF}_{\text{ULOB2100}} \{\text{ULOB2100}\}^{\text{WHF}_{\text{ULOB2100}}} \\
&\quad \text{WHF}_{\text{ULOB2110}} \{\text{ULOB2110}\}^{\text{WHF}_{\text{ULOB2110}}}; \\
\langle \text{ULOB2100} \rangle &:= \text{“DLOB2101“ } \ldots \text{“DLOB2106“;} \\
\langle \text{ULOB2110} \rangle &:= \text{“DLOB2111“;}
\end{aligned}
$$

Die in der EBNF verwendeten Abkürzungen „ULOBxxxx“ bzw. „DLOBxxx“ sind aus der „Allgemeinen Datenbeschreibung ATKIS“([Lan93]) des Landesvermessungsamtes Sachsen übernommen. Die Zuordnung der Bezeichner der entsprechenden Datenaggregate zu den Abkürzungen können dort entnommen werden.

3.4.3 Implementation des EDBS-Parsers

Um die in den EDBS Dateien enthaltenen DLM-Daten auszulesen, müssen die EDBS-Datenaggregate in die Datenstruktur des ATKIS Basis-DLM zurückgeführt werden. Um dies zu erreichen, wurde das ATKIS Datenmodell durch ein Klassenmodell nachgebildet. Die Klasse `ATKISObject` funktionert hierbei als Container für die fachlichen und geometrischen Daten. Die abstrakte Klasse `ATKISObjectPart` modelliert den Objektteil des ATKIS-Datenmodells. Da die Objektgeometrie punkt-, linen-, flächen- oder rasterförmige Ausprägung besitzen kann werden von `ATKISObjectPart` die Klassen `ATKISArea`, `ATKISLine`, `ATKISPoint` und `ATKISRaster` abgeleitet. Diese besitzen wiederum eine Liste von Vektorelementen, welche die geometrische Information repräsentiert. Die Klasse `TopoCoordinate` beinhaltet Datenstrukturen für Geographische Koordinaten, Gauss-Krüger Koordinaten, Rendering Koordinaten, sowie die Funktionalität zur Umrechnung der verschiedenen Koordinatensysteme. Abbildung 15 zeigt eine graphische Darstellung der Klassenstruktur als UML-Klassendiagramm. Im Diagramm werden nur die wichtigsten Klassen und deren Datenfelder gezeigt. Ferner sind die Namen der Datenfelder in deutscher Sprache gehalten, die Implementierung verwendet für Bezeichner konsequent die englische Sprache. Die eigentlichen Parserfunktionen sind in der Klasse `ATKISHandler` zusammengefasst. Der Parser baut zu wesentlichen Teilen auf Conta iner-Klassen der STL auf, insbesondere auf den Klassentemplates `vector<T, Alloc>` und `set<Key, Compare, Alloc>`. Weiterhin wurde auf die Klassen `string` und `ifstream` der Standardbibliothek zurückgegriffen. Der Parser liefert als Ergebnis eine Liste von Referenzen auf `ATKISObject`, implementiert als `vector<ATKISObject*>`.

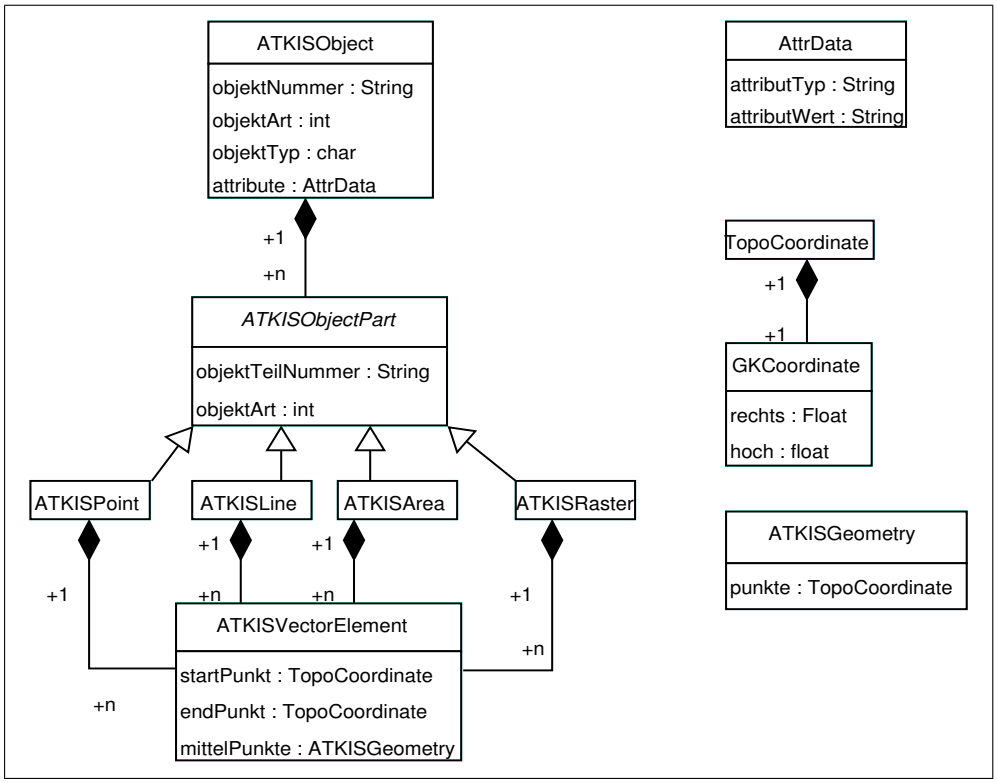


Abbildung 15: Klassenmodell des ATKIS DLM-Datenmodells

4 Geometrieerzeugung und Verwaltung

4.1 Terrain-Relief

In der Kartographie wird die Darstellung der Oberflächenstruktur der Erdoberfläche als Relief bzw. Reliefmodell bezeichnet. Der Begriff des Reliefmodells soll hier allein die Oberflächenstruktur der Landschaft beschreiben. Landschaftsobjekte wie Gebäude, Vegetation und Infrastruktur fallen nicht unter den Begriff des Reliefs. Diese Objekte werden quasi auf das Relief "aufgesetzt". Im Kontext von 3D-Echtzeitrendering werden Terrain-Reliefs durch eine Menge von dreiecksvermaschten Punkten beschrieben. Dieses Dreiecksnetz wird im allgemeinen als *Triangulation* bezeichnet. Die Triangulation eines Terrain-Reliefs lässt sich grundsätzlich in zwei Kategorien einteilen:

- Reguläre Triangulation
- Irreguläre Triangulation

Abbildung 16 veranschaulicht die Unterschiede der beiden Triangulationsarten. Dabei resultiert eine reguläre Triangulation direkt aus der regulären Anordnung

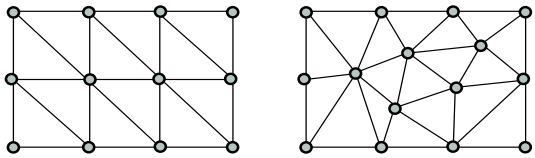


Abbildung 16: reguläre und irreguläre Triangulation

der Eckpunkte. Sind die zu triangulierenden Eckpunkte nicht in einem regulären Gitter angeordnet, so ergibt sich daraus zwangsläufig eine nicht-reguläre Triangulation. Der Algorithmus zur regulären Triangulation ist im Gegensatz zur irregulären Triangulation trivial.
Im Rahmen dieser Arbeit wurde aus den Vermessungsdatensätzen ATKIS-DLM und DGM ein Reliefmodell erstellt. Dabei wurde auf Grund der verwendeten Daten auf ein hybrides Verfahrem aus regulärer und irregulärer Triangulation zurückgegriffen.

4.2 Geometrie Datenstrukturen

4.2.1 Triangle Strip und Indexed Face Set

Die Datenhaltung für Geometriedaten ist für die Leistungsfähigkeit eines Rendering-Systems von entscheidender Bedeutung. Es ist sinnvoll, die Menge der entstehenden Daten soweit wie möglich zu begrenzen, zum Einen um den Speicherverbrauch an sich zu minimieren, zum Anderen, um den Transportaufwand der

Daten vom Arbeitspeicher in den Graphikspeicher zu minimieren. Werden aus einer Liste von vier Eckpunkten $(v_0 \dots v_3)$ zwei Dreiecke d_0, d_1 gebildet, so teilen sich diese Dreiecke zwei Eckpunkte aus der Liste. Speichert man jede Koordinate eines Eckpunktes als Fließkommawert mit 32 Bit Genauigkeit, so fallen pro Eckpunkt 12 Bytes an Daten an, für ein einzelnes Dreieck folglich 36 Bytes. Da sich die zwei Dreiecke zwei Eckpunkte teilen, besteht eine Optimierung nun aus der Vermeidung von redundanter Speicherung der Eckpunkte. Eine Datenstruktur, die dies leistet, ist das sog. *Indexed Face Set*. Hierbei werden für jedes Dreieck (Face) nicht mehr die Eckpunktkoordinaten selbst gespeichert, sondern Indizes, welche die entsprechenden Eckpunkte aus einer Liste referenzieren. Speichert man die Indizes als positive 16 Bit Ganzzahl (unsigned short), so können damit 65536 Eckpunkte adressiert werden. Mit dieser Menge an verfügbaren Eckpunkten kann z.B. ein 256×256 Eckpunkte großes Rechteck erzeugt werden. Dies kann für sehr große Flächenstücke bzw. sehr hohe Abtastraten nicht ausreichend sein. Da das in dieser Arbeit erzeugte Landschaftsmodell aus einer großen Zahl wesentlich kleinerer Einzelflächen zusammengesetzt ist und das Gelände mit einem Punktabstand von $10\ m$ abgetastet wurde, konnten 16 Bit-Indizes verwendet werden.
Eine weitere optimierte Datenstruktur ist der sog. *Triangle Strip*. Dabei ist eine Menge von Dreiecken so angeordnet, dass sich das jeweils nächste Dreieick in der Liste genau zwei Eckpunkte mit seinem Vorgänger teilt. Vorteilhaft an dieser Datenstruktur ist, dass zur Erzeugung von n Dreiecken lediglich $n + 2$ Eckpunkte gespeichert werden müssen, da zur Definition des jeweils nächsten Dreiecks nur ein Eckpunkt benötigt wird. Ferner kann auf Indexlisten verzichtet werden, da die Reihenfolge der Eckpunkte implizit gegeben ist. Allerdings unterliegt die als Triangle Strip repräsentierbare Geometrie Einschränkungen. So muss eine Menge von Dreiecken, welche die Triangle-Strip Bedingung nicht erfüllt, in mehrere Strips zerlegt werden. Dies kann für komplexere Geometrien relativ aufwändig werden. Daher werden in dieser Arbeit Triangle Strips nur dann verwendet, wenn die Geometrie direkt in dieser Form repräsentiert werden kann. Dies trifft auf alle Geometrien zu, welche aus linienförmigen ATKIS-Objekten erzeugt werden. Dies ist insbesondere bei Straßen und Bahnlinien der Fall.

4.3 Hardwarespezifische Aspekte

Um größere Mengen an Geometriedaten zu rendern, ist es von Vorteil, die Daten in größeren Stücken an die Grafikhardware zu übertragen, anstatt jeden Vertex einzeln, wie es z.B. im *Immediate Mode* von OpenGL der Fall ist. Werden größere Datenblöcke an die Grafikkarte gesendet, so veringert sich der Overhead durch API-calls signifikant. Als Umsetzung dieses Prinzips stellt z.B. OpenGL sog. *Vertex Arrays* zur Verfügung. Vertex Arrays sind im Wesentlichen Speicherblöcke im Hauptspeicher, in denen Vertex Daten abgelegt sind. Zum Rendern wird OpenGL

ein Zeiger auf diese Daten übergeben, um dann die Daten in das Video RAM zu übertragen. So ist es möglich, Daten für Vertices, Eckpunktfarben und Texturkoordinaten in Blöcken zu übergeben. Um die in Kapitel 4.2.1 beschriebenen Indexed Face Sets zu realisieren, besteht weiterhin die Möglichkeit, einen Zeiger auf Index Daten zu übergeben.
Eine Erweiterung der Vertex Arrays stellen die *Vertex Buffer Objects (VBOs)* dar. VBOs wurden mit der Spezifikation von OpenGL 1.5 Corefeature des Standards. Der Unterschied zu den Vertex Arrays besteht darin, dass die Vertexdaten im VRAM der Graphikkarte gepuffert werden. Im Unterschied zu *Display Listen*, bei denen die Datenhaltung auch serverseitig erfolgt, können die von den VBOs verwalteten Vertexdaten performant geändert werden. Weiterhin besteht die Möglichkeit, Vertexdaten direkt via DMA in den Grafikkartenspeicher zu schreiben, ohne den Umweg über den Hauptspeicher nehmen zu müssen.
Der Vorteil von Vertex Arrays und VBOs liegt, neben dem deutlichen Performance-Gewinn, in der Möglichkeit, die zu rendernden Vertexdaten zu ändern. Dies ist im Hinblick auf Level of Detail Funktionalitäten eine notwendige Eigenschaft, welche ein Rendering-System zur Verfügung stellen muss.

4.4 Implementationsdetails

4.4.1 Klassenstruktur

Die Verwaltung der zu rendernden Geometriedaten, welche Vertex-, Farb-, Normalen- und Texturkoordinateninformation umfasst, wird durch die Klasse `Geometry` realisiert. Die Klasse ist in der Lage, sowohl Daten in der Form von Indexed Face Sets als auch Triangle Strips zu verwalten. Gesteuert wird dies durch das Flag `isTriangleStrip`. Die Liste der Faces ist im Falle von Triangle Strips leer. Um die Erzeugung der Geometrie zu vereinfachen, werden Vertices, Vertex-Normalen und Texturkoordinaten durch Instanzen der Klasse `Vector3` repräsentiert. Diese Klasse implementiert einen Vektor aus $\mathbf{R}^3$ mit den Operationen Skalarprodukt, Kreuzprodukt, Addition und Multiplikation. Sie dient dazu, mathematische Berechnungen auf Vektorebene duchzuführen. Um die Verwendung von Vertex Arrays bzw. VBOs als optionalen Aspekt zu realisieren, wurden die Datenspeicher für die Vertexarrays redundant als Array von Single Precision Float implementiert, einem für die Verwendung von Vertex Arrays günstigen Format.
Um Operationen wie View Frustum Culling zu ermöglichen, wurde in der `Geometry` Klasse eine *Axis Aligned Bounding Box (AABB)* implementiert.

4.4.2 Vertexnormalen

Die `Geometry` Klasse implementiert die Erzeugung von Vertexnormalen. Hierzu werden für jeden Vertex die umliegenden Dreiecke bestimmt. Die Flächennormalen

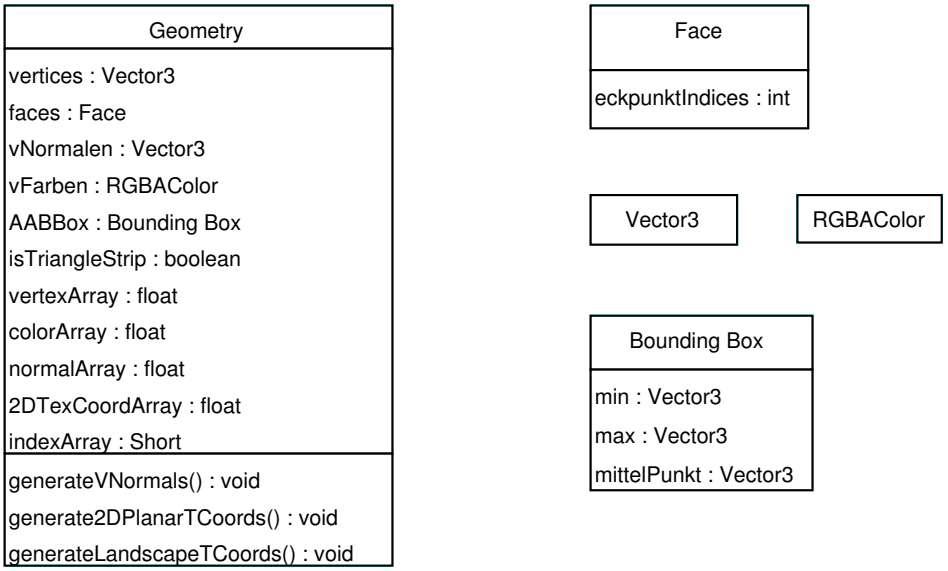


Abbildung 17: Geometriedaten Klassenmodell

werden dann mit der Flächengröße als Gewichtungsfaktor zur Vertexnormale aufaddiert.

4.4.3 Beleuchtung

Da die Beleuchtung großer Terrains in Echtzeit auf Grund der großen Zahl an Vertices relativ aufwändig ist, wird zur Performancesteigerung ein vorberechnetes, statisches Beleuchtungsmodell implementiert. Ein einfaches *Gouraud Shading* Modell mit einer eizigen, direktionalen Lichtquelle, welche als Simulation der Sonne dient. Die Lichteinfallsrichtung sowie die Lichtfarbe sind, um verschiedene Tageszeiten simulieren zu können, parametriesierbar. Aus den Ergebnissen der Beleuchtungsberechnungen resultiert ein entsprechender Wert der Vertexfarbe. Der Beleuchtungseffekt wird dann beim Rendering durch Modulation der Vertexfarbe mit dem entsprechenden Farbwert der verwendeten Textur erzielt.
Selbstverständlich stellt dieses Beleuchtungsmodell nur ein sehr einfaches Abbild der Realität dar, eine realistische Beleuchtungssimulation, welche z.B. atmosphärische Effekte und ähnliches berücksichtigt, lag jedoch außerhalb des Fokusses dieser Arbeit. Eine Möglichkeit zur Erhöhung des Realitätsgrades besteht in der Simulation von Selbstverschattung des Terrains. In [Mar02] wird ein einfacher Algorithmus hierfür vorgestellt. Da die Beleuchtung des Terrains nicht in Echtzeit erfolgen soll, ist dieses Verfahren zur Schattensimulation geeignet.

4.4.4 Texturkoordinaten

Die Texturierung des Terrains hängt im Wesentlichen von der Texturkoordinatenerzeugung ab. Eine einfache Art der Koordinatenerzeugung besteht in der Generierung von sog. *Planarer Texturkoordinaten*. Hierbei wird der Wertebereich der s, t Koordinaten der Tetxur auf x, z Ausdehnung der Axis Aligned Bounding Box des zu texturierenden Objektes abgebildet. Seien nun die Punkte b^{min}, b^{max} die Ausdehnung der Bounding Box des zu texturierenden Objektes, dann berechnen sich die Texturkoordinaten eines Vertexes v durch:

$$\begin{aligned} s &= \left(v_x - b_x^{min}\right) \cdot \frac{1}{b_x^{max} - b_x^{min}} \\ t &= \left(v_z - b_z^{min}\right) \cdot \frac{1}{b_z^{max} - b_z^{min}} \end{aligned}$$

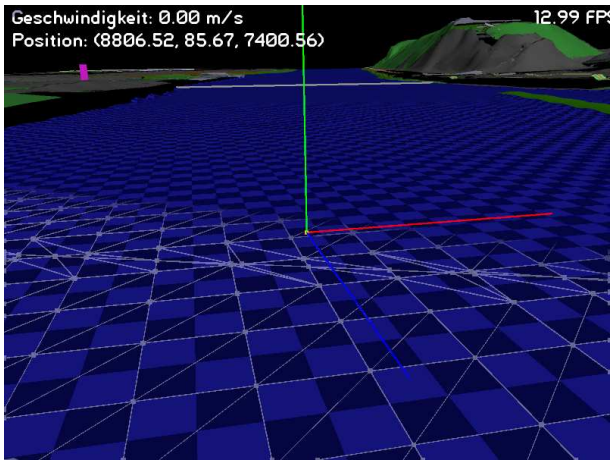
Um mehrere zusammenhängende Flächen mit einer einheitlichen Textur, beispielsweise einer gekachelten Textur, zu überziehen, eignen sich planare Texturkoordinaten nicht. Abbildung 18 zeigt die entstehenden Artefakte, welche an den Nahtstellen zweier Flächen des Reliefmodells auftreten. Die Artefakte entstehen zum Einen durch die unterschiedlichen Größen der Bounding Boxes, zum Anderen durch deren Überlappung. Um diese Artefakte zu vermeiden, muss sich die Generierung von Texturkoordinaten an einem regelmäßigen Gitter orientieren. Hier kann beispielsweise das dem DGM zugrunde liegende Gitternetz gewählt werden. Die so erzeugten Texturkoordinaten werden in dieser Arbeit als *Landschaftstexturkoordinaten* bezeichnet. Die Berechnung der Texturkoordinaten verläuft ähnlich wie die Berechnung planarer Texturkoordinaten. Es wird jedoch zu dem aus der Bounding Box bestimmten Punkt b^{min} der am nächsten liegenden Gitterpunkte berechnet. Der Gitterpunkt g^{min} lässt sich durch die Floor Operatoren und einer gegebenen Gitterweite w wie folgt berechnen:

$$\begin{aligned} g_x^{min} &= \lfloor\left(b_x^{min} \cdot \frac{1}{w}\right) \cdot w \\ g_z^{min} &= \lfloor\left(b_z^{min} \cdot \frac{1}{w}\right) \cdot w \end{aligned}$$

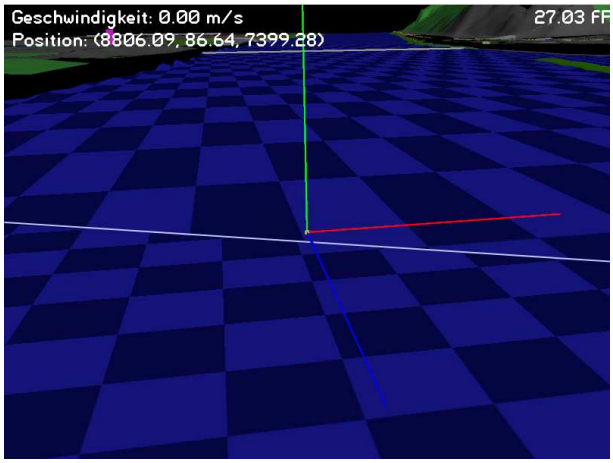
Die abschließende Berechnung der Texturkoordinaten erfolgt nun durch:

$$\begin{aligned} s &= \left(v_x - g_x^{min}\right) \cdot \frac{1}{w} \\ t &= \left(v_z - g_z^{min}\right) \cdot \frac{1}{w} \end{aligned}$$

Sollen für eine durch ein linienförmiges ATKIS-Objekt erzeugte Geometrie Texturkoordinaten berechnet werden, so muss dafür gesorgt werden dass eine Textur nahtlos über die erzeugte Geometrie gekachelt werden kann. Um dies zu errei-



(a) Landschaftstexturkoordinaten



(b) planare Texturkoordinaten

Abbildung 18: Erzeugung von TexturKoordinaten

chen, muss die s oder t Richtung der Textur auf die Mittellinie der Geometrie zyklisch abgebildet werden. Abbildung 19 zeigt die Vorgehensweise zur Texturkoordinatenerzeugung auf einem Triangle Strip. Die gestrichelten Linien stellen jeweils die Ränder in s-Richtung der gekachelten Textur dar. An den Punkten ist jeweils die durch die zyklische Abbildung entstandene t-Koordinaten notiert. Seien nun c_i und c_{i+1} zwei aufeinanderfolgende Punkte der Mittellinie, so kann man bei gegebener Texturweite w die t-Koordinaten durch

$$t = |\vec{c_{i+1}} - \vec{c_i}| \cdot \frac{1}{w}$$

berechnen. Der Faktor w kann dazu benutzt werden, die Textur maßstabsgetreu auf die Geometrie abbzubilden.

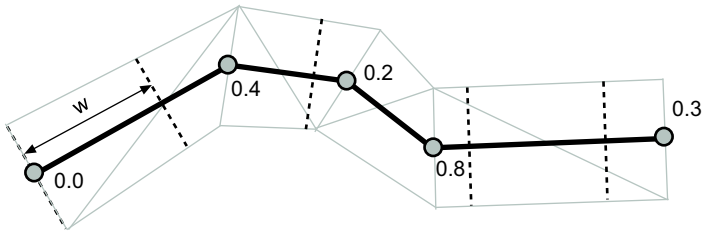


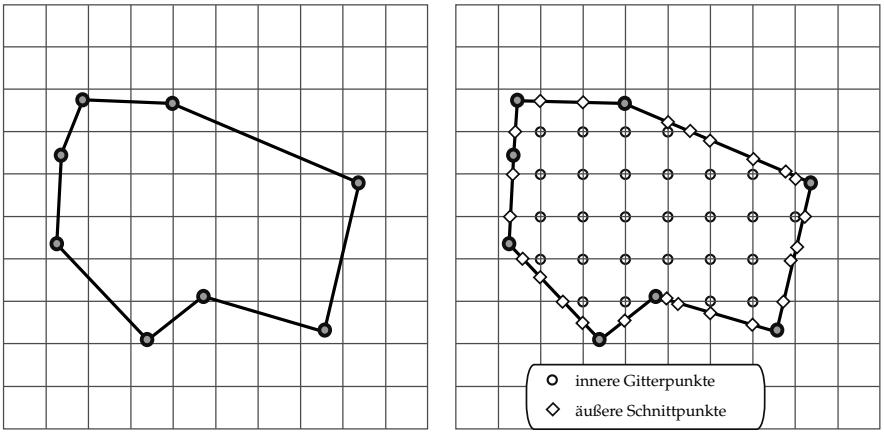
Abbildung 19: Texturkoordinaten auf Triangle Strip

4.5 Triangulierung flächenförmiger Objekte

Die Geometrieinformation der in den ATKIS-Daten modellierten flächenförmigen Geometrien liegt nach dem Einlesen der entsprechenden EDBS-Grundrissdateien in Form einer Umrisslinie vor, welche die modellierte Fläche beschreibt. Diese Repräsentation der Flächen eignet sich nicht für die Verwendung in einem Echtzeit-Rendering System, daher ist eine Zerlegung der durch die Umrisslinie beschriebenen Flächen in Dreiecke notwendig. Dabei muss diese Triangulierung der Flächen das zugrunde liegende DGM miteinbeziehen, um das Oberflächenrelief der modellierten Landschaft korrekt abzubilden. Die Trennung von Höhen- und Umrissdaten macht diese Fusionierung der Datensätze nötig. Um eine Umrisslinie mit dem regelmäßigen Gitternetz des DGMs zu verbinden, sind im Wesentlichen zwei Schritte erforderlich:

- Bestimmung der m innenliegenden Punkte
- Bestimmung der s Schnittpunkte der Umrisslinie mit den Gitternetzlinien

Abbildung 20 zeigt die Vorgehensweise. Die verwendeten Verfahren arbeiten im



(a) Ausgangssituation (b) Hinzugefügte Punkte

Abbildung 20: Fusion von Umrisslinie und DGM

zweidimensionalen, respektive auf den x, z Koordinaten des Rendering-Koordinatensystems (vgl. Kapitel 2.3.3). Da in den Vermessungsdaten keine senkrechten Steigungen vorkommen können (vgl. Kapitel 3.1), ist dies ohne die Entstehung von Fehlern möglich. Im Folgenden sollen nun die zwei Schritte genauer beschrieben werden.

4.5.1 Hinzufügen der inneren Gitterpunkte

Um die Menge der DGM-Gitterpunkte innerhalb der Umrisslinie einer Fläche zu bestimmen, wird jeder Gitterpunkt innerhalb der Axis Aligned Bounding Box getestet, ob er sich innerhalb der Umrisslinie befindet und bei positiv verlaufendem Test zur Liste der inneren Gitterpunkte hinzugefügt. Hierbei werden die Punkte "zeilenweise" in x-Richtung angeordnet. Abbildung 21 zeigt die AABB um das

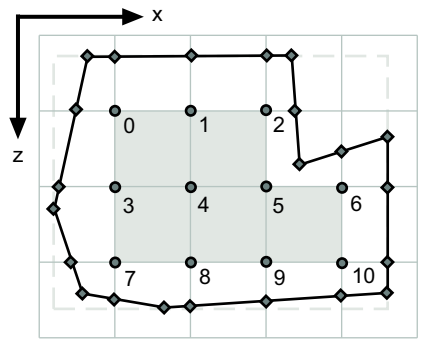


Abbildung 21: Indizierung der inneren Gitterpunkte

Polygon als gestrichelte Linie, sowie die Reihenfolge, in welcher die Gitterpunkte abgelegt werden.
Es existiert eine Vielzahl an Algorithmen, welche einen Punkt-Polygontest realisieren. Die Problematik an den aus den ATKIS-Daten erzeugten Polygonen ist deren praktisch beliebige Ausprägung der Polygonart, die Umrisslinien können sowohl konvex als auch konkav sein, die Umlaufrichtung des Polygons ist nicht festgelegt. In [Hai92] werden verschiedene Algorithmen hinsichtlich ihres Laufzeitverhaltens an verschiedenen Polygonarten getestet. An Hand der dort erzielten Ergebnisse erweist sich der *MacMartin Line Crossing Test* als effizienteste Lösung, daher fiel die Entscheidung auf diesen Algorithmus. Das Verfahren basiert auf dem *Line Crossing Test*, bei dem von einem gegebenen Punkt p aus ein Strahl mit beliebiger Richtung d erzeugt wird. Für jede Kante des gegebenen Polygons P wird nun der Schnittpunkt mit dem Strahl bestimmt. Ist die Anzahl der Schnittpunkte eine gerade Zahl, so liegt der Punkt außerhalb des Polygons, bei ungerader Anzahl der Schnittpunkte liegt der Punkt innerhalb. Abbildung 22 zeigt, dass die Strahlen, welche im Punkt p_o ihren Ursprung haben, stets eine gerade

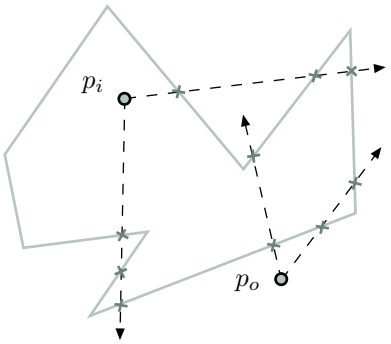
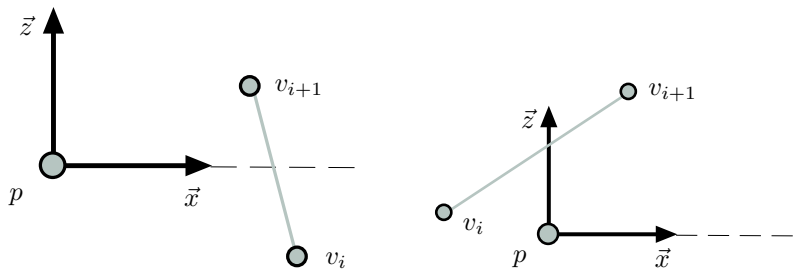


Abbildung 22: Schnitttest mittels Line Crossing

Anzahl an Schnittpunkten aufweisen, für den Punkt p_i trifft das Gegenteil zu. Es existiert eine von *Stuart MacMartin* entwickelte Optimierung des Line Crossing Verfahrens, welche die Zahl der Schnitttests zwischen Strahl und Polygonkante deutlich minimiert. Im Algorithmus von MacMartin wird der zu testende Punkt als Ursprung des Koordinatensystems gesehen. Als Richtung des Strahls wird ein Basisvektor des Koordinatensystems verwendet; in dieser Arbeit wäre dies der Vektor $(1,0,0)^T$ bzw. der Vektor $(0,0,1)^T$. Da sämtliche Berechnungen in der x/z Ebene erfolgen, werden die x- bzw. z-Komponenten der Vektoren verwendet. In diesem Beispiel sei der verwendete Richtungsvektor des Strahls $d = (1,0,0)^T$. Für ein Polygon P, welches aus einer Liste von n Eckpunkten $(v_0, v_1, \ldots, v_{n-1})$ besteht, wird nun jede Kante (v_i, v_{i+1}) auf einen Vorzeichenwechsel der z-Komponenten von v_i und v_{i+1} bezüglich des Ursprungs p untersucht. Bei einem solchen Vorzeichenwechsel ist ein Schnitt des Strahls mit der Kante prinzipiell möglich. Sind nun die x-Komponenten von v_i und v_{i+1} bezüglich des Ursprungs p positiv, so kann, ohne einen eigentlichen Schnitttest mit dem Strahl und der Polygonkante durchführen zu müssen, ein Schnitt des Strahls mit der Kante (v_i, v_{i+1}) gemeldet werden. Unterscheiden sich die x-Komponenten in ihrem Vorzeichen, so muss ein extra Schnitttest zwischen dem Strahl und der Polygonkante durchgeführt werden. Abbildung 23 veranschaulicht die Situationen der Vorzeichenwechsel.

4.5.2 Hinzufügen der Schnittpunkte mit den Gitterlinien

Damit die aus den ATKIS-Daten gewonnene Umrisslinie dem durch das DGM vorgegebenen Geländerelief folgen kann, müssen die Schnittpunkte der Umrisslinie mit den Gitterlinien des DGMs bestimmt werden. Hierbei erfolgt die Berechnung der Schnittpunkte für jede Polygonkante bestehend aus den Eckpunkten (v_i, v_{i+1}). Jede Polygonkante kann dabei beliebig viele Gitterlinien in x- und z-Richtung schneiden. Der Algorithmus zur Schnittpunktberechnung muss dabei gewährleisten, dass die Schnittpunkte in der richtigen Reihenfolge ermittelt



(a) VZW bezügl. z-Koordinate

(b) VZW bezügl. x- und z-Koordinate

Abbildung 23: MacMartin Schnitttest

und in die Umrisslinie eingefügt werden. Dazu wird der Schnittpunkt der Gerade $\vec{v_i} + t \cdot (\vec{v_{i+1}} - \vec{v_i})$ mit den entsprechenden Gitterlinien in x- und z-Richtung berechnet. Die Reihenfolge, in der die Gitterlinien abgearbeitet werden, ist für das Ergebnis nicht von Bedeutung, da die Schnittpunkte bezüglich des errechneten Geradenparameters t in aufsteigender Reihenfolge sortiert werden. Abbildung 24

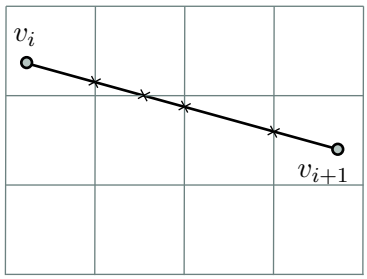


Abbildung 24: Schnittpunkte mit dem Gitternetz

verdeutlicht das Vorgehen und die Notwendigkeit, die Schnittpunkte zu sortieren. Die Implementation der Schnittpunktsortierung wird durch die Containerklasse `set<Key, Compare, Alloc>` der STL automatisch realisiert.

4.5.3 Triangulierung der Punktmenge

Nach Durchführung der in Abschnitt 4.5.1 und Abschnitt 4.5.3 beschriebenen Arbeitsschritte liegen flächenförmige ATKIS-Objektgeometrien als eine Menge von Punkten vor. Hierbei lässt sich die Punktmenge in eine Menge, welche die innerhalb der Fläche liegenden Gitterpunkte des DGMs enthält, und in eine Menge, welche die um die Schnittpunkte mit den Gitterlinien des DGMs erweiterte Punktmenge der Umrisslinie enthält, aufteilen. Diese beiden Punktmengen stellen die Eingabedaten eines Triangulationsverfahrens dar. Im speziellen Kontext der Vermessungsdaten werden an den zu verwendenden Algorithmus zwei grundsätz-

liche Anforderungen gestellt:

- Um die Landschaft originalgetreu abzubilden, darf die Umrisslinie der zu triangulierenden Flächen nicht verändert werden
- Auf Grund der großen Datenmengen muss der gewählte Algorithmus ein günstiges Laufzeitverhalten aufweisen

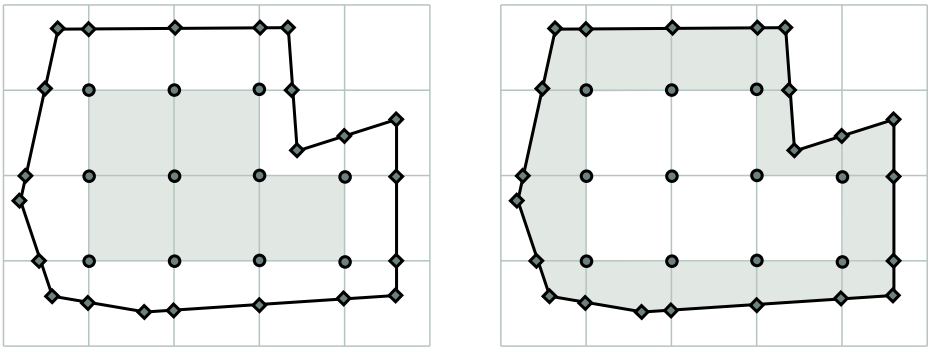
In der Literatur wird eine Vielzahl von Triangulationsalgorithmen beschrieben. Eine erste grundsätzlich zu treffende Entscheidung stellt die Verwendung der Art des Triangulationsalgorithmuses dar. Die Wahl besteht hierbei zwischen

- Triangulation in 3D
- Triangulation in 2D

Triangulationen auf 3D-Punktwolken sind im Vergleich zu 2D-Triangulationen komplexer, sowohl aus Sicht des Implementationsaufwandes als auch aus Sicht des Berechnungsaufwandes. Die gleiche Argumentation, welche zu einer Durchführung der Vorverarbeitungsschritte (vgl. Abschnitt 4.5.1 und Abschnitt) im Zweidimensionalen führte, greift auch hier: Bedingt durch die DGM Abtastung können im Gelände keine Überhänge oder senkrechte Steigungen vorkommen, daher ist eine Vereinfachung der Berechnungen auf die Ebene möglich. Weiterhin müssen für die Punktmenge keine vollständig geschlossenen Hüllen berechnet werden, wofür eine Triangulation in 3D unumgänglich wäre. Ferner ergibt sich durch die Wahl eines 2D-Verfahrens die Möglichkeit, Algorithmen in Betracht zu ziehen, welche die Triangulation konvexer Umrisslinien bewältigen können.
Ein Standardverfahren zur Triangulation ist in der Computergraphik sicherlich die *Delaunay Triangulation*. Diese Triangulation liefert ein Dreiecksnetz, bei welchem die kleinsten Innenwinkel eines jeden Dreiecks die maximal mögliche Größe erreichen. An dieser Stelle soll auf die speziellen Details des Verfahrens nicht weiter eingegangen und stattdessen auf die Literatur zum Verfahren verwiesen werden. Die Problematik in der Verwendung der Delaunay Triangulation liegt in der Form der Umrisslinie der zu triangulierenden Fläche: Die Delaunay Triangulation liefert für eine gegebene Punktmenge stets eine konvexe Hülle um diese Punktmenge. Die in den ATKIS-Daten modellierten Flächen besitzen jedoch sehr oft konkave Formen. Daher müsste bei Verwendung der Delaunay Triangulation dafür gesorgt werden, dass alle durch den Algorithmus erzeugten Kanten, welche sich außerhalb der Umrisslinie befinden, entfernt werden. Zu der bei der Delaunay Triangulation notwendigen Berechnung der Innenwinkel muss auf trigonometrische Funktionen zurückgegriffen werden, was bei der großen Menge der Daten einen nicht unerheblichen Einfluss auf die Laufzeit hätte.
Eine Besonderheit der zu triangulierenden Punktmenge ergibt sich aus den zugrunde liegenden DGM Daten und den beschriebenen Vorverarbeitungsschritten der Umrisslinie: Ein Großteil der innen liegenden Punkte einer Fläche liegt in

Form des regelmäßigen Gitters des DGMs vor. Dies ist insbesondere bei großen Flächen der Fall. Eine Triangulation dieser regelmäßig angeordneten Punkte ist letztendlich trivial und rechtfertigt die Verwendung eines teuren Algorithmuses nicht. Daher ist es sinnvoll, die Triangulation eines flächenförmigen ATKIS-Objektes in zwei Schritte aufzuteilen:

- Regelmäßige Triangulierung der innen liegenden Punkte (Innere Punktmenge)
- Unregelmäßige Triangulierung der außen liegenden Punkte (Umrisspunktmenge)

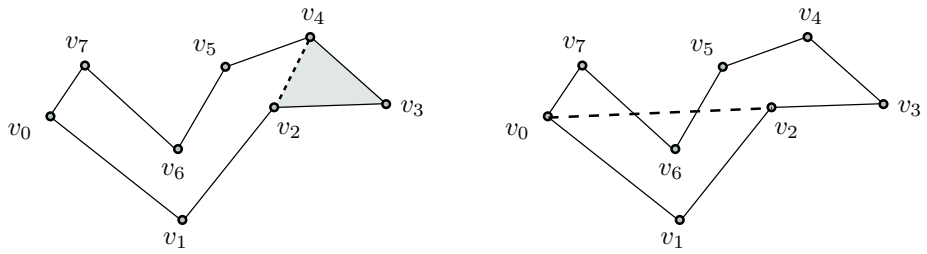


(a) Innere Punkmenge (b) Umrisspunktmenge

Abbildung 25: Unterteilung der Punktmengen

Abbildung 25 zeigt die zwei Bereiche jeweils Grau unterlegt. Die Umrisspunktmenge besteht hierbei aus den nach Kapitel 4.5.3 berechneten Punkten der Umrisslinie, zuzüglich derjenigen innen liegenden Punkte, welche zur Verankerung mit der Inneren Punktmenge notwendig sind. Durch diese Aufteilung kann eine große Zahl an Dreiecken auf eine unkomplizierte, laufzeitfreundliche Art generiert werden.
Für die Triangulation der Umrisspunktmenge muss ein Verfahren zum Einsatz kommen, welches in der Lage ist, ein Dreiecksnetz sowohl aus konvexen als auch aus konkaven zweidimensionalen Polygonen zu generieren. Ein Algorithmus, der dies leistet, ist das sog. *Ear Clipping* (manchmal auch als *Ear Cutting* bezeichnet). Der Algorithmus ist in der Lage, konvexe und konkave, *einfache Polygone* in eine Menge von Dreiecken zu zerlegen. Hierbei wird ein Polygon P, bestehend aus n Eckpunkten $(v_0, \ldots, v_{n-1})$, als "einfach" bezeichnet, wenn es zu keiner Überschneidung von Kanten kommt, d.h. wenn jeder Eckpunkt Start- bzw. Endpunkt von jeweils genau einer Kante (v_i, v_{i+1}) ist. Der Ear Clipping Algorithmus liegt in der Aufwandsklasse $O(n^2)$ (vgl. [Ebe02]) und ist einfach zu implementieren. Das Verfahren basiert grundlegend darauf, von einem Polygon sukzessive überstehende

Ecken, die sog. "Ohren", abzuschneiden. Ein Ohr ist hierbei ein Dreieck, geformt aus drei aufeinanderfolgenden Eckpunkten v_i, v_{i+1}, v_{i+2} des Polygons, in dessen Innerem sich keine weiteren Eckpunkte des Polygons befinden. Abbildung 26 veranschaulicht den Sachverhalt. Der Algorithmus durchläuft nun die Liste der Eck-



(a) Ohr (b) kein Ohr

Abbildung 26: Definition eines Ohrs

punkte des Polygons. Bilden die Eckpunkte v_i, v_{i+1}, v_{i+2} ein Ohr, so kann das durch die drei Punkte beschriebene Dreieck in die Triangulation mit aufgenommen werden. Der Eckpunkt v_{i+1} wird dann aus dem Polygon entfernt und das Polygon weiter durchlaufen. Der Algorithmus terminiert, wenn die Anzahl der verbleibenden Eckpunkte im Polygon drei erreicht hat. Der Algorithmus kann optimiert werden, indem ein Eckpunkttrippel nur dann auf die Ohreigenschaft getestet wird, wenn die an Eckpunkt v_{i+1} anknüpfenden Kanten eine konvexe Ecke formen. In Abbildung trifft dies auf alle Eckpunkte, mit Ausnahme der Eckpunkte v_2 und v_6 zu. Um zu testen, ob es sich bei einem gegebenen Eckpunkttrippel um ein Ohr handelt, muss weiterhin nicht für jeden Eckpunkt des Polygons getestet werden, ob er sich innerhalb des durch das Eckpunkttripplet geformten Dreiecks befindet. Es ist ausreichend, diesen Test für alle konkaven Eckpunkte des Polygons durchzuführen. Um das Entfernen von Eckpunkten aus der Umrisslinie effizient zu realisieren, werden die Eckpunkte in einer doppelt verketteten linearen Liste gespeichert. Listing 2 zeigt den Ablauf des Algorithmuses in Pseudo Code.

Listing 2: Ear Clipping Algorithmus

```
1 Eingabe: Liste vList //Liste der Eckpunkte
2          Liste vKonkav //Liste der konkaven Eckpunkte
3
4 v_i = vList.begin();
5 v_{i+1} = v_i.next();
6 v_{i+2} = v_{i+1}.next();
7
8 WHILE(Länge vList > 3){
9         IF(!isEar(v_i, v_{i+1}, v_{i+2}, vKonkav)){
```

```
10                  v_i++;
11                  v_{i+1}++;
12                  v_{i+2}++;
13          }
14          ELSE{
15                  new Triangle(v_i, v_{i+1}, v_{i+2});
16                  lösche v_{i+1} aus vList;
17                  v_{i+1} = v_{i+2}
18                  v_{i+2} = v_{i+1}.next();
19                  berechne konkave Eckpunkte neu;
20          }
21  }
```

Die Funktion `isEar(`v_i, v_{i+1}, v_{i+2}`, vKonkav)` prüft zunächst, ob v_{i+1} in der Liste der konkaven Eckpunkte enthalten ist. Sollte dies nicht der Fall sein, so wird für jeden konkaven Eckpunkt getestet, ob er sich innerhalb des Dreiecks v_i, v_{i+1}, v_{i+2} befindet. Dies geschieht mittels der Baryzentrischen Koordinate bezüglich des Dreiecks v_i, v_{i+1}, v_{i+2}.
Die Berechnung, ob ein Eckpunkt v_{i+1} zu einer konkaven oder konvexen Ecke des Polygons gehört, erfolgt mittels

$$(\vec{v_{i+1}} - \vec{v_i}) \times (\vec{v_{i+2}} - \vec{v_{i+1}})$$

Da die Eckpunkte des Polygons in der x/z-Ebene liegen, unterscheiden sich die Vorzeichen der y-Komponente des so berechneten, auf den beiden an v_{i+1} angrenzenden Kanten senkrecht stehenden Vektors in Abhängigkeit der konvexen oder konkaven Ausprägung der Ecke. Ob das Vorzeichen für eine konvexe bzw. konkave Kante positiv oder negativ ist, hängt von der Umlaufrichtung des Polygons ab. Die möglichen Fälle sind hier:

Eckpunkt v_{i+1}	**CW**	**CCW**
konkav	positives VZ	negatives VZ
konvex	negatives VZ	positives VZ

Die Umlaufrichtung des Polygons berechnet sich, indem man für jeden Eckpunkt v_i einen Faktor s mittels

$$s = \sum_{i=0}^{n-1} v_i^x \cdot v_{i+1}^z - v_{i+1}^x \cdot v_i^z$$

berechnet. Ist $s < 0$, so ist die Umlaufrichtung des Polygons CCW, ansonsten CW. Der Aufwand des Ear Clipping Algorithmus hängt im Wesentlichen von der An-

zahl der konkaven Ecken des zu triangulierenden Polygons ab, die Komplexität für ein Polygon mit n Eckpunkten wird in der Literatur mit $k \cdot O(n^2)$ angegeben, wobei k die Anzahl der konkaven Eckpunkte ist. Im ungünstigsten Fall strebt der Aufwand also gegen $O(n^3)$.
Zur Triangulation der Umrisspunktmenge werden die einzelnen Gitterzellen, welche die Umrisspunktmenge enthalten, getrennt betrachtet. Abbildung 27 verdeutlicht das Vorgehen. Dieses Vorgehen erlaubt die Verankerung der Umrisspunktmenge mit der Triangulation der innen liegenden Punkte. Der Aufwand zur Tri-

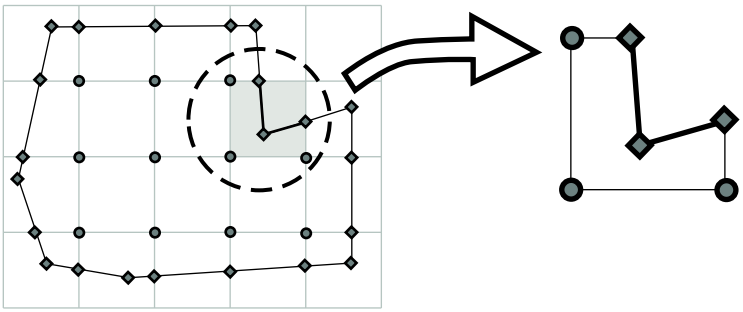


Abbildung 27: Unterteilung in einzelne Gitterzellen

angulation der Umrisspunktmenge lässt sich weiter optimieren, indem der Ear Clipping Algorithmus zur Triangulierung der Gitterzellen nur dann zum Einsatz kommt, wenn dies unbedingt erforderlich ist. Zwingend nötig ist der Einsatz von Ear Clipping letztendlich nur, wenn das durch die zu triangulierende Gitterzelle gebildete Polygon konkave Eckpunkte besitzt. Ist das Polygon konvex, so kann die Triangulierung durch Bildung eines *Triangle Fan* realisiert werden. Sind in dem durch die Gitterzelle gebildeten Polygon lediglich zwei Eckpunkte der Umrisslinie enthalten, so ist dieses Polygon auf jeden Fall konvex, d.h. der Test auf die konvexe Eigenschaft kann eingespart werden, was nochmals zur Reduktion der Rechenzeit beiträgt.
Beim Erstellen der Eckpunktliste einer Gitterzelle ist darauf zu achten, dass die entsprechenden Eckpunkte aus der Umrisslinie und aus der inneren Gitterpunktmenge in konsistenter Umlaufrichtung in die Liste eingefügt werden. Problematisch ist hierbei die Tatsache, dass die Umlaufrichtung der im ATKIS modellierten Umrisslinien nicht fest vorgegeben ist. Ein sinnvolles Vorgehen ist hier, in Abhängigkeit der Lage der Anfangs- und Endpunkte des Umrissliniensegmentes die Reihenfolge der Gitterpunkte zu bestimmen. Abbildung 28 zeigt die möglichen auftretenden Situationen.
Durch die vielfältige Form der Umrisslinien ergibt sich eine Vielzahl von Sonderfällen, welche während der Triangulierung berücksichtigt werden müssen. Komplikationen ergeben sich, wenn in einer Gitterzelle mehrere Umrissliniensegmente

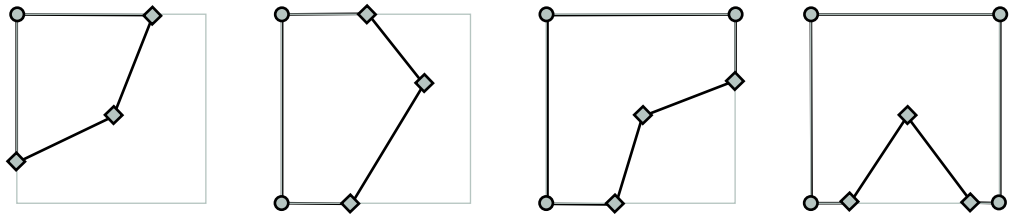


Abbildung 28: Unterschiedliche Anzahl der Gitterpunkte

auftreten. Dies führt dann dazu, dass das aus Umrissliniensegmenten und Gitterpunkten gebildete Polygon in einem nicht einfachen Polygon resultiert, welches für die Triangulation durch den Ear Clipping Algorithmus nicht verarbeitbar ist. Abbildung 29 veranschaulicht die Problematik für zwei Segmente innerhalb ei-

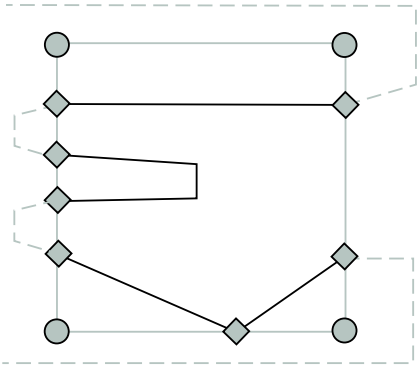


Abbildung 29: Zwei Segmente pro Gitterzelle

ner Gitterzelle. Der Verlauf der restlichen Umrisslinie außerhalb der Gitterzelle ist durch die gestrichelte Linie angedeutet. Die zwei entstehenden, seperaten Polygone sind grau unterlegt. Diese Polygone müssen, um der Forderung nach einfachen Polygonen Rechnung zu tragen, getrennt trianguliert werden. Der dargestellte Fall zeigt hier lediglich zwei Umrissliniensegmente in einer Gitterzelle, die Handhabung des allgemeinen Falls mit theoretisch beliebig vielen Segmenten in einer Gitterzelle entwickelt sich sehr schnell zu einer komplexen Problemstellung. In der Praxis hat sich gezeigt, dass sich sehr selten mehr als zwei Umrissliniensegmente innerhalb einer Gitterzelle befinden, jedoch konnte auf eine Implementation der Sonderfallbehandlung nicht verzichtet werden, da eine konsistente und lückenlose Triangulation die Grundlage des Terrainmodells darstellt. Abbildung 30 zeigt solch einen komplexeren Fall. In diesem Beispiel befinden sich die vier Umrissliniensegmente $(0, 1)$, $(2, 3)$, $(4, 5)$ und $(5, 6)$, innerhalb der Gitterzelle. Zur Vereinfachung werden hier jeweils die Start- und Endpunkte eines Umrissliniensegmentes angegeben. Ferner befinden sich alle vier Gitterpunkte der Zelle inner-

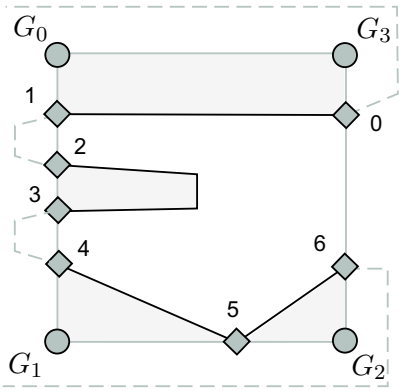


Abbildung 30: komplexere Anordnung von Umrisssegmenten

halb der Umrisslinie, müssen also in die Triangulation mit einbezogen werden. Die Gitterzelle wird durch diese Anordnung der Umrissliniensegmente in vier separat zu triangulierende Flächen unterteilt. Um solche separat zu triangulierenden Flächen zu erkennen, muss die Anordnung der Segmente auf zwei unterschiedliche Ausprägungen hin untersucht werden:

- Segmente, welche einen oder mehrere Gitterpunkte "abschneiden", sind Teil einer separat zu triangulierende Fläche
- Segmente, deren Start- und Endpunkt sich auf der gleichen Gitterlinie befinden, bilden genau dann ein separat zu triangulierendes Flächenstück, wenn sich mindestens einer der Gitterpunkte auf dieser Gitterlinie innerhalb des Flächenumrisses befindet und von einem anderen Segment abgetrennt wird.

In Abbildung 30 werden die Gitterpunkte G_0 und G_1 von Umrissliniensegment $(0, 1)$ abgetrennt. Hieraus ergibt sich ein aus den Punkten des Umrissliniensegmentes sowie den Gitterpunkten G_0, G_3 bestehendes Polygon. Start- und Endpunkt des Segmentes $(2, 3)$ befinden sich auf einer Gitterlinie der Zelle. Da die sich auf derselben Gitterlinie befindenden Gitterpunkte G_0 und G_1 innerhalb der Umrisslinie liegen und von Segment $(0, 1)$, bzw. Segment $(4, 5)$ abgetrennt werden, bilden die Punkte des Segmentes $(2, 3)$ eine eigens zu triangulierende Fläche. Umrisslinienpunkt 5 stellt eine Besonderheit dar, da er gleichzeitig Start- und Endpunkt von zwei Segmenten ist.
Der Algorithmus, nach welchem die Gitterzelle in einzelne Flächen zerlegt wird, ermittelt zunächst für jede der vier Gitterlinien die darauf liegenden Punkte. Dies können sowohl Start- oder Endpunkte der sich in der Gitterzelle befindenden Umrissliniensegmente als auch Eckpunkte der Gitterzelle sein. Zu jedem Punkt wird noch eine Reihe von Informationen gespeichert, welche in der Klasse `GridLineInfo` verwaltet werden. Diese Klasse enthält unter anderem als Datenfelder:

- Die x, y, z Koordinaten des Punktes
- Die Indexnummer des Punktes
- Eine Referenz auf dasjenige Umrissliniensegment, für welches der Punkt Startpunkt ist
- Eine Referenz auf dasjenige Umrissliniensegment, für welches der Punkt Endpunkt ist

Die Punkte werden für jede Gitterlinie in aufsteigender Reihenfolge sortiert. Abbildung

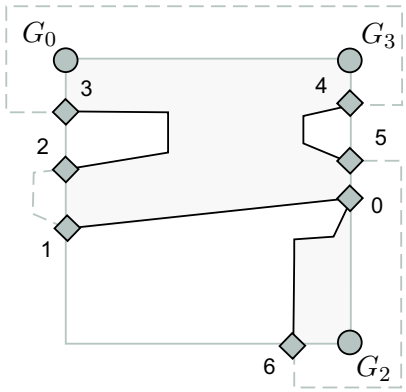


Abbildung 31: Zerlegung der Gitterzelle

31 zeigt eine zu triangulienede Gitterzelle. Für die sortierten Punkte auf den Gitterlinien ergeben sich somit die vier Punktmengen

$$\begin{aligned} L_{Xmin} &= \{G_0, 3, 2, 1\} \\ L_{Xmax} &= \{G_3, 4, 5, 0, G_2\} \\ L_{Zmin} &= \{G_0, G_3\} \\ L_{Zmax} &= \{6, G_2\} \end{aligned}$$

Nun wird in jeder der vier Punktmengen nach Segmenten, welche Gitterpunkte abschneiden, gesucht. Hierfür werden die für jeden auf der Gitterlinie liegenden Punkt gespeicherten Umrissliniensegmentreferenzen herangezogen. Jedes dieser abschneidenden Segmente bildet nun eine Kante einer separaten Fläche. In Abbildung 31 sind dies die Umrissliniensegmente $(0, 1)$ und $(6, 0)$. Zu jedem dieser Abschneidenden Segmente müssen nun, zusätzlich zu den abgeschnittenen Gitterpunkten, diejenigen Umrissliniensegmente, welche sich zwischen dem abschneidenden Segment und dem abgeschnittenen Gitterpunkt befinden, miteinbezogen werden. Im Falle des abschneidenden Segmentes sind dies die Segmente $(4, 5)$ und $(2, 3)$. Dabei ist stets darauf zu achten, dass die Segmente in korrekter Reihenfolge zusammengesetzt werden. Wurde nun eine solche separate Fläche trianguliert, so

können die beteiligten Segmente aus den Datenstrukturen entfernt werden. Hierbei ist insbesondere darauf zu achten, das die in der Klasse `GridLineInfo` verwalteten Informationen in einem konsistenten Zustand gehalten werden. Nachdem so alle durch abschneidende Segmente erzeugten Flächen entfernt wurden, kann ein weiterer Sonderfall auftreten, welcher in einer nicht-einfachen Polygongeometrie resultiert. Abbildung 32 veranschaulicht dies. Daher müssen in einem

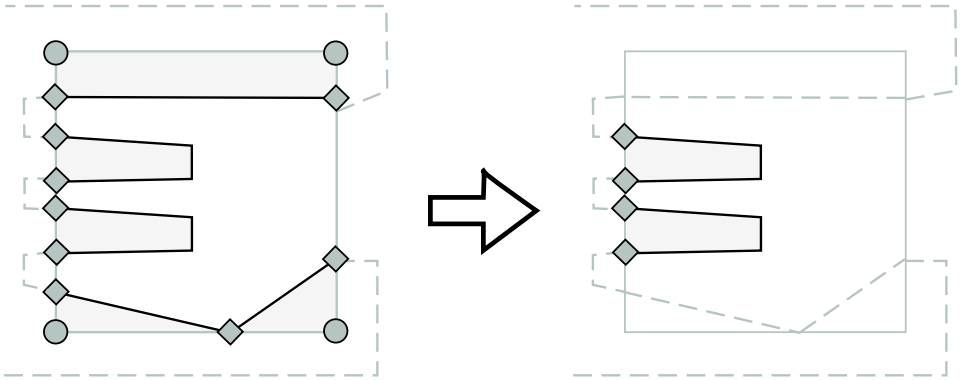


Abbildung 32: Entfernung separater Flächen

weiteren Verarbeitungsschritt alle übrig gebliebenen Segmente, deren Start- und Endpunkt auf der selben Gitterlinie liegen, einzeln trianguliert werden.
Die Triangulierung der innenliegenden Gitterpunkte gestaltet sich wesentlich einfacher, doch auch hier müssen einige Sonderfälle in Betracht gezogen werden. Um die innen liegenden Gitterzellen zu triangulieren, werden für jeden innen liegenden Gitterpunkt G_0 diejenigen Gitterpunkte gesucht, welche rechts von G_0, unterhalb von G_0 und rechts-unterhalb von G_0 liegen. G_0 bildet also die linke obere Ecke der zu triangulierenden Gitterzelle. Für die Suche muss die Liste der inneren Gitterpunkte von G_0 aus durchlaufen werden. Aufgrund der Reihenfolge, in welcher die inneren Gitterpunkte abgelegt werden (vgl. Abbildung 21), kann der Durchlauf abgebrochen werden, sobald die z-Koordinate eines betrachteten Gitterpunktes größer als die z-Koordinate von G_0 zuzüglich der Gitterweite ist. Wurden alle notwendigen Punkte zur Bildung einer Gitterzelle gefunden, so kann die Gitterzelle in zwei Dreiecke zerlegt werden, sofern sich kein Eckpunkt der Umrisslinie innerhalb der Gitterzelle, bzw. auf einer Gitterlinie dieser Zelle befindet. Abbildung 33 veranschaulicht dies. In diesem Beispiel kann die Gitterzelle, bestehend aus den Gitterpunkten $(0, 1, 3, 4)$ direkt in zwei Dreiecke zerlegt werden, da diese Zelle frei von Punkten der Umrisslinie ist. Dies ist bei den grau unterlegten Zellen $(3, 4, 7, 8)$ und $(5, 6, 9, 10)$ nicht der Fall. Würde z.B. der einzelne Punkt der Umrisslinie auf der oberen Gittelinie von Zelle $(5, 6, 9, 10)$ nicht berücksichtigt, so würde dies an dieser Stelle der Triangulation zum Auftreten eines *T-Vertex* mit all den damit verbundenen Problemen bezüglich Beleuchtung und Texturie-

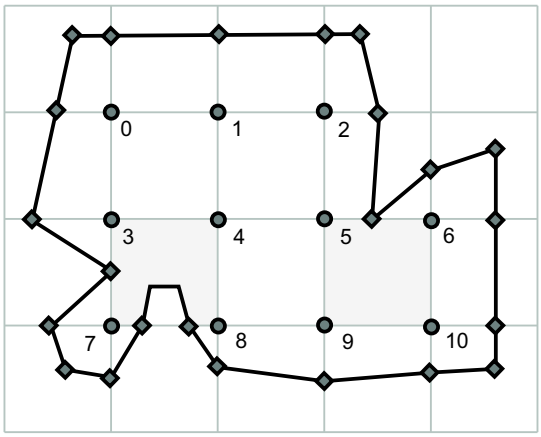


Abbildung 33: Triangulierung innerer Gitterzellen

rung führen. Weiterhin kann es, wie es bei Gitterzelle $(3, 4, 7, 8)$ der Fall ist, zu "Einstülpungen" der Umrisslinie in die Gitterzelle kommen. Diese ins Innere der Gitterzelle ragenden Segmente fallen letztendlich unter die Triangulierung der Umrissliniensegmente, daher muss eine Gitterzelle mit einem solchen Segment von der Triangulierung der inneren Gitterpunkte nicht mehr berücksichtigt werden. Da T-Vertices und Einstülpungen zusammen fallen können, wie es in Gitterzelle $(3, 4, 7, 8)$ des Beispiels zu sehen ist, und die Erkennung von T-Vertices in den Bereich der Triangulation der inneren Gitterzellen fällt, ist es notwendig, diesen Schritt vor der Triangulation der Umrisslinienpunktmenge durchzuführen, und die entsprechenden einstülpenden Segmente aus den Datenstrukturen zu entfernen. Zu beachten ist weiterhin, dass die Eckpunkte der Umrissline exakt auf einen Gitterpunkt fallen können, was in Abbildung 34 ersichtlich wird. Die grau unter-

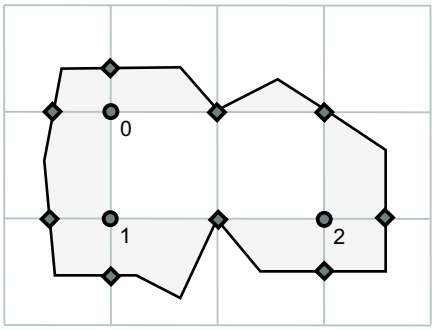


Abbildung 34: Umrisslinie auf Gitterzellenpunkten

legten Flächen zeigen die Umrisspunktmenge. Die Suche nach innen liegenden Gitterzellen würde hier keinen Erfolg liefern, da die dafür notwendigen Punkte in der Liste der innen liegenden Gitterpunkte fehlen. Dies resultiert aus dem Algo-

rithmus, welcher die innen liegenden Gitterpunkte ermittelt (vgl. Abschnitt 4.5.1). Dieser klassifiziert Punkte, welche sich genau auf der Umrisslinie des Polygons befinden, als ausserhalb liegend, folglich werden diese Punkte nicht in die Liste mitaufgenommen. Dieses Vorgehen ist notwendig, um die redundante Speicherung von Punkten zu vermeiden, da ansonsten der Punkt sowohl in der Liste der innenliegenden Gitterpunkte, als auch in der Liste der Umrisslinienpunkte vorkommen würde. Eine redundante Speicherung würde zwar die Behandlung dieses Sonderfalls überflüssig machen, jedoch wurde dieses Vorgehen im Hinblick auf die dauerhafte Speicherung des triangulierten Landschaftsmodells und die großen Datenmengen als nicht praktikabel bewertet.
Abschließend soll an dieser Stelle das grundsätzliche Vorgehen zur Erstellung einer triangulierten, Flächenförmigen Geometrie zusammengefasst werden. Als Eingabedaten wurden die im ATKIS-Basis DLM gegebene Umrisslinie der Fläche, sowie die im DGM enthaltene Höheninformation, welche in der Form von regelmäßig angeordneten Abtastpunkten gegeben ist, vorgestellt. Folgende Schritte konvertieren diese Eingabedaten in eine triangulierte, dem Höhenprofil des Geländes folgende Fläche:

1. Erweiterung der Umrisslinie um die Schnittpunkte mit den Gitternetzlinien. Dies bildet die Umrisslinie auf das regelmäßige Gitter des Höhenmodells ab.
2. Hinzufügen der Gitterpunkte, welche innerhalb der Umrisslinie liegen.
3. Triangulation der so entstandenen Punktmenge. Dies erfolgt in zwei Schritten:
   (a) Triangulation der inneren Punktmenge
   (b) Triangulation der äusseren Punktmenge

5 Prozedurale Texturen

5.1 Noise

Als *Noise (Rauschen)* wird eine stochastische, irreguläre Funktion einer oder mehrerer gegebener Eingabegrößen bezeichnet. In [EMP+03] wird das Rauschen durch folgende Eigenschaften charakterisiert:

- Wiederholbare Pseudo-Zufallsfunktion
- Bekannter Wertebereich (normalerweise $[-1, 1]$)
- Bandbeschränkt, höchste Frequenz ca. 1
- Keine offensichtlichen Periodizitäten oder reguläre Muster
- Rotatorisch und translatorisch invariant

Der Aspekt der Wiederholbarkeit bedeutet, dass die Zufallsfunktion bei gleichbleibenden Eingabewerten stets den gleichen, wenn auch zufälligen Wert liefert. Betrachtet man die Zufallsfunktion als eine Funktion im mathematischen Sinne, bei der ein Argument auf genau einen Funktionswert abgebildet wird, so lässt sich dieser scheinbare Widerspruch auflösen. Hier zeigt sich sehr deutlich der Unterschied zu einem komplett *unvorhersehbaren* Prozess, bei welchem der Aspekt der Wiederholbarkeit nicht vorhanden wäre. Bei einer Zufallsfunktion hingegen ist der Funktionswert für jeden Eingabewert stets gleich, jedoch lässt sich der tatsächliche Wert für ein gegebenes Argument im Gegensatz zu einer mathematischen Funktion nicht berechnen; er kann praktisch nur durch „ausprobieren“ ermittelt werden.

5.1.1 Zufallszahlengenerator

Als Zufallszahlengenerator oder Zufallsgenerator bezeichnet man ein Verfahren zur Erzeugung zufälliger Zahlenfolgen. Grundsätzlich lassen sich hier zwei Arten von Zufallsgeneratoren unterscheiden:

- deterministische Zufallsgeneratoren
- nicht-deterministische Zufallsgeneratoren

Ein nicht-deterministischer Zufallszahlengenerator erzeugt im Gegensatz zum deterministischen Zufallszahlengenerator unter gleichen Ausgangsbedingungen stets unterschiedliche Zufallswerte. Als nicht-deterministischer Zufallsgenerator werden meist physikalische Prozesse wie der Zerfall radioaktiver Isotope verwendet, letztendlich lässt sich jedoch jeder nicht-deterministische Prozess zur Erzeugung von Zufallszahlen verwenden. Beispielsweise liese sich die Anzahl von Autos einer bestimmten Farbe auf einer Kreuzung zu einem gegebenen Zeitpunkt zur

Erzeugung von Zufallszahlen verwenden. Ob dieser Prozess komplett zufällig abläuft oder oder lediglich auf Grund komplexer kausaler Verkettungen zufällig erscheint, ist letztendlich zur Erzeugung von Zufallszahlen unerheblich. Die Qualität der Zufälligkeit lässt sich mittels statistischer Methoden überprüfen. Die durch Computerprogramme realisierten Zufallsgeneratoren arbeiten stets deterministisch, da der zu Grunde liegende Programmablauf deterministisch ist. Es existieren verschiedene Algorithmen zur Realisierung eines Zufallszahlengenerators.
Unter http://www.boost.org/libs/random/random-generators.html sind die in der Boost Bibliothek verwendeten Zufallgeneratoren aufgelistet. Ein relativ einfacher deterministischer Zufallsgenerator ist der *LCRNG (Linear Congruential Random Number Generator)*. Der LCRNG berechnet Zufallszahlen durch

$$x_{i+1} = (ax_i + b) \ mod \, m$$

Bedingt durch die Modulo Operation erzeugt dieser Zufallsgenerator Zahlen zwischen 0 und m. Die Faktoren a und b müssen zu einem bestimmten m so gewählt werden, dass die Bedingung $x_{i+1} \neq x_i$ stets erfüllt ist. Wählt man z.B. $a = 4$, $b = 3$, $m = 16$ und als Startwert $x_0 = 5$ so ergibt sich die Sequenz

$$\begin{array}{rcccrcr} x_1 & = & (4 \cdot 5 + 3) & mod\, 16 & = & 7 \\ x_2 & = & (4 \cdot 7 + 3) & mod\, 16 & = & 15 \\ x_3 & = & (4 \cdot 15 + 3) & mod\, 16 & = & 15 \\ \vdots & & & & & & \end{array}$$

Um die wiederholte Erzeugung der gleichen Zahl bedingt durch $x_{i+1} = x_i$ zu vermeiden, müssen a und b gemäß den in [Unk] beschriebenen Bedingungen gewählt werden:

1. b und m besitzen außer 1 keinen gemeinsamen Teiler

2. k ist ein Vielfaches jedes Primfaktors von m

3. k ist ein Vielfaches von $4 \Leftrightarrow m$ ist ein Vielfaches von 4

4. $a = k + 1$

Wählt man $a = 9$, $b = 5$ und $m = 16$, so ergibt sich für $x_0 = 5$ die Sequenz

$$\begin{array}{rcll} x_1 & = & (9 \cdot 5 + 5) & mod\ 16 = 2 \\ x_2 & = & (9 \cdot 7 + 2) & mod\ 16 = 7 \\ x_3 & = & (9 \cdot 7 + 5) & mod\ 16 = 4 \\ x_4 & = & (9 \cdot 4 + 5) & mod\ 16 = 9 \\ \vdots & & & \\ x_{17} & = & (9 \cdot 5 + 5) & mod\ 16 = 2 \\ x_{18} & = & (9 \cdot 7 + 2) & mod\ 16 = 7 \\ x_{19} & = & (9 \cdot 7 + 5) & mod\ 16 = 4 \\ \vdots & & & \end{array}$$

Man sieht hier, dass sich ab einer gewissen Zahl von Iterationen die Zufallsfolge wiederholt, daher werden Zufallsgeneratoren diesen Typs auch als *periodische Zufallsgeneratoren* bezeichnet. Der Algorithmus, wecher der `rand()` Funktion der ANSI-C Standardbibliothek zu Grunde liegt [11], ist ein LCRNG mit $a = 1103515245$, $b = 12345$, $m = 2^{31}$ und $x_0 = 12345$.
Die durch deterministische Zufallsgeneratoren erzeugten Zahlen werden oft als *Pseudo-Zufallszahlen* bezeichnet.

5.1.2 Weißes Rauschen

In der Stochstik bezeichnet der Begriff *Weißes Rauschen* einen diskreten stochastischen Prozess ohne jede Korrelation der Zufallsvariablen, sowohl räumlich als auch zeitlich betrachtet. Dabei ist die Varianz des Prozesses konstant, der Erwartungswert ist Null. Weißes Rauschen kann mittels eines deterministischen Zufallsgenerators approximiert werden. Dies geschieht beispielsweise bei einem LCRNG durch die Belegung des Startwertes x_0 mit einem zufälligen Wert. Ein solcher zufälliger Wert kann z.B. durch die Systemzeit, der Position des Mauszeigers oder der Position des Festplatten-Lesekopfes angenähert werden.

5.1.3 Lattice Noise Funktionen

Zur Erzeugung einer der in Kapitel 5.1 aufgelisteten Eigenschaften entsprechenden Zufallsfunktion können sogenannte *Lattice*[12] *Noise* Verfahren zum Einsatz kommen. Die Rauschfunktion kann dabei durch beliebig dimensionierte Argumente parametrisiert werden. Die Berechnung des Funktionswertes einer n-dimensionalen Rauschfunktion wird mittels eines n-dimensionalen *Integer-Gitters* realisiert. Dabei handelt es sich um ein Gitternetz, dessen Gitterpunkte an den jeweils ganzzahli-

[11] http://random.mat.sbg.ac.at/ charly/server/node3.html
[12] engl. „Gitter“

gen Koordinaten angeordnet sind. Jedem dieser Gitterpunkte wird eine durch die Koordinaten des Gitterpunktes parametrisierte, wiederum beliebig dimensionierte Pseudo-Zufallszahl zugeordnet. Den Funktionswert für ein Argument $p \in \mathbf{R}^n$ erhält man durch Interpolation der Werte aus den p umgebenden Gitterpunkten. Für $p \in \mathbf{R}^2$ müssen demnach vier Gitterwerte interpoliert werden, für $p \in \mathbf{R}^3$ sind acht Werte zu interpolieren. Das verwendete Interpolationsverfahren kann beliebig gewählt werden, hat jedoch maßgeblichen Einfluss auf die resultierende Rauschfunktion. Häufig angewendete Verfahren sind

- Lineare Interpolation
- Cosinus Interpolation
- Kubische Interpolation
- Polynominale Interpolation

Um den Punkt p zwischen den Gitterpunkten zu interpolieren, wird der Interpolationsfaktor t für jede der n Dimensionen von p durch den Nachkommaanteil berechnet. Im zweidimensionalen Fall müssen somit die Interpolationsfaktoren t_x und t_y aus den Nachkommaanteilen der X- und Y-Koordinaten von p verwendet werden. Die Interpolation zwischen den Werten a und b anhand des Parameters t erfolgt nun bei der linearen Interpolation durch

$$I(a, b, t) = a \cdot (1 - t) + t \cdot b$$

Bei der linearen Interpolation wird demnach der Interpolationsfaktor t unverändert zur Berechnung des Ergebnisses verwendet. Im Gegensatz dazu erfährt bei den anderen erwähnten Interpolationsverfahren der Interpolationsfaktor t spezielle Transformationen. So erhält man bei der Cosinus-Interpolation den Ergebniswert durch

$$I(a, b, t) = a \cdot \Big(1 - \big((1 - \cos(t\pi)) \cdot 0.5\big)\Big) + b \cdot \Big(\big(1 - \cos(t\pi)\big) \cdot 0.5\Big)$$

Das polynominale Interpolationsverfahren wurde von Ken Perlin in [Per99] vorgestellt. Der Interplationswert berechnet sich mittels

$$I(a, b, t) = a + \big(3t^2 - 2t^3\big) \cdot (b - a)$$

Das verwendete Polynom $3t^2 - 2t^3$ bildet ein gegebenes Argument im Intervall $[0, 1]$ auf einen Ergebniswert im Intervall $[0, 1]$ ab und wird von Perlin als *„Ease Curve“* bezeichnet. Im Gegensatz zu den bisher erwähnten Interpolationsverfahren werden bei der kubischen Interpolation zusätzlich zu den zu interpolierenden Punkten a und b zwei weitere Stützpunkte a' und b', welche jeweils den Vorgänger von a bzw. den Nachfolger von b repräsentieren, hinzugezogen. Die Berechnung

des Ergebniswertes erfolgt durch

$$I(a', a, b, b', t) = \big((b' - b) - (a' - a)\big)t^3 + \big(2(a' - a) - (b' - b)\big)t^2 + (b - a)t + a$$

Es existieren innerhalb der Gruppe der Lattice-Noise Verfahren verschiedene Ausprägungen der den Gitterpunkten zugeordneten Pseudo-Zufallswerten und der Algorithmen zur Berechnung der Noise-Funkton. Die unterschiedlichen Verfahren wirken sich direkt auf die resultierende Rauschfunktion aus, sowohl in qualitativer Hinsicht, als auch hinsichtlich des Laufzeitverhaltens des Algorithmusses. In [EMP+03] und [Eri04] wird ein ausführlicher Überblick über die Vor- und Nachteile verschiedener Noise Algorithmen gegeben. Über die gängigsten Methoden sei hier eine zusammenfassende Übersicht gegeben:
Bei der **Value Noise** Methode werden den Gitterpunkten skalare Pseudo-Zufallswerte im Intervall $[-1, 1]$ zugeordnet. Die Gitterpunktwerte können direkt zum Ergebniswert interpoliert werden. Die resultierende Ergebnisfunktion hängt ausschließlich von dem gewählten Interpolationsverfahren ab.
Der **Lattice Convolution Noise** Algorithmus basiert auf dem Lattice Value Noise Verfahren, die Interpolation der Gitterpunktwerte erfolgt hier jedoch über eine diskrete Faltung der Gitterwerte, welche den Eingabepunkt p in einem definierten Radius umgeben. Als Faltungskern wird in [EMP+03] ein Catmull-Rom Filter vorgeschlagen. Lattice Convolution Noise soll die beim Lattice Value Noise auftretende, durch das regelmäßige Gitter bedingte Artefaktbildung unterdrücken. Durch die im Vergleich zum Lattice Value Noise Algorithmus größere Anzahl der in die Berechnung einfließenden Gitterpunkte ist dieses Verfahren in der Berechnung aufwändiger als der Lattice Value Noise Algorithmus.
Sparse Convolution Noise basiert auf dem Prinzip des Lattice Convolution-Noise Verfahrens, jedoch sind die Zufallszahlen in den Gitterzellen zufällig verteilt und liegen nicht mehr exakt auf den Integer- Werten des Gitters. Auch hier wird die Zufallsfunktion wieder durch Faltung mit einem geeigneten Kern berechnet. Sparse Convolution-Noise zeichnet sich durch einen relativ hohen Berechnungsaufwand aus. Es besteht jedoch die Möglichkeit, durch Modifikation des Faltungskernes bei Lattice Convolution-Noise und Sparse Convolution-Noise eine gewisse Kontrolle über das resultierende Spektrum der Noise Funktion zu erreichen.
Das **Gradient Noise** Verfahren ordnet jedem Gitterpunkt statt eines einzelnen skalaren Wertes einen $\mathbf{R}^n$ Vektor, den sog. *Gradientenvektor*, zu. Der Gradientenvektor $\vec{g}$ ist ein Einheitsvektor und wird als Pseudo-Zufallswert durch die Koordinaten des dazugehörigen Gitterpunktes parametrisiert. Um den Wert der Zufallsfunktion an der Stelle p zu berechnen, werden zunächst die Skalarprodukte $\vec{p} - \vec{P_i} \bullet g_i$ berechnet. Zwischen den resultierenden 2^n skalaren Werten kann nun der Wert der Zufallsfunktion interpoliert werden. Abbildung 35 veranschaulicht das Verfahren für den zweidimensionalen Fall. Ein Effekt der Grid-Noise Methode ist,

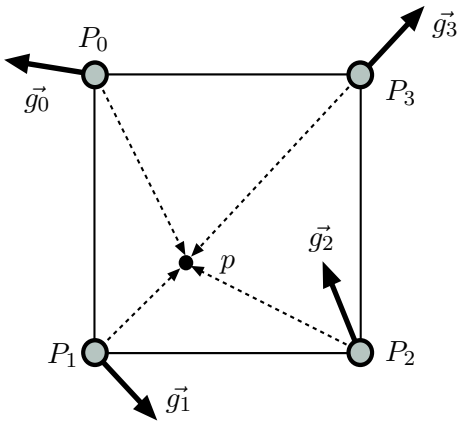


Abbildung 35: Lattice Gradient Noise Verfahren

dass die Zufallsfunktion den Wert Null annimmt, sobald der Eingabewert p auf einem Gitterpunkt liegt. Dies liegt zum einen an dem Vektor $\vec{p} - \vec{P_i}$, welcher für $p = P_i$ dem Nullvektor entspricht, zum anderen an dem Interpolationsfaktor t, welcher als Nachkommaanteil für ganzzahlige Komponenten von p den Wert 0 annimmt. Der Vorteil an diesem Verfahren ist die Tatsache, dass der so berechnete Wert der Rauschfunktion an der Stelle p nicht mehr ausschließlich von der Interpolation der Gitterpunktwerte abhängt, sondern dass auch der Abstand von p zu einem Gitterpunkt P in die Berechnung mit einfließt. Dies kann die durch das Gitter bedingten Artefakte unterdrücken.

Value-Gradient Noise umgeht das Problem des regulären Null-Musters von Gradient Noise, indem der Value-Noise Algorithmus mit dem Gradient Noise Algorithmus kombiniert wird. Eine einfache Kombinationsmöglichkeit besteht in der Bildung einer gewichteten Summe von Value-Noise und Gradient Noise. Eine anspruchsvollere Methode basiert auf einer Kubischen Hermite-Interpolation (vgl. [EMP+03]).

5.2 Perlin Noise

Die in der Computergraphik wohl bekannteste Noise Funktion ist das von *Ken Perlin*[13] entwickelte und nach ihm benannte Perlin Noise. Das Verfahren basiert grundsätzlich auf der Überlagerung von Rauschfunktionen mit unterschiedlicher Frequenz und Amplitude. Zur Vereinfachung soll dieses Prinzip an einer Gruppe von unterschiedlichen Sinusfunktionen verdeutlicht werden. Abbildung 36 zeigt die Funktionsplots von vier Sinusfunktionen, welche jeweils Schwingungen höherer Frequenz bei gleichzeitig geringerer Amplitude darstellen. Die einzelnen

[13] http://mrl.nyu.edu/ perlin

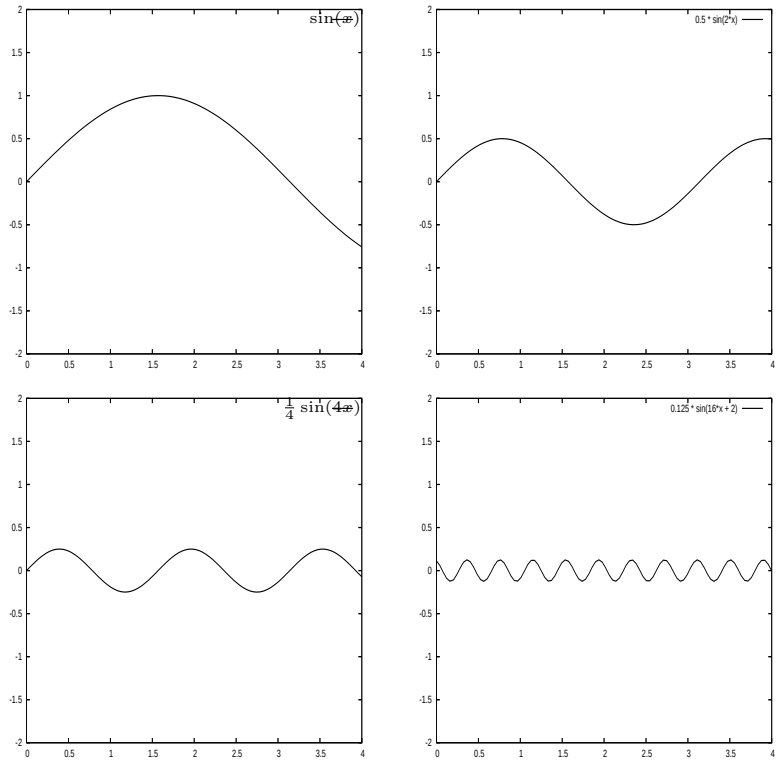


Abbildung 36: Sinus Funktionen verschiedener Frequenz und Amplitude

Funktionen sind jeweils durch eine gleichmäßige Periode charakterisiert. Werden diese vier Funktionen nun aufsummiert, so ergibt sich im Vergleich zu den Ausgangsfunktionen ein gänzlich anderes Bild. Der Funktionsgraph in Abbildung 37 zeigt, dass die Summe der einzelnen Funktionen Charakteristika der aufsummierten Funktionen aufweist. So zeigt das Schaubild in Abbildung 37, dass das Frequenzspektrum der Funktion die Frequenzen aller vier Ausgangsfunktionen beinhaltet, die niederfrequente Grundschwingung mit großer Amplitude ist ebenso zu erkennen wie die hochfrequente Schwingung mit geringer Amplitude. Das Aufsummieren von Rauschfunktionen unterschiedlicher Frequenz und Amplidtude hat auf die resultierende Ergebnisfunktion die gleichen Auswirkungen wie im Fall der Summenbildung von Sinusfunktionen. Abbildung 38 zeigt vier Graustufenbilder, welche aus einer 2D Noise Funktion mit Frequenz 2^i erzeugt wurden. Das Ergebnis der Summe der mit jeweils $\left(\frac{1}{2}\right)^i$ gewichteten Einzelbilder ist in Abbildung 39 zu sehen.

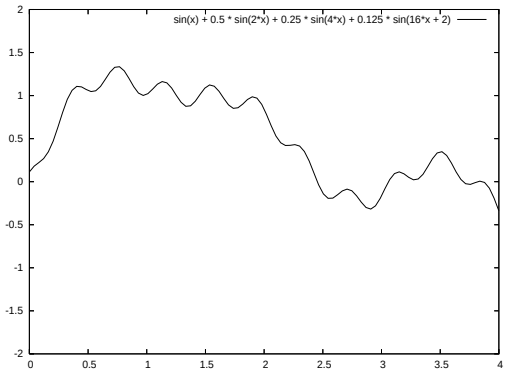


Abbildung 37: Summe der Sinus Funktionen

5.2.1 Steuerung der Noise Funktion

Obwohl die Noise Funktion prinzipiell von zufälligen Faktoren bestimmt ist und somit das letztendliche Ergebnis nicht vorhersehbar ist, ist ein gewisser Einfluss auf die Charakteristik des Ergebnisses wünschenswert. So kann man, gerade im Kontext der Texturgenerierung, eine Vielzahl unterschiedlicher Ergebnisbilder erzeugen. Eine intuitive Steuerung durch wenige Parameter, mit denen das Ergebnis relativ genau abgeschätzt werden kann, ist hier von Vorteil. Die in dieser Arbeit implementierte zweidimensionale Noise Funktion $F(x, y)$ lässt sich durch die Parameter

- Größe in X Richtung, s^x
- Größe in Y Richtung, s^y
- Frequenz f
- Amplitude a

steuern. Für die Erzeugung einer Textur dient als Eingabewert die (x, y)-Koordinaten des zu setzenden Pixels. Der Ergebniswert $r_{x,y}$ berechnet sich durch

$$r_{x,y} = F\left(x \cdot s^x \cdot f, y \cdot s^y \cdot f\right) \cdot a$$

Für die Summenbildung müssen für jede einzelne Noisefunktion die Parameter f und a angegeben werden. Die beiden Werte können für jede Einzelfunktion durch einen einzigen Parameter, der sog. *Persistenz*[14], erzeugt werden. Dabei ergeben sich Amplitude und Frequenz des i-ten Eingabebildes bei gegebener Persistenz

[14] In der engl. Literatur mit „persistence“ bezeichnet

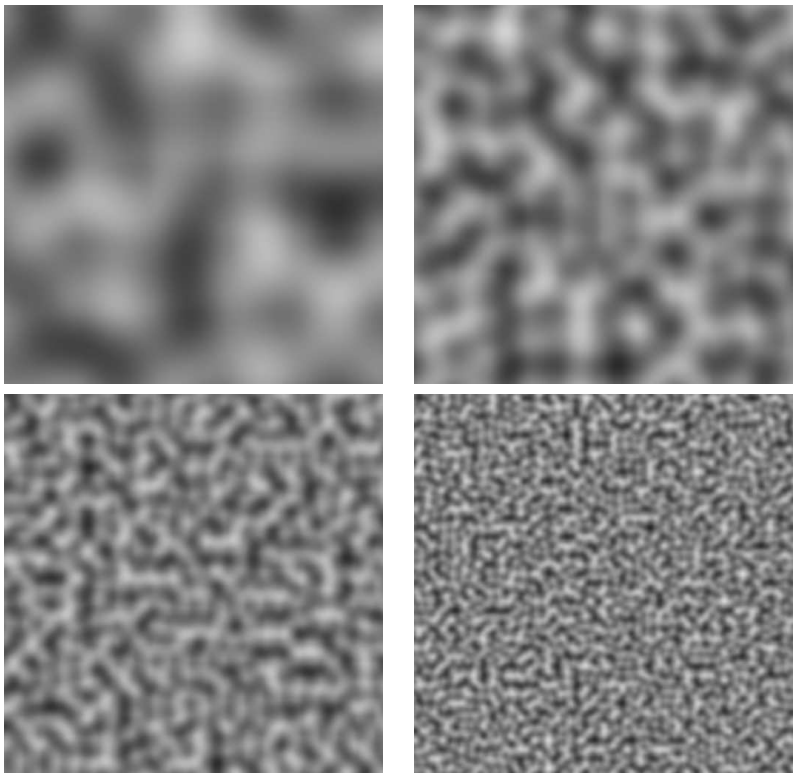


Abbildung 38: 2D Noise Funktion unterschiedlicher Frequenz

p durch

$$\begin{aligned} a_i &= p^i \\ f_i &= k^i \end{aligned}$$

Der konstante Faktor k kann beliebig gewählt werden, er steuert das Verhalten der Frequenz für jede Einzelfunktion. Ist $k > 1$, so erhöht sich die Frequenz für jedes i. Ein gängiger Wert ist $k = 2$, somit verdoppelt sich die Frequenz jeder i-ten Iteration im Vergleich zur Vorgängerfrequenz, es wird somit eine Folge von Oktaven erzeugt. Über die Persistenz lässt sich nun steuern, wie stark die Frequenz f_i in die Gesamtsumme einfließt.

5.2.2 Wertebereich

Eine einzelne Noise-Funktion liefert Werte im Bereich $[-1, 1]$. Zur Erzeugung von Texturen ist es notwendig, die Noise-Werte auf einen definierten Wertebereich, typischerweise $[0, 1]$, abzubilden. Bildet man die Summe aus mehreren Noise Funktionen, so können Werte außerhalb des Wertebereiches $[-1, 1]$ entstehen. Um die

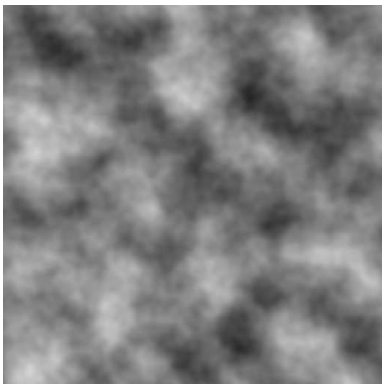


Abbildung 39: Summe der Noise Funktionen

entstehenden Werte auf den Wertebereich $[-1, 1]$ abzubilden, können zwei Ansätze verfolgt werden:

1. Clamping der Werte auf den Wertebereich $[-1, 1]$
2. Skalierung der Werte mittels des maximalen und minimalen zu erwartenden Nois-Wertes

Zur Umsetzung des zweiten Ansatzes müssen das Maximum bzw. Minimum der Summe der Noisefunktionen bekannt sein. Da die Werte der einzelnen Noise-Funktionen mit ihrer jeweiligen zugeordneten Amplitude multipliziert werden (vgl. Kapitel 5.2.1), ist der Maximal- bzw. Minimalwert der Summe

$$min, max = \pm \sum_{i=0}^{N} a_i$$

Im Vergleich zu den geclampten Werten sind die durch die Skalierung erzeugten Werte „geglättet“, die daraus resultierenden Bilder sind, wie in Abbildung 40 zu sehen ist, wesentlich kontrastärmer.

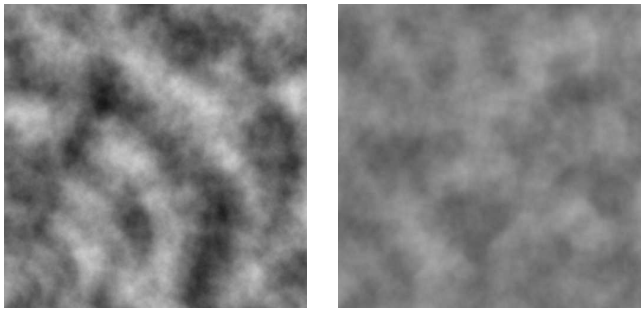


Abbildung 40: Clamping (links) und Gewichtung (rechts)

5.2.3 Implementationsdetails

Die Implementation der Perlin Noise Funktionalität orientiert sich stark an der Originalimplementation[15] von Ken Perlin, sowie an den Programmbeispielen von Hugo Elias ([Eli03]). Von Ken Perlin wurde hauptsächlich die Umsetzung des Gradient-Noise Verfahrens übernommen, von Elias eine Implementation des Value-Noise Algorithmuses.
Grundlegend für den Gardient-Noise Algorithmus von Perlin ist die Berechnung von Zufallsvektoren für jeden Gitterpunkt. Die Gradientenvektoren werden hierbei nicht zur Laufzeit berechnet, vielmehr wird in diesem Ansatz eine Menge von zufälligen Gradientenvektoren vorausberechnet. Um jedem beliebigen Gitterpunkt einen Gradientenvektor zuzuordnen, wird aus der Liste der Gradientenvektoren mittels einer Hash-Funktion und einer Liste, welche zufällig angeordnete Referenzen (die sog. Permutationstabelle) in die Liste der Gradientenvektoren enthält, dem Gitterpunkt zugewiesen.

Listing 3: Hashing in die Permutationstabelle

```
1  P: Permutationstabelle
2  G: Liste der Gradientenvektoren
3  |G| = |P|
4
5  //erzeugt eine Tabelle mit zufällig angeordneten Werten
6  genPermTab(){
7          FOR(i=0; i<|P|; i++){
8                  P[i] = i;
9          }
10
11         FOR(i=0; i<|P|; i++){
12                 k = P[i];
13                 P[i] = P[j = rand() % |G|];
14                 P[j] = k;
15         }
16 }
17
18 //liefert Index in Perumtationstabelle
19 int getIndex(int x, int y){
20         return (x + (P[y % |G|])) % |G|;
21 }
```

Listing 3 zeigt in Pseudocode die Erstellung der Permutationstabelle sowie die Ermittlung des Indexes in der Permutationstabelle in Abhängigkeit der x, y Koor-

[15] http://mrl.nyu.edu/ perlin

dinaten des Gitterpunktes.

5.3 Noise im Kontext von Echtzeitrendering

Perlin Noise wird in der Computergraphik sehr häufig zur prozeduralen Texturerzeugung eingesetzt. Gerade im Bereich der Landschaftsvisualisierung bietet diese Technik große Vorteile, da es mit Perlin Noise möglich ist, Texturen von praktisch unendlicher Größe zu erzeugen und damit sehr große Landschaftsteile ohne erkennbare wiederkehrende Muster zu texturieren. Da die Textur zur Laufzeit erzeugt wird, entfällt jegliches Speichern und somit jeglicher Verbrauch an Speicherkapazitäten. Der Nachteil des Verfahrens ist der relativ hohe Berechnungsaufwand, dies begründet den bisherigen vorrangigen Einsatz von Perlin-Noise in nicht-echtzeitfähigen 3D-Renderingapplikationen. Das Aufkommen leistungsfähiger, programmierbarer Consumer-Graphikhardware hat in den letzten Jahren einige Arbeiten zur Umsetzung von Noise Algorithmen auf der GPU hervorgebracht. In [Eri04] wird ein Überblick über verschiedene Noise Algorithmen und deren Implementierung auf der GPU gegeben. Zusammenfassend lassen sich folgende Eigenschaften einer Umsetzung von Noise Algorithmen auf der Graphikhardware herausarbeiten:

- Die begrenzte Anzahl an Instruktionen und Texturzugriffen der GPU schränkt die Zahl möglicher Algorithmen ein
- Die Algorithmen benötigen eine große Zahl an Texture-Lookups oder mehrere Rendering Passes

Trotz der prinzipiellen Realisierbarkeit von Noise auf der GPU wurde in dieser Arbeit der GPU-basierte Ansatz nicht weiter verfolgt. Die Gründe hierfür waren primär die große Zahl der zu rendernden Geometriedaten, welche einen erhebliche Teil der Leistungsreserven der Graphikkarte in Anspruch nehmen. Ferner sieht das Texturierungssystem die Kombinationsmöglichkeit verschiedener prozeduraler Algorithmen vor, die Texturierung des Terrains mit einer einzelnen Noise Textur wird eher selten verwendet. Durch die Möglichkeit, Noise Texturen zu kacheln nahm der Verbrauch an Speicherkapazität keine derartigen Ausmaße an, welche eine größere Bereitstellung von Rendering-Ressourcen zugunsten des Speicherplatzes gerechtfertigt hätte.

5.4 Der Cellular Algorithmus

Eine weitere bedeutende Klasse von prozeduralen Texturen sind die sog. *Cellular Textures*. Der Name des Verfahrens beschreibt recht gut die Erscheinungsform dieser Texturen. Abbildung 41 zeigt ein Beispiel einer mit diesem Verfahren erzeugten Textur. Ausführliche Beschreibungen dieser Art prozeduraler Texturerzeugung finden sich in [Scont] und [EMP$^+$03]. Der Algorithmus beginnt mit der

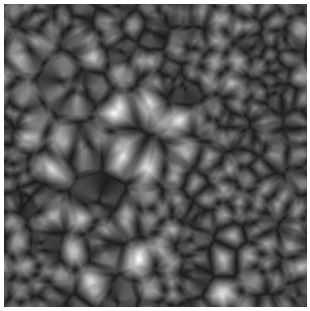
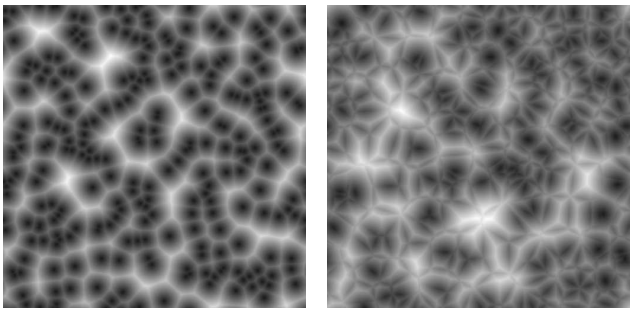


Abbildung 41: Beispiel einer Cellular Textur

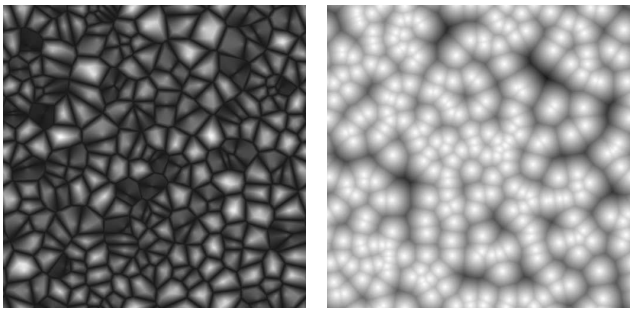
zufälligen Verteilung einer benutzerdefinierten Anzahl an Stützpunkten in dem zu generierenden Bild. Diese Punkte repräsentieren die Zentren der zu erzeugenden Zellen. In einem nächsten Schritt wird für jedes Pixel der Abstand zu jedem Stützpunkt ermittelt, bei k Stützpunkten ergeben sich somit k verschiedene Abstandswerte. Sind $D_{x,y}$ die nach Betrag sortierten Abstandswerte des Punktes (x, y) zu allen Stützpunkten, so repräsentiert $D^1_{x,y}$ den kleinsten Abstandswert, $D^k_{x,y}$ den Größten. Der Wert eines Pixels ermittelt sich nun aus der Verechnung der Abstandswerte miteinander. Dies kann im einfachsten Fall die Zuweisung eines einzelnen Abstandswertes, aber auch eine beliebig komplexe Verrechnung der Abstandswerte sein. So vielfältig wie die Verrechnungsmöglichkeiten der Abstandswerte sind, so vielfältig sind auch die unterschiedlichen visuellen Charakteristika der entstehenden Texturen. In der Regel sind jedoch nur Differenzbildungen und Multiplikation der ersten zwei oder drei Abstandswerte sinnvoll, da sich die Textur bei Verrechnung weiterer Abstandswerte nicht mehr maßgeblich ändert.
Eine weitere Möglichkeit, die Ergebnistextur zu beeinflussen stellt sich in der Wahl des Abstandsmaßes dar. In dieser Arbeit wurden als Abstandsmaße der *Euklidische Abstand* sowie die *Manhattan Distanz* implementiert, die beiden Berechnungsverfahren liefern Texturen deutlich unterschiedlicher Charakteristik. Im folgenden sollen einige Beispielbilder die Auswirkung der verschiedenen Parameter verdeutlichen.

5.5 Colorierung

Die durch die prozeduralen Texturen berechneten Werte $p_{x,y}$ für jedes Pixel befinden sich im Wertebereich $[0, 1]$. Um diese Werte in den RGB-Farbraum abzubilden, wurde der Weg der Interpolation zwischen vom Benutzer definierbaren Farben gewählt. Hierbei können dem System eine beliebige Anzahl von Farbwerten übergeben werden. Jedem Farbwert wird zusätzlich ein Wert k zwischen 0.0 und 1.0 zugewiesen, zwei Farben müssen jeweils dem Wert 0.0 bzw. 1.0 zugeordnet sein (vgl. Abbildung 44). Für jeden Pixel wird nun an Hand des korrespondieren-

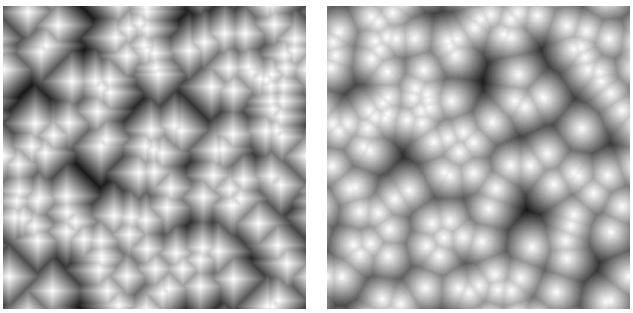


(a) $p_{x,y} = D^0_{x,y}$ (b) $p_{x,y} = D^1_{x,y}$



(c) $p_{x,y} = D^1_{x,y} - D^0_{x,y}$ (d) $p_{x,y} = 1 - D^0_{x,y}$

Abbildung 42: Verschiedene Cellular Texturen



(a) Manhattan Distanz (b) Euklidische Distanz

Abbildung 43: Abstandsmaße

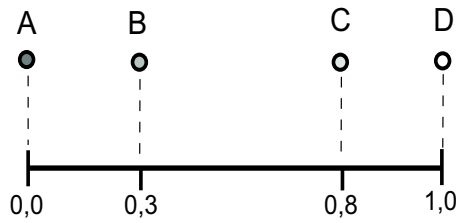
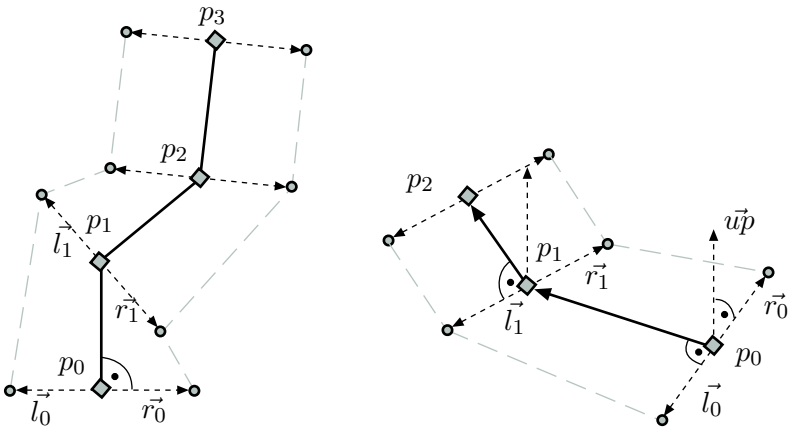


Abbildung 44: Interpolation der Farbwerte

den prozeduralen Wertes $p_{x,y}$ zwischen den Farben mit dem nächst größeren bzw. kleineren k kanalweise linear interpoliert.

6 Prozedurale Infrastruktur

Der Begriff der Infrastruktur beschreibt „alle langlebigen Grundeinrichtungen personeller, materieller und institutioneller Art, die das Funktionieren einer arbeitsteiligen Volkswirtschaft garantieren“[16]. In diesem Kapitel soll der Begriff auf die Gesamtheit der im ATKIS-Basis DLM erfassten Objekte, welche dem Transport dienen, eingeschränkt werden. Dies sind insbesondere topographische Objekte aus den Bereichen Verkehr, Kommunikation und Energieübertragung. Der sicherlich landschaftlich prägendste Aspekt der Infrastrukur ist das Straßennetz. Weiterhin sind Verkehrswege für den Schienenverkehr sowie Einrichtungen zur Übertragung von Energie, hierbei insbesondere in Form von Elektrizität, von den meisten Punkten im Gelände deutlich sichtbar. In dieser Arbeit wird daher schwerpunktmäßig die Erzeugung prozeduraler Geometrie für das Verkehrsnetz, sowohl Straßen- als auch Bahnverkehr, behandelt.
Ausgangsdaten für die Erzeugung der Verkehrswege ist hierbei das im ATKIS-Basis DLM durch linienförmige Objekte modellierte Straßen- und Schienenverkehrsnetz. Hierbei muss eine Umwandlung dieser einfachen Linienzüge in eine komplexe, räumlich ausgedehnte Geometrie vorgenommen werden. Dazu wird aus jedem Basisliniensegment eine Grundfläche berechnet, auf welcher dann die konkrete Verkehrsweggeometrie erzeugt wird. Abbildung 45 zeigt die Vorgehensweise. Die Eckpunkte der Grundfläche werden werden mittels der Punkte p_{i+1}



(a) Aufsicht (b) perspektivisch

Abbildung 45: Erzeugung Verkehrsweggeometrie

und p_i sowie den Vektoren $\vec{l_i}$ und $\vec{r_i}$ berechnet, welche sich aus einem gegebenen

[16] http://de.wikipedia.org/wiki/Infrastruktur

Hilfsvektor $\vec{up}$ durch

$$\begin{aligned} \vec{l_i} &= \vec{up} \times (\vec{p_{i+1}} - \vec{p_i}) \\ \vec{r_i} &= (\vec{p_{i+1}} - \vec{p_i}) \times \vec{up} \end{aligned}$$

berechnen. Der Hilfsvektor $\vec{up}$ wird dabei so gewählt, dass er in entgegengesetzte Richtung des Gravitationsvektors zeigt; er zeigt daher nach "oben". Der Betrag von $\vec{l_i}$ und $\vec{r_i}$ beträgt jeweils die Hälfte der gewünschten Breite des Verkehrsweges.
Da die ATKIS-Daten keine Höheninformation enthalten, müssen hier ähnliche Überlegungen wie bei der Erzeugung flächenförmiger Geometrien aus Umrisslinien angestellt werden. Hauptproblem ist hierbei das Überlagern des zugrunde liegenden Geländereliefs mit der Geometrie der Verkehrswege. Die grundsätzliche Vorgehensweise gestaltet sich daher durch:

- Anpassung der Basislinie an das Geländeprofil
- Erzeugung der eigentlichen Geometrie
- Texturierung zur Hervorhebung von Details

In den folgenden Abschnitten soll auf diese Punkte im Detail eingegangen werden.

6.1 Anpassung an das Geländeprofil

Modellierungsgrundlage der Verkehrsweggeometrie sind Linienzüge, welche die Mittellinie des zu modellierenden Verkehrsweges repräsentieren. Dabei sind auch hier die Höheninformationen gesondert im DGM enthalten. Um die Linie entlang des Geländeprofils zu führen, müssen, wie in Kapitel 4.5.3 beschrieben, die Linienzüge um die Schnittpunkte mit den Gitterlinien des DGMS erweitert werden. Die Überlagerung eines Geländemodells mit Verkehrswegen lässt sich letztendlich auf die Problematik der Einpassung einer Fläche in ein gegebenes Geländemodell reduzieren. Grundsätzlich existieren zwei Möglichkeiten, das Geländemodell mit Flächen zu überlagern:

- Stitching
- Draping

Beim Stitching werden die durch die einzufügende Geometrie überlagerten Dreiecke des Geländemodells entfernt und die eingefügte Geometrie mit dem Geländemodell vermascht. Die Geometrie fügt sich somit nahtlos in das Geländemodell ein. Dies stellt hinsichtlich der Darstellungsqualität sicherlich die optimale Lösung

dar. Der Nachteil des Verfahrens ist, neben dem hohen Berechnungs- und Implementationsaufwand, dass die geometrische Trennung der verschiedenen Landschaftsobjekte verloren geht sowie regelmäßige Geometriestrukturen unterbrochen werden. Möchte man beispielsweise Verkehrswege ausblenden oder in ihrer Lage verändern, würden sich durch die eingebettete Geometrie neue Komplikationen ergeben. Beim Draping wird im Gegensatz zum Stitching die Terraingeometrie nicht verändert, die überlagernden Flächen werden einfach über das Gelände gelegt. Auf Grund der genannten Nachteile des Stitching Verfahrens wurde in dieser Arbeit das Draping Verfahren realisiert.
Bei der Überlagerung eines durch ein Dreiecksnetz modellierten Geländemodells mit zusätzlicher Geometrie, wie z.B. Straßenkörper, muss dafür gesorgt werden, dass die überlagernde Geometrie an keiner Stelle von dem unten liegenden Geländemodell durchdrungen wird. Weiterhin muss dafür gesorgt werden, dass der Übergang zwischen den verschiedenen Geometrieebenen so wenig wie möglich sichtbar wird, was durch eine geeignete Texturierung realisiert werden kann, indem man z.B. die in Abschnitt 6.2.2 gezeigte Übergangstextur verwendet (vgl. Abb. 51, Seitenstreifentextur). Für die Anpassung der Verkehrsweggeometrie an das Geländerelief wird die am Anfang des Kapitels beschriebene Grundfläche eines jeden Segmentes betrachtet. Fügt man nun für die vier Eckpunkte der Grundfläche die aus dem DGM interpolierte Höhe hinzu, so wird die Grundfläche mit hoher Wahrscheinlichkeit nicht mehr lotgerecht auf dem Gelände liegen. Je nach Grad der Neigung ergibt sich somit ein unrealistisches Bild des Straßenverlaufes. Daher ist es notwendig, die Fahrbahnoberfläche am Lot auszurichten. Abbildung

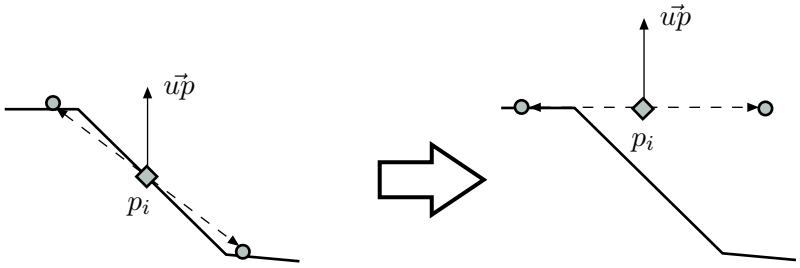


Abbildung 46: Ausrichtung der Fahrbahn

46 zeigt das prinzipielle Vorgehen. Um eine Durchdringung durch das Gelände zu verhindern, muss für die am Lot ausgerichtete Fahrbahn geprüft werden, ob ein Gitterpunkt innerhalb der Grundfläche liegt, und ob sich dieser Gitterpunkt über dem Niveau der Fahrbahnfläche befindet. Ist dies der Fall, so muss die Fahrbahn auf Höhe des Gitterpunktes angehoben werden, was in Abbildung 47 zu sehen ist. Hierbei wird für jedes Segment der Basislinie die Grundfläche ermittelt. Dann wird für jeden Gitterpunkt, welcher sich innerhalb der AABB der Grundfläche befindet, getestet, ob er sich bezüglich der x/z-Koordinaten innerhalb der Grund-

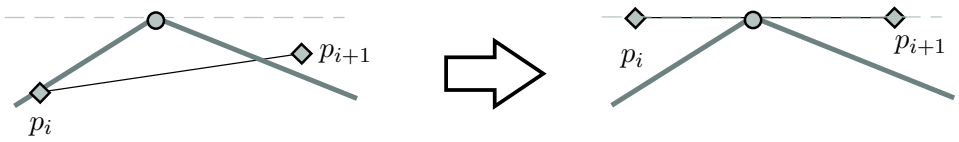


Abbildung 47: Anheben auf Niveau des Gitterpunktes

fläche befindet. Hierfür wird der in Kapitel 4.5.1 beschriebene MacMartin Line Crossing Test verwendet. Liegt der Gitterpunkt innerhalb der Fläche und ist dessen y-Koordinate als die y-Koordinate der Basislinienpunkte p_i, p_{i+1}, so werden die entsprechenden Punkte auf die Höhe des ermittelten Gitterpunktes angehoben.
Der Verlauf der so angepassten Basislinie kann im Anschluss noch verschiedenen Glättungsoperationen unterzogen werden, um einen durch verrauschte DGM-Höhenwerte verursachten unrealistischen Verkehrswegverlauf entgegenzuwirken.

6.2 Schienenbahn

Die ATKIS Objektart 3201 modelliert den Verkehrsweg Schienenbahn. Die Definition lautet im ATKIS-Objektartenkatalog: *„Verkehrsweg der schienengebundenen Verkehrsmittel außerhalb von flächenförmigen Bahnhofsanlagen. Die Schienenbahn umfaßt den Bahnkörper und die Bahnstrecken einschließlich der auf Über- oder Unterführungen (im Tunnel) verlaufenden Abschnitte. Zum Bahnkörper gehören auch kleinere Gräben zur Entwässerung des Bahnkörpers, Seiten- und Schutzstreifen und kleinere Böschungen.“*
Die Objektart 3201 wird durch verschiedene Attribute feinstrukturiert. Die in dieser Arbeit betrachteten Attribute sind insbesondere

- Breite des Verkehrsweges (BRV)
- Anzahl der Streckengleise (GLS)
- Spurweite (SPW)

6.2.1 Erzeugung der Gleiskörpergeometrie

Um die Geometriemenge so gering wie möglich zu halten, ist es notwendig, einen Kompromiss zwischen detailgetreuer Modelierung und notwendiger Vereinfachung zu Gunsten performanter Visualisierung zu finden. Das in dieser Arbeit erstellte Modell berücksichtigt ein- und zweigleisige Schienenwege und besteht aus den drei Basiskomponenten

- Schienen mit Bahnschwellen
- Gleisbett
- Seiten- und Mittelstreifen

Abbildung 48 zeigt eine schematische Darstellung eines zweigleisigen Schienenweges. Die angegebenen Maße sind aus der *Eisenbahn Bau- und Betriebsordnung*

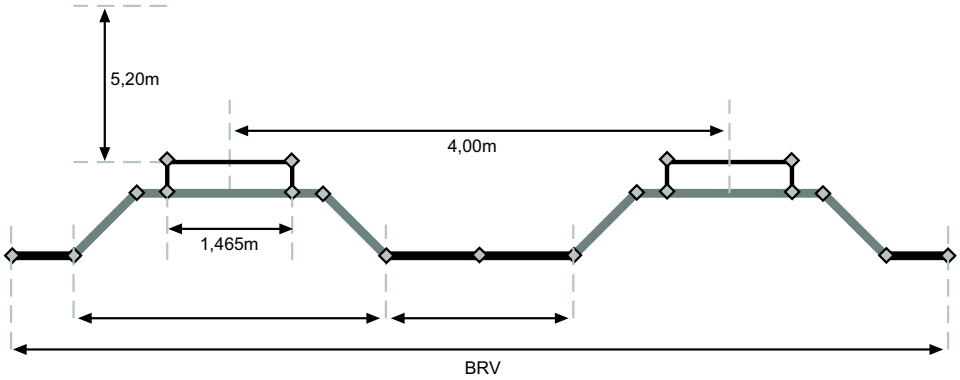


Abbildung 48: Aufbau eines Schienenweges

(EBO)[Ges67] übernommen. Die Abstände zwischen den Gleisen bzw. nach oben ergeben sich aus dem dort festgelegten, um den Fahrweg freizuhaltenden Raum, dem sog. *Lichtraumprofil*. Die für die Geometrieerzeugung relevanten Maße sind insbesondere die Spurweite der Gleise und der Abstand zweier benachbarter Fahrwege. Die Breiten der Gleisbettungen sind in der EBO nicht verzeichnet und mussten an Hand von Fotografien abgeschätzt werden. Da die Gesamtbreite des Schienenverkehrsweges im ATKIS-Basis DLM durch das Attribut "BRV" vorgegeben ist, kann es notwendig sein, in Abhängigkeit der gegebenen Breite auf die Seitenstreifen zu verzichten, um die in der EBO vorgeschriebenen Abstände einzuhalten.
Die Geometrie wird segmentweise komplett in Form von Triangle Strips repräsentiert, wobei Gleiskörper und Gleisbett in jeweils drei Triangle Strips aufgeteilt werden. Ein solches Segment ist durch zwei Punkte der zugrunde liegenden Basislinie gegeben. Abbildung 49 zeigt zwei Segmente eines eingleisigen Schienenweges in perspektivischer Ansicht. Trotz der stark vereinfachten Geometrie erzeugt diese Form der Geometrieerzeugung eine relativ große Menge an Geometriedaten, so ist beispielsweise ein zweigleisiges Schienenwegsegment aus 15 verschiedenen Triangle Strips aufgebaut, was insgesamt 30 Dreiecke pro Segment ausmacht.

6.2.2 Texturerzeugung

Um dem Modell der Gleisanlage einen größeren Detailreichtum zu verleihen ist die Texturierung von entscheidender Bedeutung. Für die Textur werden die Materialen

- Schotter für das Gleisbett

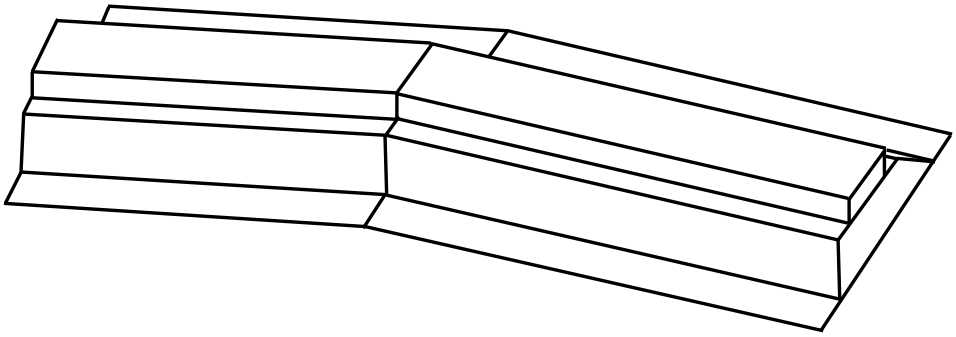


Abbildung 49: Schienenweggeometrie

- Holz für die Gleisschwellen
- Metall für die Schienen

prozedural erzeugt. Die Textur wird dabei so berechnet, dass sie nahtlos entlang des Schienenweges gekachelt werden kann. Abbildung 50 zeigt das Prinzip. Ermöglicht wird das nahtlose Kacheln durch die Anordnung der Gleisschwellen, von denen eine der beiden Schwellen in der Textur halbiert und an den Bildrand verschoben wird. Dabei kann das Texturmodell durch drei Parameter gesteuert

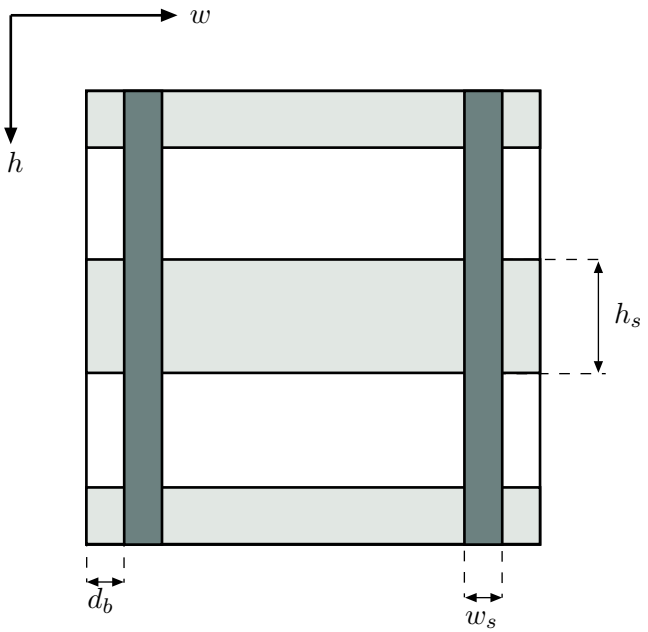


Abbildung 50: Schema der Gleistextur

werden:

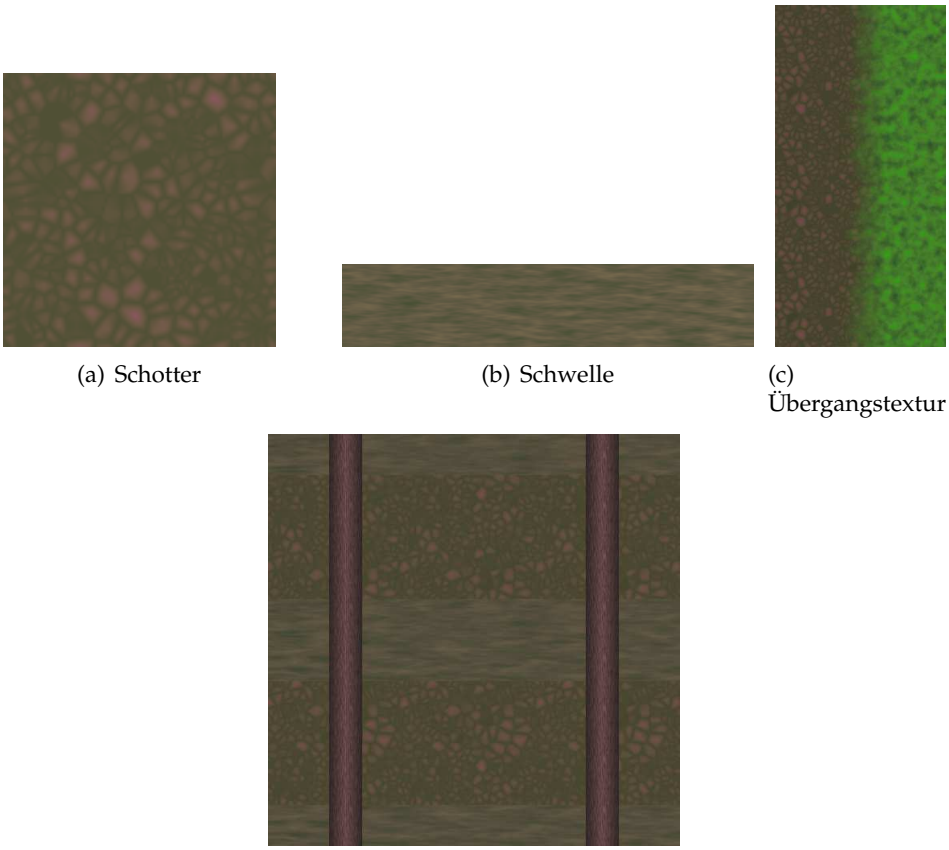
- Abstand der Schiene vom Bildrand (d_b)

- Breite der Schiene (w_s)
- Breite der Schwellen (h_s)

Für die Erzeugung des Schotters wird auf eine Cellular Noise Textur zurückgegriffen. Hierbei wird auf die Abstandsfunktion $d_1 - d_0$, als Abstandsmaß der Euklidische Abstand verwendet. Die Farbe des Schotters kann letztendlich frei gewählt werden, in der Natur sind, auf Grund der Witterungseinflüsse, zumeist dunkle, rötliche Farben vorherrschend.
Zur Erzeugung der Holztextur, welche für die Darstellung von Eisenbahnschwellen verwendet wird, kommt eine Perlin Noise Textur zum Einsatz. Zur Noise-Generierung werden drei Oktaven mit einer Persistenz von 0.7 verwendet. Der Effekt der Holzmaserung ergibt sich durch Skalierung der Noise-Funktion in x-Richtung. Die Farbgebung ist auch hier benutzerdefiniert, jedoch sind auch hier im Gelände dunkle Farben vorzufinden.
Die Textur für die Schienen wird ebenfalls mit Perlin Noise realisiert. Hier findet ebenfalls eine Skalierung der Noise Funktion in y-Richtung statt, dies soll den Effekt von Rost, welcher durch den Radlauf der Bahn abgerieben wird, simulieren.
Die Textur des Seitenstreifens soll den Übergang von Schotter zu einer anderen Geländetextur, z.B. zu Gras oder ähnlichem Bewuchs, simulieren. Hierzu muss die Schottertextur mit der entsprechenden anderen zu verwendenden Textur gemischt werden. Um den Übergängen zwischen den Texturen ein natürlicheres Aussehen zu verleihen, wird die Überblendung mittels einer 1D-Noise Funktion realisiert.
Abbildung 51 zeigt die einzelnen Komponenten der Schienenwegtextur sowie das Ergebnis aus der Vereinigung dieser Komponenten.

6.3 Straßen

Straßen werden im ATKIS-Basis DLM durch die Objektart 3101 beschrieben. Gemäß dem ATKIS Objektartenkatalog ist eine Straße ein *„Befestigter, dem allgemeinen Verkehr dienender Verkehrsweg. Die Straße umfaßt den Straßenkörper und die Fahrbahnen einschließlich der auf Über- und in Unterführungen (im Tunnel) verlaufenden Abschnitte. Zu den Straßen gehören auch Seiten- und kleinere Trennstreifen, begleitende Gräben zur Entwässerung der Straße, kleinere Böschungen, Parkstreifen und ähnliche Einrichtungen. Begleitende Fuß- und Radwege werden ebenfalls der Straße zugerechnet, soweit sie nicht durch Zwischenräume von mehr als 3 m Breite von der Fahrbahn getrennt sind.“* Zu den Straßen sollen in dieser Arbeit auch Wege gezählt werden, welche im ATKIS-Basis DLm als eigenständige Objektart aufgeführt sind (Objektart 3102). Da Wege prinzipiell die gleichen Eigenschaften haben, sollen sie hier an dieser Stelle unter einem Begriff zusammengefasst werden. Im Unterschied zu Straßen können Wege auch unbefestigt sein. Für die Objektarten 3101 und 3102 ist eine Vielzahl an



(a) Schotter (b) Schwelle (c) Übergangstextur

(d) Gleis

Abbildung 51: Komponenten der Gleistextur

Attributen definiert. Für diese Arbeit relevant sind die Attribute

- Besondere Fahrstreifen (BFS): Hiermit wird angegeben, ob der Straßenkörper Rad- und Fußgängerwege besitzt
- Breite der Fahrbahn (BRF), Angabe in dm
- Breite des Verkehrsweges (BRV)
- Anzahl der Fahrstreifen (FSZ)
- Befestigung (BEF)

6.3.1 Erzeugung der Straßengeometrie

Die Straßengeometrie kann im Vergleich zum Schienenweg wesentlich einfacher aufgebaut werden. Der überwiegende Teil an Detailreichtum kann über Texturen erzeugt werden. Abbildung 53 zeigt den schematischen Aufbau des Straßenkörpers. Auch hier wird die Geometrie aus Triangle Strips erzeugt.

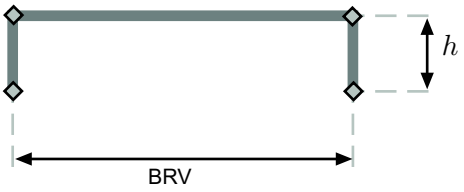


Abbildung 52: Schema der Straßengeometrie

6.3.2 Texturerzeugung

Zur Erzeugung der Straßentextur wird zunächst mittels Perlin Noise eine asphaltartige Struktur erzeugt. Hierzu wird eine Noise Funktion aus vier Oktaven mit einer Persistenz von $0,55$ verwendet. Die Fahrbahnmarkierungen werden dann durch Einfärben der in Abbildung 53 aufgezeigten Stellen in die Textur eingearbeitet. Wie bei der Schienenbahn Textur wird auch hier durch Anbringen jeweils einer Hälfte der Mittellinienmarkierung dafür gesorgt, dass die Textur kachelbar bleibt. Das Aussehen der Textur lässt sich an die Art der zu texturierenden Fahr-

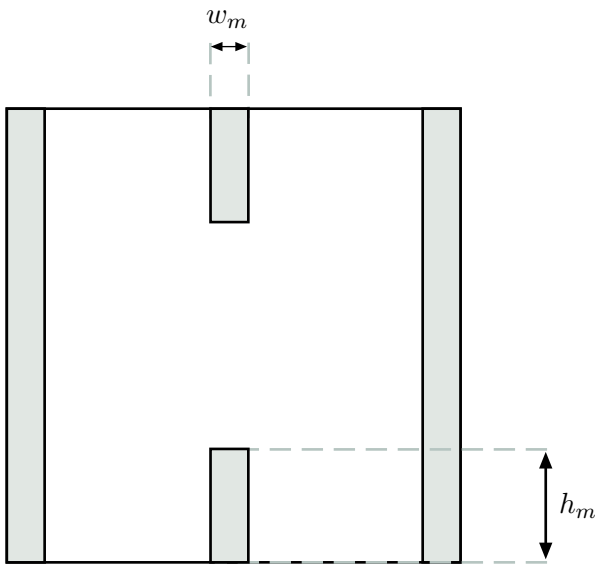


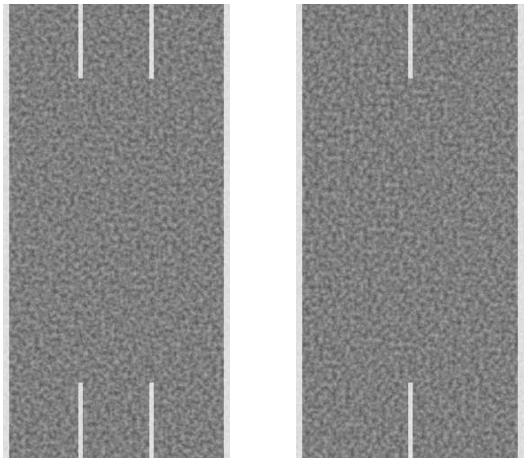
Abbildung 53: Schema der Straßen

bahn anpassen. Die vom Benutzer veränderbaren Parameter sind

- Art der Mittellinienmarkierung (durchgezogen / unterbrochen)
- Optionale Seitenmarkierungen
- Anzahl der Fahrstreifen

- Breite der Markierungslinien

Abbildung 54 zeigt Beispielbilder der so erzeugten Texturen.



(a) dreispurig (b) zweispurig

Abbildung 54: Straßentextur

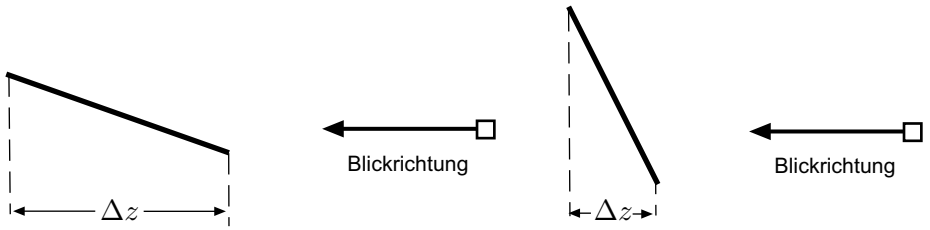
6.4 Rendering der Verkehrswege

Beim Rendering der das Gelände überlagernden Geometrie können Z-Buffer Artefakte auftreten. Dies geschieht zum einen bei der Betrachtung des Geländemodells aus größerer Entfernung, zum anderen, wenn Flächen des Terrains mit Flächen der darübergelegten Geometrie deckungsgleich sind bzw. die Flächen sehr nahe beieinander liegen. Auf Grund der begrenzten Präzision des Z-Buffers werden diese Flächen auf den gleichen Tiefenwert abgebildet, es kommt zum sog. *Z-flickering*. Schon bei mittlerem Betrachtungsabstand kommt dies auf Grund des nichtlinearen Z-Buffers auch bei den wie in Abschnitt 6.2.1 und 6.3.1 erzeugten Geometrien vor, obwohl hier die Fahrbahnen bzw. Schienenwege deutlich über der Basislinie angeordnet sind. Ein Weg, die Z-Buffer Genauigkeit zu erhöhen, besteht in der Einschränkung der maximalen Sichtweite, da ein kleinerer Bereich $Z_{far} - Z_{near}$ präziser abgetastet wird als ein großer Bereich. Da jedoch gerade bei der Visualisierung von ganzen Landschaften eine große Sichtweite erforderlich ist, um einen realistischen Eindruck zu erzielen, kann auf diese Option nicht uneingeschränkt zurückgegriffen werden. In Abhängigkeit der verwendeten Graphikhardware besteht die Möglichkeit, die Präzission des Z-Buffers treiberseitig zu erhöhen. Ein Umstellen von 16 Bit Z-Buffer Tiefe auf 24 Bit führte zu einer deutlichen Verbesserung der Darstellungsqualität. Wesentlich bessere Ergebnisse bezüglich der Z-Buffer Problematik werden erzielt, wenn beim Rendering der überlagernden Geometrie auf Z-Buffer Ebene ein geeigneter Offset auf die Tiefenwerte addiert wird.

Man erzielt so den Effekt, dass diese Geometrie einen Tiefenwert besitzt, welcher näher am Betrachter liegt und damit den Z-Buffer Test stets besteht. Die OpenGL API bietet hierfür die Funktion `glPolygonOffset(GLfloat factor, GLfloat units)`[17]. Der hiermit berechnete Offset setzt sich aus einem konstanten und / oder einem variablen Teil zusammen. Die Funktion arbeitet vor der Durchführung des Z-Buffer Tests auf den interpolierten Tiefenwerten einzelner Fragmente. Der Offset o berechnet sich hierbei durch

$$o = \text{factor} \cdot \Delta z + r \cdot \text{units}$$

Dabei ist Δz die maximale Änderung der Tiefenwerte des gerenderten Dreiecks in Relation zu dessen Fläche in Bildschirmkoordinaten. Der Faktor r beschreibt den kleinsten darstellbaren Abstand zweier Tiefenwerte, hängt also von $Z_{far} - Z_{near}$ sowie der eingestellten Z-Buffer Präzission ab. Der Parameter `factor` kontrolliert daher den variablen Teil des Offset-Wertes, da sich Δz in Abhängigkeit des Betrachtungswinkels ändert, wie in Abbildung 55 zu sehen ist. Durch die Addi-



(a) flach (b) steil

Abbildung 55: Tiefenwerte in Bahängigkeit der Blickrichtung

tion eines Offsets auf die Tiefenwerte können Situationen auftreten, in denen ein Fragment auf Grund des veränderten Tiefenwertes den Z-Buffer Test besteht, obwohl es von einem anderen Objekt verdeckt ist und somit nicht dargestellt werden dürfte. Dies tritt z.B. bei weiter entfernt darstellenden Objekten auf, da hier die Tiefenwerte der beiden Objekte, durch die Z-Buffer Präzision bedingt, zu nahe beieinander liegen. Wird der Offset-Wert zu niedrig gewählt, so treten Artefakte durch z-flickering auf. Durch Anpassung der Werte für `factor` und `units` kann dies für einen bestimmten Entfernungsbereich korrigiert werden. Um die korrekte Darstellung in allen Entfernungsbereichen sicherzustellen, müssen die Parameter für `glPolygonOffset` in Abhängigkeit der Entfernung des zu rendernden Objektes gesetzt werden. Zur Ermittlung optimaler Parameter wurden Testreihen mit verschiedenen Parametern durchgeführt und das Ergebnis an Hand der visuellen Qualität bewertet. Hierbei wurde die gerenderte Straßengeometrie untersucht. Als Bewertungskriterien für die Darstellungsqualität wurden hierbei die Faktoren

[17] http://www.opengl.org/documentation/specs/man_pages/hardcopy/GL/html/gl/polygonoffset.html

- Geländemodell durchdringt Straßengeometrie (Z-flickering)
- Straßengeometrie wird trotz Verdeckung sichtbar (Z-Buffer Präzission)

herangezogen. Für den Test wurde der Z-Buffer auf 24 Bit Präzision eingestellt, die Werte für Z_{near} und Z_{far} betrugen $0,5$ bzw. 10000. Zur Vereinfachung wurde der Entfernungsbereich in drei Intervalle unterteilt:

- Nahbereich
- Mittlerer Entfernungsbereich
- Fernbereich

Die Ergebnisse der Untersuchung werden in Tabelle 6 aufgezeigt.

Factor	Unit	Nahbereich	Mittlerer Entfernungsbereich	Fernbereich
0.0	0.0	sehr schlecht	sehr schlecht	sehr schlecht
0.0	-10.0	sehr schlecht	sehr schlecht	mittel - gut
0.0	-20.0	sehr schlecht	sehr schlecht	sehr gut
0.0	-40.0	sehr schlecht	mittel	sehr gut
0.0	-80.0	sehr schlecht	mittel	sehr gut
-5.0	0.0	mittel - schlecht	gut - mittel	gut
-10.0	0.0	schlecht	mittel	gut
-20.0	0.0	schlecht	mittel	schlecht
-5.0	-80.0	mittel	mittel	gut
-5.0	-130.0	gut	schlecht	schlecht

Tabelle 6: Werte für `glPolygonOffset`

An Hand der ermittelten optimalen Werte wird nun beim Rendering die Funktion `glPolygonOffset` mit veränderten Parametern in Abhängigkeit der Entfernung des zu rendernden Objektes aufgerufen. Hierfür muss der Abstand eines mit `glPolygonOffset` zu rendernden Objektes zum Augpunkt bestimmt werden. Um die Performanz nicht zu sehr zu beeinträchtigen, wird nur jeweils der Abstand zum Mittelpunkt der das Objekt umgebenden Bounding Box berechnet. Insgesamt ist die eingesetzte Technik doch relativ aufwändig, die Framerate wird dadurch in etwa halbiert.

7 Prozedurale Vegetation

Ein wesentlicher Bestandteil von Landschaften ist das Vorkommen von Vegetation. Ein großer Teil der Landmasse ist mit Pflanzenbewuchs überzogen, die immense Vielfalt an unterschiedlichen Ausprägungen der Pflanzenformen geben den verschiedenen Landschaften der Erde ihr charakteristisches Aussehen. In der Computergraphik stellt die Darstellung von Vegetation ein aktuelles Forschungsgebiet dar, und mit dem Aufkommen leistungsfähiger Hardware wurden in diesem Bereich zahlreiche Fortschritte erzielt.
Auf Grund der immensen Anzahl an verschiedenen Pflanzen in einem Gelände ist es bei der Erstellung eines Landschaftsmodells letztendlich unmöglich, jede einzelne in der Realität vorkommende Pflanze in das Modell zu integrieren, da die hierbei entstehenden Datenmengen schnell in nicht-handhabbare Größen anwachsen. Daher ist es sinnvoll, die Vegetation automatisch an Hand von parametrisierbaren Modellen erzeugen zu lassen. Dieser Prozess lässt sich in zwei grundsätzliche Bereiche aufgliedern:

- Erzeugung von Pflanzenmodellen
- Anordnung der verschiedenen Pflanzen auf dem Terrain

In den folgenden Abschnitten sollen diese Punkte genauer erläutert werden.

7.1 Vegetationsformen

In der Biologie werden die verschiedenen Formen von Pflanzen in einem hierarchisch aufgebauten System unterschieden. Da dieses System sehr umfangreich ist, sei an dieser Stelle auf die zahlreiche Literatur zur Systematik der Pflanzen verwiesen. Eine für diese Arbeit sinnvolle Vereinfachung ist die grobe Einteilung in

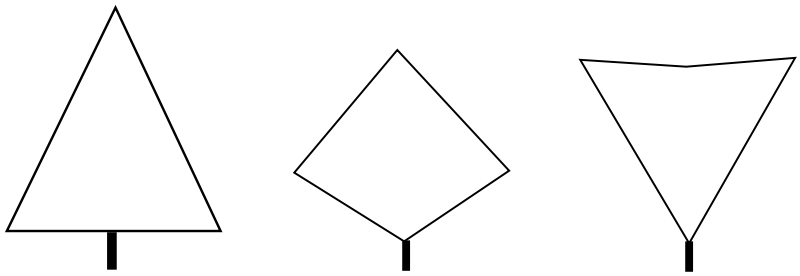
- Bäume
- Sträucher und Kleinpflanzen
- Gras, Flechten, Moose

Abgesehen von Moosen und Flechten besitzen die aufgeführten Pflanzen den grundlegenden Aufbau

- Wurzel
- Sprossachse
- Blatt

Im Allgemeinen befindet sich das Wurzelsystem unterirdisch und ist somit nicht zu sehen. Die Sprossachse bildet das tragende Gerüst der Pflanze und dient neben

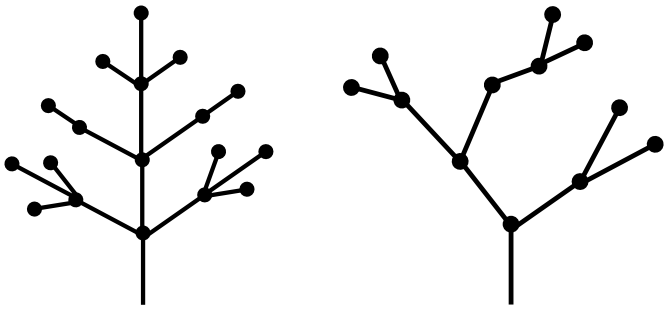
der Stabilisierung und Formgebung dem Nährstofftransport sowie der Ausrichtung der Pflanze. Letztendlich ergibt sich die Vielfalt der verschiedenen Pflanzenformen aus der unterschiedlichen Ausprägung dieser Grundbausteine. Betrachtet man die äußere Form verschiedener Pflanzen, so lassen sich drei Grundmuster erkennen, welche in Abbildung 56 dargestellt werden. Die unterschiedlichen



(a) basitonisch (b) mesotonisch (c) acrotonisch

Abbildung 56: Äußere Wuchsform

Wuchsformen resultieren letztendlich aus den Verzweigungen der Sprossachse. Es existieren zwei Grundtypen von Verzweigungen, zum einen die *Monopodiale Verzweigung*, bei der eine durchgehende Achse bestehen bleibt (wie z.B. bei Fichten), zum anderen die *Sympodiale Verzweigung*, bei der das Wachstum von den austreibenden Seitenachsen fortgesetzt wird. Von diesen Verzweigungs-Grundtypen gibt es noch zahlreiche Abwandlungen. Wie bei der Sprossachse existieren auch



(a) Monopodiale Verzweigung (b) Sympodiale Verzweigung

Abbildung 57: Verzweigung der Sprossachse

bei den Blättern zahlreiche unterschiedliche Wuchsformen. Grundsätzlich kann hier zwischen einfachen und zusammengesetzten Blättern unterschieden werden. Einen guten Überblick über die verschiedenen Blattformen gibt [Sen].
Im ATKIS Basis-DLM wird die konkrete Ausprägung der Vegetation nur sehr grob modelliert. So unterscheiden die Objektarten 4107 (Wald, Forst), 4108 (Gehölz),

4201 (Baum) und 4202 (Baumreihe) lediglich zwischen Laub-, Nadel,- oder Mischholz. Weiterhin existiert noch die Objektart 4203 (Hecke). Die Objektart 4102 (Grünland) steht für Gras- und Rasenflächen. Weiterhin beschreiben die Objektarten 4104 (Heide), 4105 (Moor, Moos) und 4106 (Sumpf, Ried) nur die grobe Kategorie Vegetation, genaue Anbgaben über die tatsächlich darin enthaltenen Pflanzenarten werden nicht gemacht. Die Objektart 4101 (Ackerland) besitzt das Attribut „Streuobst“ was einen Hinweis auf die zu erwartende Vegetation gibt.

7.2 Platzierung im Gelände

Um die Verteilung der Vegetation im Gelände zu modellieren, müssen die verschiedenen Faktoren, welche das Pflanzenwachstum beeinflussen, in die Modellbildung mit einbezogen werden. Dabei kann die Gesamtheit der im Gelände vorkommenden Pflanzen als *Ökosystem* zusammengefasst werden. Das gesamte Ökosystem setzt sich aus einer Vielzahl von lokalen, abgegrenzten Ökosystemen zusammen, z.B. können in einem bestimmten Gebiet Wald- und Wiesenflächen gemischt vorkommen. Das Vorkommen unterschiedlicher Vegetationsformen und deren Platzierung in einem lokalen Ökosystem wird durch die dort vorherrschenden Standortfaktoren bestimmt. In [Ham] wird die Modellierung von Ökosystemen im Detail beschrieben. Als grundlegende Standortfaktoren werden

- Höhe des Geländes über NN
- Lokale Höhenschwankungen
- Hangneigung
- Hangrichtung
- Lokales Gruppenverhalten von Pflanzen

beschrieben. Der Einfluss der Geländehöhe auf die Vegetation lässt sich sehr gut in Gebirgslandschaften beobachten, wo verschiedene Vegetationszonen in Abhängigkeit der Höhe vorzufinden sind. Ab einer gewissen Höhe kann letztendlich keine Vegetation mehr gedeihen. Die Hangneigung beeinflusst im wesentlichen das Ausmaß der Errosion des Bodens, was sich dann letztendlich mit der Verfügbarkeit von Nährstoffen und damit der möglichen Dichte der Vegetation äussert. Ab einer gewissen Steigung des Geländes wird der Boden vollständig errodiert und der blanke Fels freigelegt. Lokale Höhenschwankungen wie Täler oder Gebirgsgrate beeinflussen die Vegetationsformen, da hier durch Verdeckungen die Witterungs- und Sonnenexposition definiert wird. Die Hangrichtung hat hauptsächlich Einfluss auf die Zahl der Sonnenstunden sowie die Windexposition in Abhängigkeit der lokalen Hauptwindrichtungen. Lokales Gruppenverhalten hat schließlich Einfluss auf das Vorkommen und die Verteilung von Pflanzengruppen. Dies ist hauptsächlich aus dem Reproduktionsverhalten der verschiedenen Pflanzen ableitbar,

Pflanzen welche ihre Samen mit dem Wind oder durch Tiere wie Vögel oder Insekten verbreiten, neigen tendenziell weniger zur Ausbildung von Gruppen als Pflanzen, welche ihre Samen direkt auf den Boden fallen lassen.

7.3 Lindenmayer Systeme

Lindenmeyer Systeme oder kurz L-Systeme stellen einen von *Aristid Lindenmeyer* vorgestellten mathematischen Formalismus dar, welcher für die Erzeugung von Pflanzenmodellen verwendet werden kann. Dabei werden die Pflanzenmodelle auf Grundlage einer formalen Grammatik erzeugt, die Elemente des durch die Grammatik definierten Alphabets werden in eine graphische Repräsentation des Pflanzenmodells übersetzt. Formal ist ein L-System ein Quadrupel $G_L = (V, S, \omega, P)$. Dabei ist V eine Menge von nicht-terminalen Symbolen (Variablen), S eine Menge von terminalen Symbolen, $\omega \in V \times S$ ein nichtleeres Wort, welches das Startsymbol der Grammatik darstellt und auch als *Axiom* des L-Systems bezeichnet wird, sowie den Produktsionsregeln $P : V \rightarrow V \times S$. Im Gegensatz zu den formalen Grammatiken der Chomsky-Hierarchie werden bei den L-Systemen stets alle nicht-terminalen Symbole (Variablen) einer Produktionsregel ersetzt. Zur Berechnung eines durch G_L definierten Wortes werden, beginnend mit dem Axiom ω und einer gegebenen Rekursionstiefe n, die entsprechenden Produktionsregeln n mal angewendet.
Um eine graphische Interpretation des eines durch ein L-System erzeugen Wortes zu erhalten, werden die aus den n Produktionen resultierenden terminalen Symbole des Wortes in konkrete Zeichenbefehle für ein Graphiksystem übersetzt. Dabei stellen die Zeichenbefehle Anweisungen für einen sog. *Turtle-Graphik* Mechanismus dar. Die Grundelemente des Turtle-Graphiksystems sind hier

- Drehung um einen gegebenen Winkel α
- Zeichnen eines Segmentes der Länge l

Für die Drehung muss hierbei angegeben werden, um welche der drei Basisachsen des Koordinatensystems gedreht werden soll. Grundsätzlich sind hier noch weitere optionale Angaben denkbar, für das Zeichnen eines Segmentes wurde in dieser Arbeit noch ein Parameter für die Breite des zu zeichnenden Elementes implementiert.
Es existieren verschiedene Klassen von L-Systemen, welche sich in der Mächtigkeit der von ihnen beschriebenen formalen Sprachen unterscheiden. In den folgenden Abschnitten soll ein grober Überblick über die in dieser Arbeit implementierten Klassen gegeben werden, für eine vollständige Beschreibung des theoretischen Hintergundes sei auf [AL90] verwiesen.

Kontextfreie L-Systeme

Kontextfreie L-Systeme stellen die einfachste Form von Lindenmeyer Systemen dar. Sie sind determiniert, d.h. für jedes Symbol aus V gibt es maximal eine Ersetzungsregel.

L-Systeme mit Kelleroperationen

Mit den bisher vorgestellten Befehlen „Rotation“ und „Zeichenen eines Elementes“ lassen sich letztendlich nur Linienzüge erstellen. Um die in Abbildung 57 aufgezeigten Verzweigungen zu realisieren, wird die Grammatik um eine Stack-Funktionalität erweitert. Hierbei handelt es sich um einen LIFO-Stack, auf welchem die aktuelle Position und Ausrichtung der Turtle gespeichert wird.

Probabilistische L-Systeme

Bei Probabilisitsche L-Systemen können einer zu ersetzenden Variablen mehrere Produktionsregeln zugewiesen werden, von denen dann mit einer für die jeweilige Produktionsregel gegebenen Wahrscheinlichkeit eine ausgewählt wird. Damit lassen sich aus einer Grammatik mehrere mögliche Ableitungen berechnen.

Tropismus

Für das Wachstum von Pflanzen spielen eine Vielzahl von äußeren Faktoren eine Rolle (vgl. Abschnitt 7.2). Auf die konkrete Pflanze, welche unter bestimmten Bedingungen wächst, haben diese äußeren Faktoren u.a. auf die Wuchsrichtung Einfluss. Dies wird als *Tropismus* bezeichnet. Das Ausmaß dieser Beeinflussung lässt sich durch eine vektorielle Größe beschreiben. Faktoren, welche die Wuchsrichtung beeinflussen, sind z.B. Wind, Richtung der Sonneneinstrahlung und Gravitation.
Die Wirkung des Tropismusvektors besteht letztendlich darin, das die Ausrichtung der Turtle bei der Abarbeitung der Produktionsreglen verändert wird. Hierfür wird aus dem Tropismusvektor $\vec{t}$ und der aktuellen Ausrichtung der Turtle $\vec{h}$ ein Offset α_t mittels

$$\alpha_t = 90 \cdot \left(1 - \left(\frac{1}{1 + |\vec{h} \times \vec{t}|}\right)\right)$$

berechnet.

7.4 Rendering von Vegetation

Um eine gegebene graphische Beschreibung der Vegetation, welche z.B. mittels L-Systemen erzeugt wurde, in von einem Echtzeit 3D-Renderingsystem darstell-

bare Geometriedaten zu übersetzen, müssen verschiedene Punkte beachtet werden. Da Pflanzen im Allgemeinen sehr komplexe Strukturen darstellen, geht eine 3D-Modellierung mit einer hohen Zahl an zu rendernden Dreiecken einher. Da dies für das Rendering der kompletten Vegetation einer Landschaft nicht tragbar ist, muss die Pflanzengeometrie vereinfacht dargestellt werden. Um trotzdem einen hohen Detailgrad zu simulieren, bietet sich die Verwendung von texturiereter Geometrie an, insbesondere die Verwendung von sog. *Billboards*. Hiermit lässt sich die Zahl der zu rendernden Dreiecke erheblich einschränken. Bei der Erzeugung der Billboards wird dazu zwischen Stamm bzw. Astwerk und dem Blätterdach unterschieden, welches seinerseits wieder aus Ästen und den eigentlichen Blättern besteht. Diese beiden Grundelemente werden dann jeweils als zwei gekreuzte Bilboards gezeichnet. Durch diese Methode lässt sich eine dreidimensionale Struktur der Pflanze erzeugen. In Abbildung 58 wird dies beispielhaft für den Stamm und das Astwerk veranschaulicht. Für beide Teile der Baumgeometrie

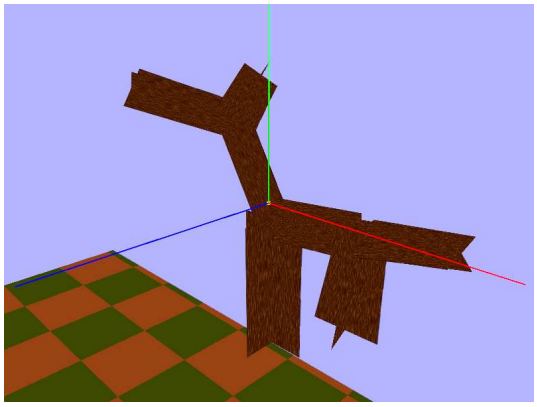
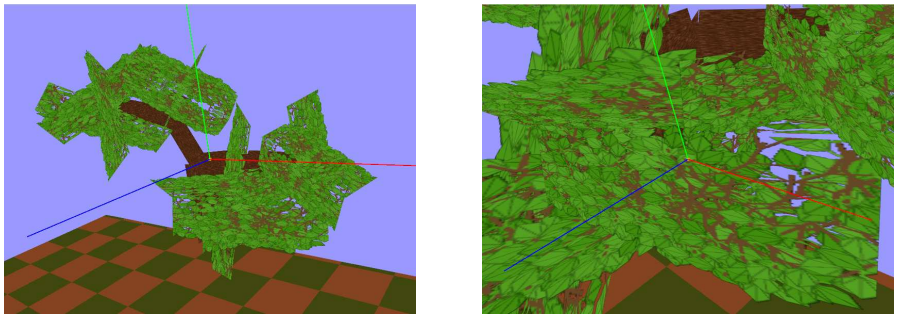


Abbildung 58: Stammgeometrie aus Bilboards

wird bei der Initialisierung des L-Systems jeweils eine maximale Rekusionstiefe angegeben. Die Bilboards für das Blätterdach werden mit ihrer entsprechenden Anzahl an Rekusionen vorausberechnet. Bei der Erzeugung der Stamm- und Astgeometrie wird in der letzten Rekursionsstufe die Blätterdachgeometrie an Stelle der Stammgeometrie gezeichnet. Bei der Berechnung des Blätterdachs wird in selbiger Weise verfahren, das Ergebnis wird jedoch in eine Textur gerendert, welche dann zur Texturierung des Bilboards verwendet wird. Da dieser Schritt bei der Erstellung der Pflanzengeometrie durchgeführt wird, kann hier wesentlich mehr Geometrie erzeugt werden. So können z.B. die Blätter einzeln durch mehrere Dreiecke modelliert werden. Abbildung 59 zeigt das Ergebnis dieser Vorgehensweise.



(a) Blätterdach (b) aus der Nähe

Abbildung 59: Erstellung des Blätterdachs

8 Bewertung

Das in der Aufgabenstellung definierte Ziel dieser Arbeit ist die Erstellung eines Digitalen Landschaftsmodells auf Grundlage der Vermessungsdaten ATKIS-Basis DLM und dem Digitalen Geländemodell (DGM). Dabei sollen für die verschiednen Aspekte der Modellerzeugung prozedurale Verfahren zum Einsatz kommen. In diesem Kapitel sollen die vorgestellten Verfahren nun unter dem Gesichtspunkt der Automatisierbarkeit bzw. der Parametrisierbarkeit der Modellerstellung bewertet werden. Weiterhin soll die Eignung des erzeugten Modells zur Verwendung in einem Echtzeitrenderingsystem kritisch betrachtet, sowie der Grad an erreichtem Realismus im Vergleich zur Realität beleuchtet werden.

8.1 Landschaftsmodell

Die Verwendung von topographischen Karten als Datengrundlage bietet für das daraus erzeugte Modell die Möglichkeit, die genaue funktionale Prägung eines Landschaftsteiles zu bestimmen. Im ATKIS Basis-DLM wird eine relativ große Zahl von unterschiedlichen Objektarten unterschieden, so dass sich ein relativ genaues Bild des grundsätzlichen Aufbaus der modellierten Landschaft ergibt. So ist es mit den in Kapitel 4 vorgestellten Verfahren möglich, aus den ATKIS-Basis DLM sowie den Höheninformationen des DGM ein trianguliertes Oberflächenrelief zu erstellen. Die im ATKIS-Basis DLM enthaltenen funktionalen Informationen der einzelnen Flächenbestandteile stehen im dreidimensionalen Landschaftsmodell zur Verfügung (vgl. Abbildung 60). Dies stellt einen klaren Vorteil gegenüber der Texturierung eines Höhenmodells mit Luftbild- oder Satelitenaufnahmen dar.

Jedoch ist auch die Genauigkeit der ATKIS-Basis DLM Daten beschränkt. Besonders durch die linienförmig modellierten Verkehrswege ergeben sich mitunter Ungenauigkeiten, da Verkehrswege letztendlich immer flächenförmige Objekte voneinander abgrenzen und somit die Grenzlinie für diese Objekte repräsentieren. Der den Verkehrsweg repräsentierende Linienzug verläuft in der Fahrbahnmitte, die Flächen auf beiden Seiten der Grenzlinie schließen bündig an diese an und werden somit durch den Verkehrsweg überdeckt (vgl. Abbildung 61), was letztendlich in einer nicht-eindeutigen Zuordnung von räumlicher Koordinate zur funktionalen Ausprägung resultiert. Bei der Platzierung von Objekten auf einer solchen Fläche kann dies zu Überschneidungen mit dem Verkehrsweg führen

Durch das gezeigte Triangulationsverfahren entsteht ein dreidimensionales Modell der Landschaft, welches durch die gezeigten Verfahren zur Texturkoordinaten- und Normalenberechnung die notwendigen Daten zur Verwendung in einem Echtzeitrenderingsystem enthält. Die Verwendung von Indexed Face Sets ermöglicht die Nutzung moderner Rendering Techniken wie Vertex Arrays oder Vertex Buffer Objects. Durch die im Vergleich zur erfassten Fläche relativ hohe Abtastrate des

DGMs (10 Meter) entsteht eine große Zahl an Dreiecken. Für eine Fläche von 1 km^2 entstehen bei einer regelmäßigen Triangulierung bereits 20000 Dreiecke. Durch die Unterteilung der Landschaft in funktionale Flächen und der damit verbundenen irregulären Triangulierung erhöht sich die Zahl der Dreiecke nochmals deutlich. Daher sind für die effiziente Darstellung des Landschaftsmodelles *Level of Detail (LOD)* Mechanismen erforderlich. Die gängigsten aktuellen LOD-Algorithmen (vgl. [Bae06]) bauen auf einer regelmäßig triangulierten Terraingeometrie auf. Eine der Anforderungen an das implementierte Triangulationsverfahren war die Erzeugung von möglichst regelmäßiger Geometrie. Es zeigt sich, dass diese Anforderung durch die in Abschnitt 4.5 dargelegte Unterscheidung zwischen der Triangulation innerer DGM-Gitterpunkte und der Triangulation der Umrisslinienpunkte erfüllt ist. Hierdurch bieten sich Ansätze für die Anwendung effizienter, echtzeitfähiger LOD-Verfahren.
Obwohl die Abtastrate des DGMs mit 10 Metern relativ hoch ist, reicht diese Samplingrate nicht immer aus, um eine fehlerfreie Verbindung von ATKIS-Basis DLM mit DGM-Höhendaten zu erzielen. In Abschnitt 3.3.1 werden die durch die diskrete Abtastung entstehenden Probleme beschrieben. Besonders die fehlerhaften Übergänge zwischen Land- und Gewässerflächen wirken einem realistischen Eindruck entgegen. Ein automatisierter Umgang mit dieser Situation ist prinzipiell denkbar. Hierfür müsste die Umrisslinie einer Wasserfläche auf einen einheitlichen Höhenwert, welcher dem durchschnitlichen Höhenwert der inneren Gitterpunkte der Wasserfläche entspricht, abgesenkt werden. Da hierfür auch alle an die so bearbeitete Wasserfläche angrenzenden Flächen berücksichtigt werden müssen, ist eine Erfassung von Nachbarschaftsbeziehungen für jede Fläche notwendig. Durch die Eigenheiten der Speicherung von Geometrieinformationen des EDBS-Formates stecken die Nachbarschaftsbeziehungen implizit in den EDBS-Datensätzen und können daher genutzt werden. Dies wurde jedoch im Rahmen dieser Arbeit nicht implementiert.
Bei der Abtastung des Geländes treten durch die Vermessungstechnik bedingt Artefakte auf. Obwohl die DGM-Daten seitens der Vermessungsbehörde scheinbar geglättet wurden, gibt es doch einige Stellen im DGM, welche auf Grund der baulichen Situation an den Messpunkten Unebenheiten aufweisen. Abbildung 62 veranschaulicht dies am Beispiel des Deutschen Ecks in Koblenz.

8.2 Prozedurale Infrastruktur

In Kapitel 6 wird ein System zur Erzeugung prozeduraler Straßen- und Schienenwege aus den in den Vermessungsdaten als Linienzüge repräsentierten DLM-Objekten vorgestellt. Mittels dieser Verfahren lassen sich texturierte Straßen- und Schienenwegmodelle automatisiert erstellen. Dabei ist das erzeugte Modell sowohl auf Geometrieebene (Anzahl der Gleise im Schienenweg), als auch in seiner

Texturierung parametrisierbar. Sowohl die Anzahl der Gleise bei Schienenwegen als auch die Anzahl der Fahrspuren auf Straßen sind in den ATKIS-Basis DLM Datensätzen enthalten, so dass die Verkehrswege hinsichtlich dieser Parameter automatisch erstellt werden können. Für eine realistische Nachbildung der realen Situation ist diese Datengrundlage jedoch in der Regel nicht ausreichend, da die Beschaffenheit von Verkehrswegen in der Realität ungleich komplexer ist. So sind z.B. bei Schienenverkehrswegen eisenbahntechnische Einrichtungen wie Signalanlagen oder Oberleitungsmasten nicht mit erfasst. Die Objektart 3201 (Schienenbahn) besitzt lediglich Angaben zur Elektrifizierung der Strecke. Weiterhin sind Weichenanlagen nicht modelliert.
Für die Straßenverkehrswege sind als visuell darstellbare Eigenschaften in den Vermessungsdaten die Breite des Verkehrsweges, die Anzahl der Fahrstreifen und das Vorhandensein von Fußgänger- und Radwegen erfasst. Weiterhin werden Angaben über das Material der Fahrbahnoberfläche (Pflaster, Beton, Bitumen / Asphalt) gespeichert. Die genaue Art der Straße (Bundesautobahn, Bundesstraße etc.) sowie deren Verkehrsbedeutung (Durchgangsverkehr, Anliegerverkehr) sind in den Daten enthalten. Bezüglich dieser Aspekte ist daher eine Parametrisierbarkeit der zu erzeugenden Straßengeometrie gegeben. Nicht modelliert werden Angaben für Fahrbahnmarkierungen, Verkehrszeichen, Ampelanlagen, Parkstreifen und Verkehrsinseln. Gerade diese Details sind jedoch die Grundlage für ein deutlich realistischeres Straßenmodell; auf Grund des Fehlens dieser Daten können diese Aspekte jedoch nicht automatisch aus ATKIS-Basis DLM Daten erzeugt werden. Ferner werden Kreuzungen, Einmündungen etc. von Straßen nicht explizit modelliert (vgl. Abbildung 64).

Um eine automatische Generierung von Kreuzungen zu realisieren, müssen aus den im ATKIS-Basis DLM enthaltenen Straßenobjekten Informationen über die Topologie des Straßennetzes gewonnen werden. Informationen, welche Straßenteile an ihren Start- bzw. Endpunkten miteinander verbunden sind, sind in den Vermessungsdaten nicht enthalten.
Als weitere Schwierigkeit hat sich die Erfassung der durch das ATKIS Datenmodell vorgegebenen möglichen Daten von Seiten der Vermessungsbehörde erwiesen. So ist z.B. das Attribut „BFS“ (Besondere Fahrstreifen) der ATKIS-Objektart 3101 (Straße), welches die Information über eventuell vorhandene Fußgänger- und Radwege enthält, in den Daten durchweg mit dem Wert „Attribut trifft nicht zu“ belegt. Daher können auf Basis der erhaltenen Datensätze Fußgänger- und Fahrradwege nicht automatisch erzeugt werden. Gerade Fußgängerwege prägen jedoch das Straßenbild in urbanen Gebieten sehr stark, hierdurch geht ein erheblicher Teil an realistischer Wirkung verloren. Weiterhin ist in einigen Fällen die Anzahl der Fahrstreifen im Vergleich zur realen Situation nicht korrekt in den Datenbeständen erfasst. Hierbei sind entweder zu wenige bzw. ein einzelner Fahrstreifen, für Straßen welche an dieser Stelle in der realen Situation mehrere Fahrstreifen

besitzen, angegeben, oder es sind für Straßen, welche in der Realität keine Fahrstreifen besitzen, zwei oder mehrere Fahrstreifen angegeben (siehe Abbildung 65).
Die Überlagerung des Geländereliefs mit Verkehrsweggeometrie durch die in Abschnitt 6.1 aufgezeigten Vorgehensweisen kann einen Großteil der visuellen Artefakte beheben. Jedoch kommt es durch die unregelmäßige Triangulierung des Geländereliefs immer wieder zu Durchdringungen von Verkehrsweg- und Geländegeometrie (vgl. Abbildung 66). Dies resultiert hauptsächlich aus der in Kaptitel 3.1.1 beschriebenen Interpolation der Höhenwerte. Durch die unregelmäßige Triangulierung des Oberflächenreliefs können sich Kantenverläufe ergeben, welche in ihrer Höhe überhalb der interpolierten Höhendaten liegen. Ist die bilineare Interpolation der DGM-Daten für die Erstellung des Reliefmodells ausreichend, so muss für die korrekte automatische Platzierung von Verkehrswegen eine genauere Technik zur Höhenermittlung verwendet werden. Die exaktesten Resultate würde mit Sicherheit ein Strahl-Gelände Schnitttest erbringen, um einen solchen effizient zu implementieren wäre jedoch in Anbetracht der großen Datenmengen auf jeden Fall eine Erfassung der Nachbarschaftsbeziehungen der DLM-Objekte notwendig. Hiermit könnten effizient diejenigen Flächen ausgewählt und getestet werden, über welche der Verkehrsweg verläuft. Durch die Verwendung von `glPolygonOffset` können Artefakte durch Z-flickering zuverlässig unterdrückt werden. Das Rendering entfernter Objekte unter Verwendung von `glPolygonoffset` ist jedoch erst durch die Verwendung einer Z-Buffer Präzision von 24 Bit zum großen Teil frei von visuellen Artefakten. Ferner ist es mittels `glPolygonOffset` möglich, leichte Durchdringungen der Verkehrsweggeometrie durch das Gelände zu unterdrücken. Auf die beschriebene aufwändige Anpassung der Verkehrsweggeometrie an das Geländerelief kann daher trotz des Einsatzes von `glPolygonOffset` nicht verzichtet werden.
Durch die Anpassung der Verkehrswege an das Geländerelief entsteht ein große Menge an Geometriedaten. Verläuft der Verkehrsweg über relativ ebenes Gelände, können hier LOD-Techniken zum Einsatz kommen, um die Anzahl der Verkehrswegsegmente zu reduzieren. Im Gegenzug muss die Anzahl der Segmente in Kurven bzw. Einmündungen erhöht werden, um an diesen Stellen eine detailiertere Darstellung zu erreichen. Das automatische Hinzufügen bzw. Entfernen von zusätzlichen Punkten in die Basislinie lässt sich in der vorgestellte Verarbeitungskette der Daten prinzipiell realisieren, wurde aber im Rahmen dieser Arbeit nicht mehr implementiert.
Letztendlich hat sich gezeigt das bei der Anpassung von Verkehrswegen an ein Geländeprofil der in Abschnit 6.1 beschriebene Ansatz des Draping für die gegebenen Daten prinzipiell funktioniert und zufriedenstellende Ergebnisse liefert. Bei verrauschten oder grob abgetasteten Höhendaten ist eine Glättung bzw. eine präzise Anpassung des Verkehrswegverlaufes möglich. Würde hier die Technik des Stitchings verwendet werden, so müsste sehr wahrscheinlich eine geeignete

Glättung der Höhendaten im Bereich der Straßengeometrie vorgenommen werden. Durch die Methode des Draping bleiben die einzelnen Geometrien der Verkehrswege und der Flächen des Geländereliefs voneinander getrennt.
Die im ATKIS-Basis DLM unter der Objektgruppe 3500 zusammengefassten *„Anlagen und Bauwerke für Verkehr, Transport und Kommunikation“* eignen sich nur bedingt zur automatisierten Generierung. Ein großes Problem ist hierbei das fehlen von Höhenangaben. So sind zum Beispiel Brücken in ihrer Bauart relativ vielschichtig modelliert (Hängebrücke, Bogenbrücke, etc.), doch fehlen hier letztendlich genauere Angaben wie Brückenhöhe, Anzahl und Position der Brückenpfeiler Farbgebung und ähnliches. Gerade große Brücken sind jedoch weithin sichtbare Landmarken, welche oftmals sogar den Status eines regionalen Wahrzeichens besitzen, und in ihrer genauen Bauform zumeist so einzigartig, dass durch eine automatische, parametrisierte Modellierung der notwendige Realitätsgrad sehr wahrscheinlich nicht erreicht werden würde. Denkbar wäre eine automatische Modellierung von Über- bzw. Unterführungen von Verkehrswegen, wie z.B. Eisenbahnunterführungen. Die Höhe über dem unterführten Objekt würde sich aus der Höhe des überführenden Verkehrsweges ergeben. In den verwendeten DGM-Daten sind an solchen Überführungen meist Einbuchtungen vorzufinden, so dass die Höhe im DGM dem überbauten Objekt entspricht. Für die automatisierte Erstellung von Verkehrswegen wäre die Generierung von Bahnhöfen bzw. Haltepunkten interessant. Da große Bahnhöfe sicherlich markante, sehr individuelle Gebäude mit hohem Wiedererkennungswert darstellen, sind sie für eine automatische Erzeugung jedoch eher ungeeignet. Auch auf Grund der in den Vermessungsdaten erfassten Parameter lässt sich eine automatische Generierung eher schwer realisieren, da für die Objektart 3501 (Bahnhofsanlage) als visuell darstellbare Eigenschaft lediglich das Attribut BFK (Bahnhofskategorie) existiert, welches die Objektart 3501 in *„Bahnhof“*, *„Haltestelle“*, oder *„Haltepunkt“* unterteilt. Selbst für eine semi-automatische Erzeugung von kleineren Bahnhöfen sollte zumindest die Anzahl der Gleise bzw. Bahnsteige in den Daten enthalten sein.
Bei den in den Vermessungsdaten erfassten Masten und Antennen stellt sich wieder das Problem der Fehlenden Höhenangaben, sowohl der eigentlichen Gebäudehöhe als auch der Höhe des Fußpunktes der als punktförmige Objekte modellierten Anlagen. Die Gebäudehöhe ist als Attribut HHO (Objekthöhe) für die Objektart 3541 vorgesehen, wird jedoch in den vorhandenen Daten von Seiten der vermessenden Behörde nicht verwendet. Da Masten, insbesondere Sendemasten welche als große Fernmelde- oder Fernsehtürme gebaut wurden, das Landschaftsbild erheblich prägen, sind sie wie letztendlich allen markanten Bauwerke für eine automatische Erzeugung weniger geeignet.

8.3 Prozedurale Texturerzeugung

In Kapitel 5 werden Methoden zur prozeduralen Texturerzeugung vorgestellt. Hierbei werden zwei grundlegende Basisalgorithmen vorgestellt: Perlin Noise und der Cellular Algorithmus. In der gängigen Literatur, wie z.B. in [EMP+03], werden vielfältige Einsatzmöglichkeiten prozeduraler Texturen aufgezeigt. Gerade in der Simulation natürlicher Oberflächen ist der Perlin Noise Algorithmus ein Standardverfahren der Computergraphik.
Im Kontext des Echtzeitrenderings stellt sich immer wieder die Frage in wie weit prozedurale Texturen in Echtzeit berechnet werden können. In [EMP+03], [Pha05] und [Eri04] werden Verfahren zur Implementation der Perlin Noise Funktion auf der Graphikhardware beschrieben. Hierbei wird zur Erzeugung der Noise Funktion die Interpolation zwischen Texeln einer Lookup-Textur seitens der Graphikhardware ausgenutzt. Die Erzeugung von Perlin Noise mit mehreren Oktaven resultiert daher in mehreren Textur Lookups, eine Operation welche auch bei modernen Graphikkarten ein deutlicher Performanzfaktor ist. Insgesamt ist die Erzeugung von Perlin Noise im großen Maßstab, d.h. im Kontext dieser Arbeit auf Ebene einer kompletten Landschaft zu teuer, um wirklich sinnvoll eingesetzt werden zu können. Eine Implementation des Cellular Verfahrens auf der GPU war zum Zeitpunkt dieser Arbeit nicht bekannt. Auf Grund des hohen Berechnungsaufwandes, insbesondere der zahlreichen Arrayzugriffe, ist eine derartige Implementation jenseits der Möglichkeiten aktueller Grafikhardware. Es existieren jedoch eine Reihe von prozeduralen Texturen, bei welchen sich eine Implementierung mittels Shader anbietet. In [Roo06] sind hiervon einige zu finden, so z.B. der bekannte *Brick-Shader* oder ein Shader zur Erzeugung von Holztexturen. In dieser Arbeit kommen diese echtzeitfähigen prozeduralen Texturen mangels geeigneter Einsatzmöglichkeiten jedoch nicht zur Verwendung.
In Abschnitt 6.2.2 und 6.3.2 werden Verfahren zur Erstellung komplexerer, prozeduraler Texturen für die Darstellung von Straßen- und Schienenverkehrswegen beschrieben. Hierbei werden Perlin Noise und Cellular Texturen mittels parametrisierter Masken miteinander verknüpft. An diesem Beispiel zeigt sich, dass der Einsatz von prozeduralen Texturen in Bereichen, in denen ein Grundmuster durch variable Ausgangsdaten in verschiedenen Ausprägungen vorkommt, wie dies zum Beispiel bei der Anzahl der Fahrstreifen auf Straßen der Fall ist, Vorteile gegenüber der Verwendung von Fotographien realer Umweltbedingungen bietet. Ein prozedurales Texturierungssystem bietet in solchen Fällen eine schnelle, flexible und an die Bedürftnisse der Modelierung anpassbare Lösung. Die Erstellung von realistisch aussehenden prozeduralen Texturen ist jedoch relativ schwierig und unintuitiv, die Ergebnisse wirken mitunter sehr künstlich und sind von einer photorealistischen Erscheinung weit entfernt. Oft kann ein gewünschtes Ergebnis nur durch Testen verschiedenster Parameterkombinationen erreicht werden.

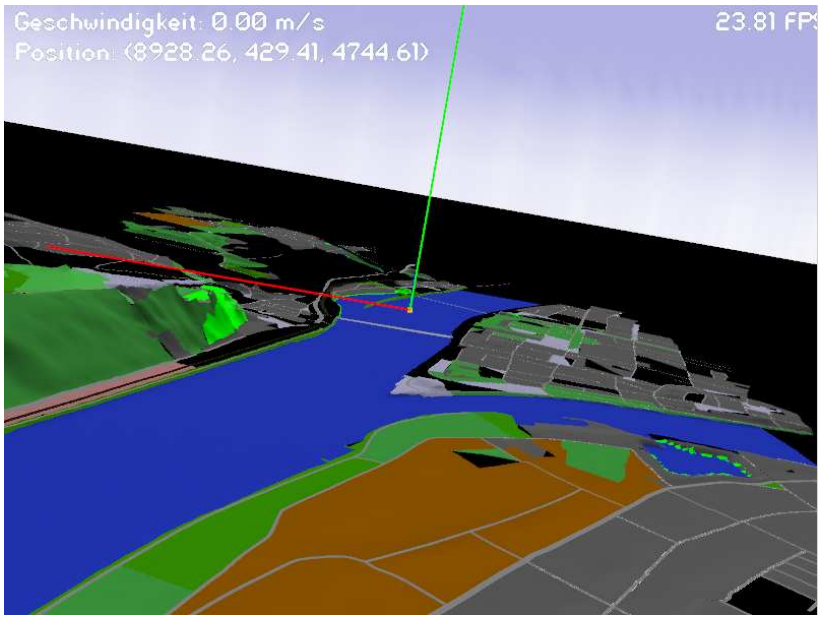
Sinnvoll kann hier die Kombination von realen Fotografien mit einem prozeduralen Textursystem sein, um die Parametrisierbarkeit einer prozeduralen Textur mit dem Realismus einer Fotografie zu verbinden. Denkbar wäre ein solcher Ansatz z.B. für die Erzeugung der in Kapitel 6.2.2 dargestellten Schienenverkehrstexturen, dies wurde jedoch im Rahmen dieser Arbeit nicht untersucht.

8.4 Prozedurale Vegetation

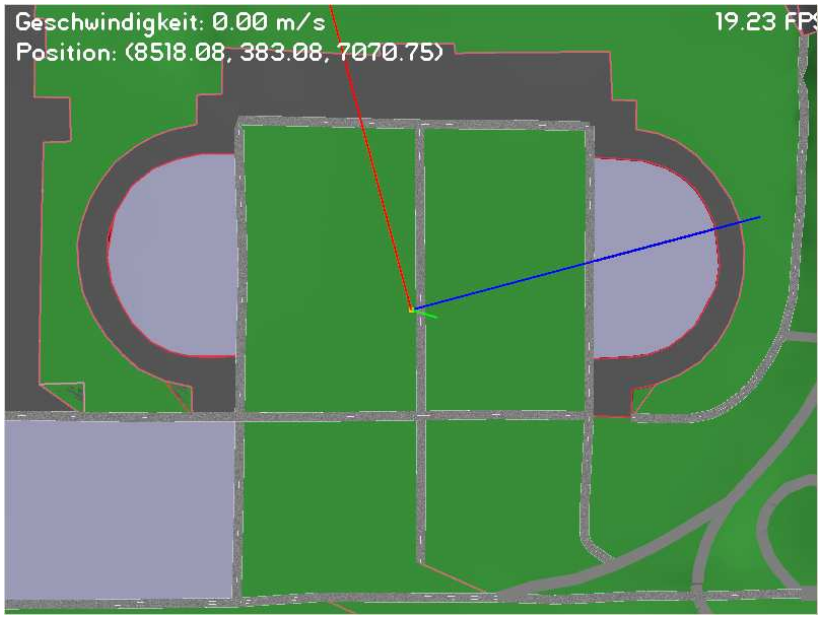
In Kapitel 7 werden Möglichkeiten zur Erzeugung prozeduraler Vegetation und deren Platzierung auf dem Terrain aufgezeigt. Da die Realisierung des Vegetationssystems aus Mangel an Zeit nur ein relativ rudimentäres Niveau erreicht hat, und der Hauptfokus dieser Arbeit nicht auf der Implementierung eines Vegetationssystems lag, sind die hier gezogenen Schlussfolgerungen eher theoretischer Natur.
In Abschnitt 7.2 werden die Grundlagen für ein parametrisierbares Modell zur Platzierung von Vegetation beschrieben. Die dort genannten Parameter wie Höhe des Geländes, Hangneigung und Hangausrichtung etc. lassen sich aus dem erstellten Geländemodell ableiten. Zusammen mit der im ATKIS-Basis DLM enthaltenen Information über Vegetationsflächen und deren grober Klassifizierung (vgl. Abschnitt 7.1) steht eine solide Datengrundlage zur parametrisierten, automatischen Platzierung von Vegetation im Gelände zur Verfügung. Unter der Objektart 4201 (Baum) wird im ATKIS-Basis DLM ein "*einzeln stehender Baum, der als Naturdenkmal eingestuft oder landschaftsprägend ist*" modelliert. Hierbei wird lediglich zwischen Laub- und Nadelholz unterschieden, da aufgrund der zu erwartenden Bekanntheit eines Naturdenkmals sicherlich die genaue Art des Baumes (Eiche, Fichte etc.) mit erfasst werden müsste, um hieraus eine realistische Visualisierung mit Wiedererkennungseffekt zu realisieren. Bei den als Objektart 2227 erfassten Grünanlagen ist eine automatische Generierung nur dann sinnvoll, wenn hiermit keine besonders markante und speziell bepflanzte Gartenanlage modelliert ist, da die Objektart 2227 lediglich als Fläche modelliert ist, ohne genaue Informationen über die Bepflanzung zu enthalten. Flächen mit landwirtschaftlicher Vegetation sind in den Vermessungsdaten in einer gewissen Variation modelliert, so wird zum Beispiel die Objektart 4109 (Sonderkultur) durch das Attribut KLT (Kulturart) in *Baumschule*, *Hopfenfeld*, *Weingarten* und *Obstbaumplantage* untergliedert. Da z.B. der Weinbau stark landschaftlich prägende Eigenschaften besitzt ist diese Datenlage sehr günstig, um ein wesentliches Landschaftsmerkmal automatisiert zu erstellen.
In Abschnitt 7.3 wird auf die Erstellung von konkreten Pflanzen auf der Basis von Lindenmeyer Systemen eingegangen. Hierbei wird gezeigt, dass die grundlegenden Wuchsformen (vgl. Abschnitt 7.1) von Pflanzen durch entsprechende Verzweigungen der Sprossachsen mittels L-Systemen modelliert werden können.

8.4 Prozedurale Vegetation

Durch die Modellierung von Tropismuseinflüssen können Vegetationsmodelle auf der Basis von L-Systemen in Interaktion mit ihrer Umgebung treten. Durch den Tropismusvektor kann die Einwirkung äusserer Umwelteinflüsse auf die Vegetation modelliert werden, was einen Vorteil gegenüber der Verwendung von statischen Modellen darstellt. In der Literatur werden mitunter äusserst komplexe Vegetationsmodelle auf der Basis von L-Systemen beschrieben. Die Möglichkeiten zur Erstellung eines exakten, auf biologischen Grundlagen basierenden parametrisierbaren Vegetationsmodells sind hier sehr vielfältig. Die Problematik von L-Systemen besteht in der Tatsache, dass für jede zu modellierende Pflanze die entsprechende Grammatik angegeben werden muss, was im Kontext von L-Systemen auch als Interferenzproblem bezeichnet wird. Hier zeigt sich wieder das Grundproblem von prozedural modellierten Aspekten: Die Ermittlung der Parameter zur Erreichung eines bestimmten Ergebnisses ist nicht intuitiv und setzt eine gewisse Erfahrung von Seiten des Modellierers vorraus.
Die Visualisierung der Pflanzenmodelle mittels Billboards stellt ein Standardverfahren der Computergraphik dar. Besonders bei naher Betrachtung wirkt diese Darstellung relativ unrealistisch, da der Aufbau der Bilboardgeometrie sichtbar wird. Für eine qualitativ hochwertige Darstellung sind hier LOD-Verfahren unbedingt erforderlich. Sollen Pflanzen aus der Nähe dargestellt werden, so ist zumindest die Modellierung der Sprossachsen durch zylinderförmige Geometrie als Option in Betracht zu ziehen. Ferner müssen durch das in der Bilboard-Technik begründete Alpha-Blending die zu rendernden Vegetationsobjekte entlang der Blickrichtung sortiert werden, um eine korrekte Darstellung zu erreichen, was besonders bei einer großen Zahl von Objekten eine relativ teure Operation darstellt.



(a) Ueberblick



(b) Flächen mit funktionaler Unterscheidung

Abbildung 60: digitales Landschaftsmodell

8.4 Prozedurale Vegetation

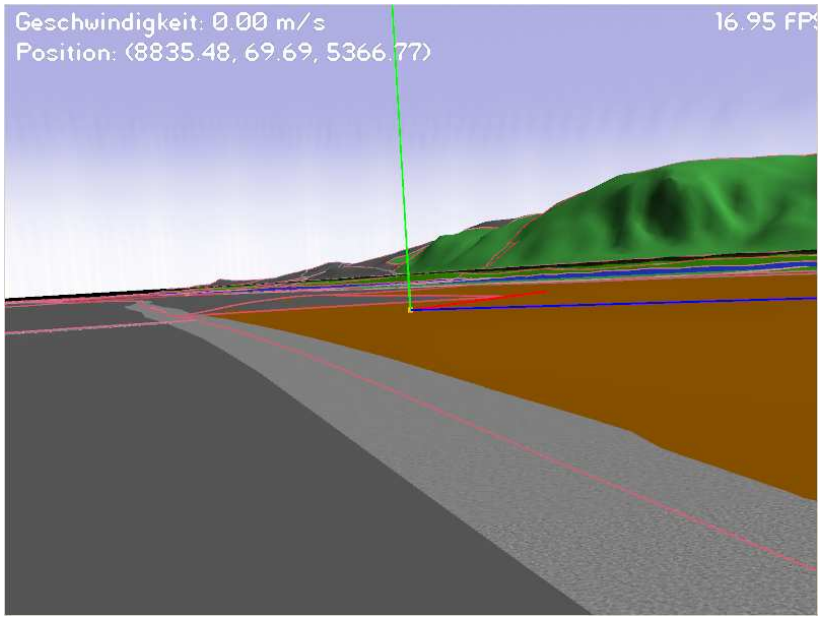


Abbildung 61: Straße verläuft auf der Grenzlinie

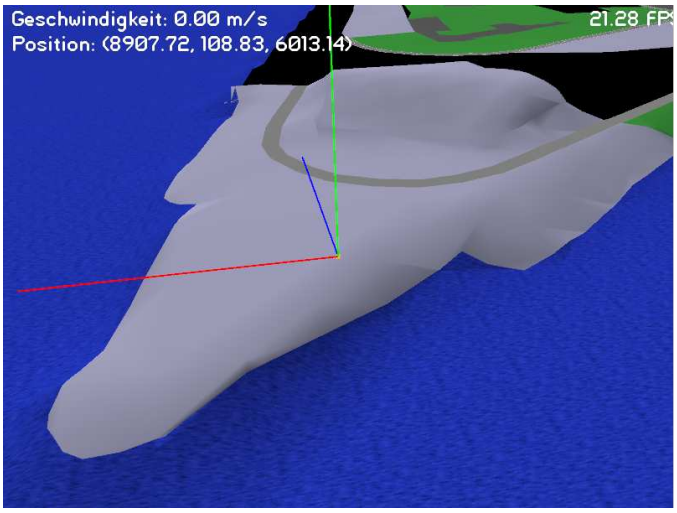
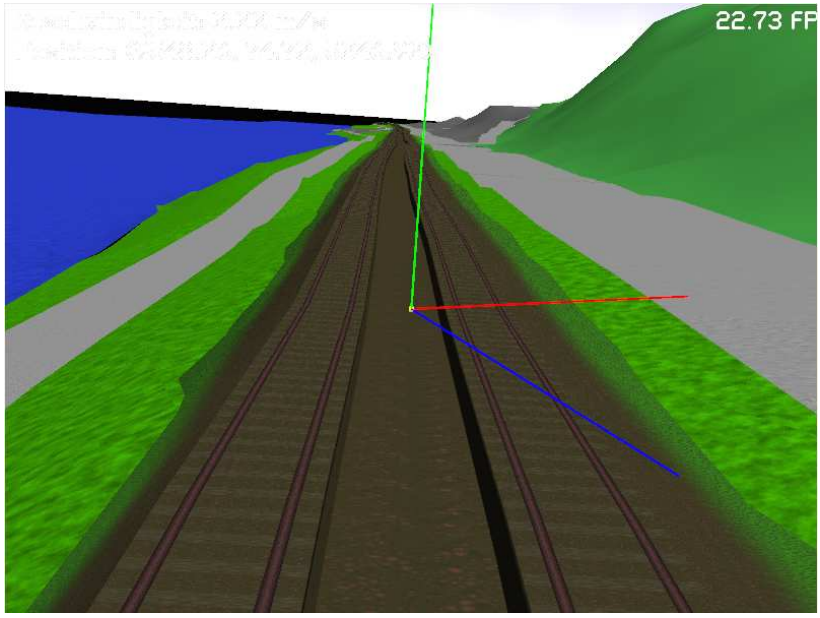
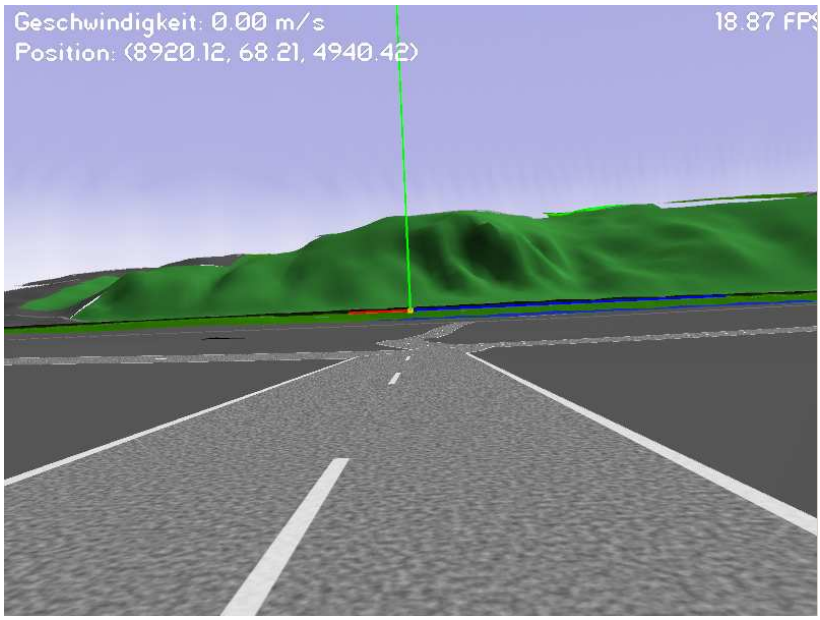


Abbildung 62: verrauschte DGM Daten



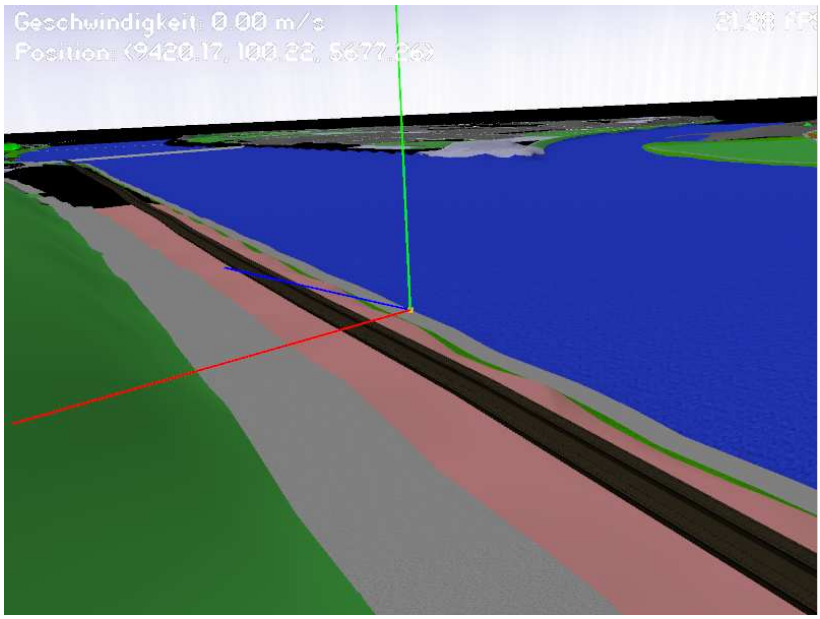
(a) Gleisanlage



(b) Straße

Abbildung 63: Verkehrswege

Abbildung 64: Fehlen von Kreuzungen



(a) fehlende Fahrbahnmarkierung

(b) fehlerhafte Fahrbahnmarkierung am kurfürstlichen Schloss

Abbildung 65: Probleme mit Fahrbahnmarkierungen

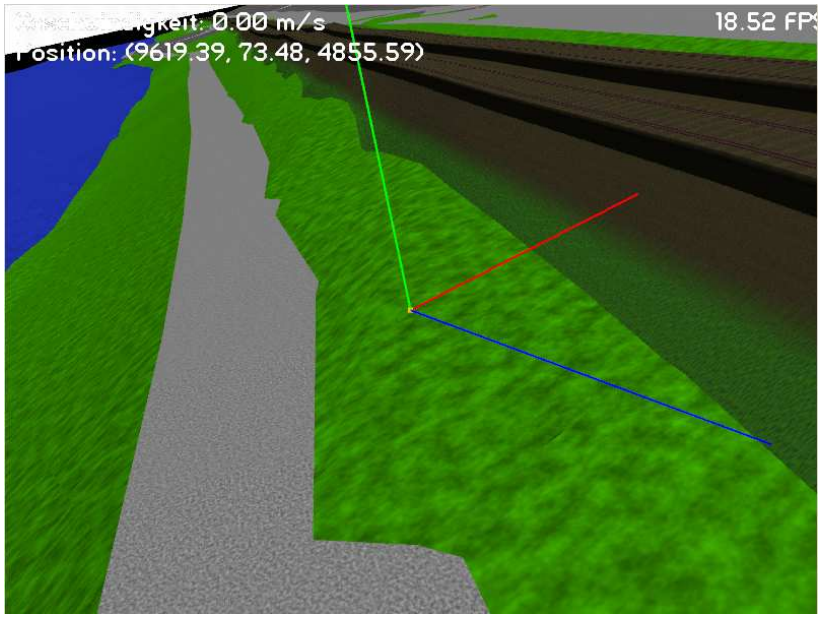


Abbildung 66: Durchdringung von Weg und Gelände

9 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde ein System zur Erzeugung eines dreidimensionalen digitalen Landschaftsmodells aus den Vermessungsdaten ATKIS-Basis DLM und dem Digitalen Geländemodell (DGM) entwickelt. Das Landschaftsmodell lässt sich mittels eines 3D-Echtzeitrenderingsystems visualisieren. Es wurde untersucht, in wie weit sich Geländemerkmale automatisiert durch ein parametrisierbares Modell erstellen lassen. Verfahren zur prozeduralen Geometrie- und Texturerzeugung wurden hauptsächlich bei der automatischen Generierung von Verkehrswegen angewendet. Weiterhin wurde ein grundlegendes Modell zur parametrisierbaren prozeduralen Vegetationserzeugung auf der Basis von Lindenmeyer Systemen vorgestellt. Die Arbeit sollte die Möglichkeiten der verwendeteten Vermessungsdaten im Kontext des Terrain-Renderings aufzeigen und die Grenzen der automatisierten Modellerstellung auf Grundlage dieser Datenbasis ausloten.
Sicherlich stellen die in dieser Arbeit erzielten Ergebnisse lediglich einen ersten Schritt in der Erstellung eines visuell überzeugenden Abbildes der realen Landschaft dar. Auf zahlreiche Aspekte konnte auf Grund der zeitlichen Rahmung einer Diplomarbeit nicht intensiver eingegangen werden. Daher soll an dieser Stelle ein Einblick in die noch zu untersuchenden Fragestellungen und Probleme gegeben werden.

9.1 Vermessungsdaten

Eine Erfassung des kompletten ATKIS Datenmodells und die Erstellung entsprechender Datenstrukturen für alle im ATKIS-Basis DLM enthaltenen Objektarten und ihren zugehörigen Attributen wurde in dieser Arbeit nicht vorgenommen und sollte daher Ziel zukünftiger Arbeiten sein. Vor allem die Behandlung komplexer ATKIS-Objekte sowie die Verwendung der im ATKIS-Datenmodell enthaltenen Referenzen von Objekten untereinander ist Grundlage für die Erstellung von Brücken- bzw. Überführungsbauwerken.
Eine Konvertierung der fachlichen Information vom EDBS Format in ein geeignetes Datenbanksystem, wie es z.B. in [Fal00] vorgeschlagen wird, würde deutliche Vorteile für zukünftige, auf den Daten aufbauenden Geoinformationssysteme bieten.
Eine Funktionalität mit großem Nutzen wäre mit Sicherheit die Erstellung einer Nachbarschaftstopologie aus den ATKIS-Basis DLM Daten, was auch in Kapitel 8 Erwähnung findet.
Ferner würde eine exakte Höhenermittlung auf Basis des triangulierten Geländereliefs gerade im Bereich der automatischen Verkehrswegerzeugung in einem deutlichen Zuwachs an Präzission resultieren, da sich besonders an dieser Stelle die bilineare Interpolation der Höhendaten aus dem DGM als zu ungenau erwie-

sen hat.

9.2 Landschaftsmodell

Das in dieser Arbeit vorgestellte Verfahren zur Erzeugung eines Triangulierten Oberflächenreliefs hat zwar seine grundsätzliche Eignung unter Beweis gestellt, jedoch würde der Vergleich mit verschiedenen anderen Verfahren sicher einem weiteren Erkenntnisgewinn bezüglich des optimalen Triangulationsverfahrens beitragen.
Ein weiterer wichtiger Punkt ist die Persistenzmachung der Daten in einem geeigneten Format, was sowohl den Anforderungen eines Echtzeitrenderingsystems gerecht wird, als auch fachliche Information der Vermessungsdaten beinhaltet. Eine Kombination von Standard 3D-Austauschformaten wie VRML mit einem Datenbankmanagementsystem ist hier ebenso denkbar wie die Verwendung von speziell für den Bereich Geoinformationssysteme entwickelten Formaten wie GML.
Ein eng mit der Persistenz verbundenes Thema ist im Kontext von Terrainrendering das sog. *paging*, d.h. das dynamische Nachladen der Daten. Besonders bei großen Datenmengen müssen hier sehr ausgefeilte Algorithmen zum Einsatz kommen. Momentan sind in dem erstellten Modell lediglich Verkehrswege und ein rudimentäres Vegetationssystem als Landschaftsobjekte enthalten. Das Hinzufügen von Gebäuden ist hier sicherlich der nächste Schritt in Richtung eines realistischen Abbildes der Landschaft. Für die prozedurale Erzeugung von Gebäudemodellen sind die in den Vermessungsämtern vefügbaren Katasterdaten, insbesondere die *Automatisierte Liegenschaftskarte (ALK)* als Datengrundlage von Bedeutung. In [Bob] und [bec05] werden die Möglichkeiten und Probleme der Integration von Gebäudegrundrissen aus ALK-Daten dargestellt.
Da das Vegetationsmodell in dieser Arbeit eher theoretisch betrachtet wurde, macht eine Implementation der hier gezeigten Ansätze Sinn. Darauf aufbauend können komplexere, durch biologische Grundlagen motivierte Vegtationsmodelle untersucht werden.
Für das effiziente Rendering des Terrain sind aufgrund der großen Datenmengen Level of Detail Verfahren unumgänglich, um eine performante Darstellung zu erreichen. Durch aus den Vermessungsdaten abgeleitete Form des Landschaftsmodells muss hier ein von den Standardalgorithmen abweichendes bzw. eine Modifikation der Standardverfahren zum Einsatz kommen.
Im Rahmen der Texturerzeugung wurden in dieser Arbeit lediglich grundlegende Algorithmen vorgestellt. Eine Implementation von anspruchsvolleren Texturen für Wasser, athmosphärische Effekte, Gesteinsformen etc. würde die visuelle Qualität des Modelles deutlich aufwerten. Die Einbeziehung der bereits vorhandenen Luftbilddaten in die Texturgenerierung sollte ebenfalls in Erwägung gezogen werden.

Literatur

[AdV03] AdV-Arbeitsgruppe ATKIS: *ATKIS Objektartenkatalog (ATKIS - OK)*. Version 3.2. 2003

[AL90] ARISTID LINDENMAYER, Przemyslaw P.: *The Algorithmic Beauty of Plants*. Springer Verlag, New York, 1990

[AMH02] AKENINE-MÖLLER, Tomas ; HAINES, Eric: *Real-Time Rendering*. A K Peters, Ltd., 2002

[Bae06] BAERZ, Robert: *Effizientes Rendering von Landschaften*, Universität Koblenz-Landau, Diplomarbeit, 2006

[bec05] GERD BECKER, Hans. *Einbeziehung vorhandener Daten in den Aufbau und die Aktualisierung von 3D-Stadtmodellen aus der Sicht der Vermessung*. 2005

[Bob] BOBRICH, Joachim. *Zur Integration von ALK-gebäudedaten in ATKIS-Datenbestände*

[Bre] BREIER, Florian. *L-Systeme und andere künstliche Pflanzen*

[Böt01] BÖTTCHER, Arne. *Das Lernprogramm Lily*. Website, http://olli.informatik.uni-oldenburg.de/lily/LP/start.html. 2001

[Div06] DIVERSE. *Perlin Noise*. DGL Wiki, http://wiki.delphigl.com/index.php/Perlin_Noise. 2006

[Ebe01] EBERLEY, David H.: *3D Game Engine Design*. Morgan Kaufmann Publishers, 2001

[Ebe02] EBERLY, David. *Triangulation by Ear Clipping*. Website, http://www.geometrictools.com. 2002

[Ebe05] EBERLEY, David H.: *3D Game Engine Architecture*. Morgan Kaufmann Publishers, 2005

[Eli03] ELIAS, Hugo. *Perlin Noise*. Website, http://freespace.virgin.net/hugo.elias/models/m_perlin.htm. 2003

[EMP+03] EBERT, Davis S. ; MUSGRAVE, F. K. ; PEACHEY, Darwyn ; PERLIN, Ken ; WORLEY, Steven: *Texturing and Modeling - A Procedural Approach*. Morgan Kaufmann Publishers, 2003

[Eri04] ERIKSSON, Erik. *Noise in Real-time 3D Graphics*. 2004

[Fal00] FALKE, Stephan: Entwurf und Implementierung einer räumlichen Datenbank für ATKIS-Daten mit Datenimport / Institut für Informatik B: Datenbanken und Informationssysteme, Universität Hannover. 2000. – Forschungsbericht

[Fuh98] FUHRER, Heinz: *Feldmessen und Kartographie*. Justus Perthes Verlag Gotha GmbH, 1998

[Ges67] GESETZGEBER. *Eisenbahn-Bau- und Betriebsordnung (EBO)*. Website, http://www.wedebruch.de/gesetze/betrieb/ebo1.htm. 1967

[Hai92] HAINES, E.: Fastest point in polygon test. In: *Ray Tracing News* 5 (1992), September, Nr. 3. – Electronic journal

[Ham] HAMMES, Johan. *Modeling of ecosystems as a data source for real-time terrain rendering*

[Lan93] Landesvermessungsamt Sachsen: *Allgemeine Datenbeschreibung ATKIS*. 1993

[Leh05] LEHMANN, Anke: *Echtzeitvisualisierung einer dynamischen Landschaft für Erosionssimulation*, Technische Universität Dresden, Institut für Software- und Multimediatechnik, Diplomarbeit, 2005

[Mar02] MARGHIDANU, Mircea. *Fast Computation of Terrain Shadow Maps*. Website, http://www.gamedev.net/reference/articles/article1817.asp. 2002

[MO06] MCCLELLAN, Matt ; OWENS, Kipp. *Alternatives to Using Z-Bias to Fix Z-Fighting Issues*. Website, http://www.intel.com/cd/ids/developer/asmona/eng/segments/games/168495.htm?page=1. 2006

[Nad] NADOR, Rachel. *INTERSECTION CONSTRUCTION*. Website, http://www.vterrain.org/Culture/RFP/intersections/index.html

[Per99] PERLIN, Ken. *Making Noise*. Website, http://www.noisemachine.com/talk1/index.html. 1999

[Pha05] PHARR, Matt (Hrsg.): *GPU Gems 2*. Addison-Wesley, 2005

[Roo06] ROOST, Randi J.: *OpenGL Shading Language*. Second Edition. Addison-Wesley Longman, Amsterdam, 2006

[Scont] SCOTT, Jim. *Making Cellular Textures*. Website, http://www.blackpawn.com/texts/cellular/default.html. unbekannt

[Sei05] SEIDEL, Ingo: Pflanzenmodellierung mit prozeduralen Methoden und L-Systemen / WSI/GRIS, Universität Tübingen. 2005. – Forschungsbericht

[Sen] V. SENGBUSCH, Peter. *Botanik Online - Blätter*. Website, http://www.biologie.uni-hamburg.de/b-online/d02/02c.htm

[Ulr02] ULRICH, Thatcher: Rendering Massive Terrains using Chunked Level of Detail Control / Oddworld Inhabitants. 2002. – Forschungsbericht

[Unk] UNKNOWN. *What is a Linear Congruential Random Number Generator?* Website, http://probabilitynet.com/lincong.htm

[Ver93] Vermessungs- und Katasterverwaltung Rheinland-Pfalz: *Dokumentation der Vermessungs- und Katasterverwaltung Rheinland-Pfalz zum Datenaustausch*. 1993

[Zuc01] ZUCKER, Matt. *The Perlin noise math FAQ*. Website, http://www.cs.cmu.edu/ mzucker/code/perlin-noise-math-faq.html. 2001