World Weather 3D

Henning Wenke, Ralf Kunze, Oliver Vornberger
Universität Osnabrück

Zusammenfassung

Um sich dynamisch ändernde Daten im Internet, wie z.B. Wetterdaten, zu visualisieren und zu präsentieren ist eine interaktive Anwendung erforderlich. Die Anwendung sollte intuitiv bedienbar sein und muss über Animationsmöglichkeiten verfügen, um die Änderungen über die Zeit deutlich zu machen. In diesem Paper stellen wir eine Möglichkeit vor, mit der solche Daten in einem Java Applet dreidimensional dargestellt werden. Dazu sind Echtzeitanimationstechniken erforderlich, da die sich ständig ändernden Daten aufgrund der Interaktivität nicht vorberechnet werden können. Dafür sind effektive Algorithmen nötig, da die Rechenzeit auf der Clientseite gering gehalten werden muss. Für die Isoliniendarstellung stellen wir einen erweiterten Marching Squares Algorithmus vor, der zum einen sehr effizient ist und zum anderen zusammenhängende Polygone in linearer Laufzeit in Abhängigkeit der Gitterzellen bilden kann, sowie weitere Techniken, um z.B. die Temperatur anzuzeigen.

1 Einleitung

Wer kennt nicht Google Earth (Google, 2006), das Programm, mit dem Satellitenbilder der Erdoberfläche im Internet auf eindrucksvolle Weise angezeigt werden können. Es ist dabei sowohl die Betrachtung der gesamten Erde als auch von Ausschnitten aus einem beliebigen Winkel möglich. Bewegt sich die Kamera näher an die Erde, wird diese mit immer detaillierteren Oberflächentexturen versehen, welche dynamisch über das Netz nachgeladen werden. Google Earth bietet allerdings keine Möglichkeit sich über die Zeit ändernde Daten mittels Animation zu veranschaulichen.
Gegenstand dieses Papers ist ein Projekt zur 3D-Visualisierung und Animation von orts- und zeitabhängigen Klima- und Wetterdaten im Web, welches gegenwärtig an der Universität Osnabrück läuft. Dafür ist ein Programm entwickelt worden, welches wie Google Earth die gewünschten Daten über das Internet bezieht und zusätzlich um die zeitliche Dimension erweitert wurde. Es handelt sich dabei um ein Java-Applet, das mithilfe von JOGL (JOGL, 2007) die OpenGL API (OpenGL, 2007) zum effizienten Rendern der 3D-Grafik in Echtzeit nutzt. Die aktuell verwendeten Wetterdaten stammen von Raymarine.com und liegen im GRIB-File Format vor. Sie liefern eine dreitägige Wettervorhersage und werden täglich aktualisiert.

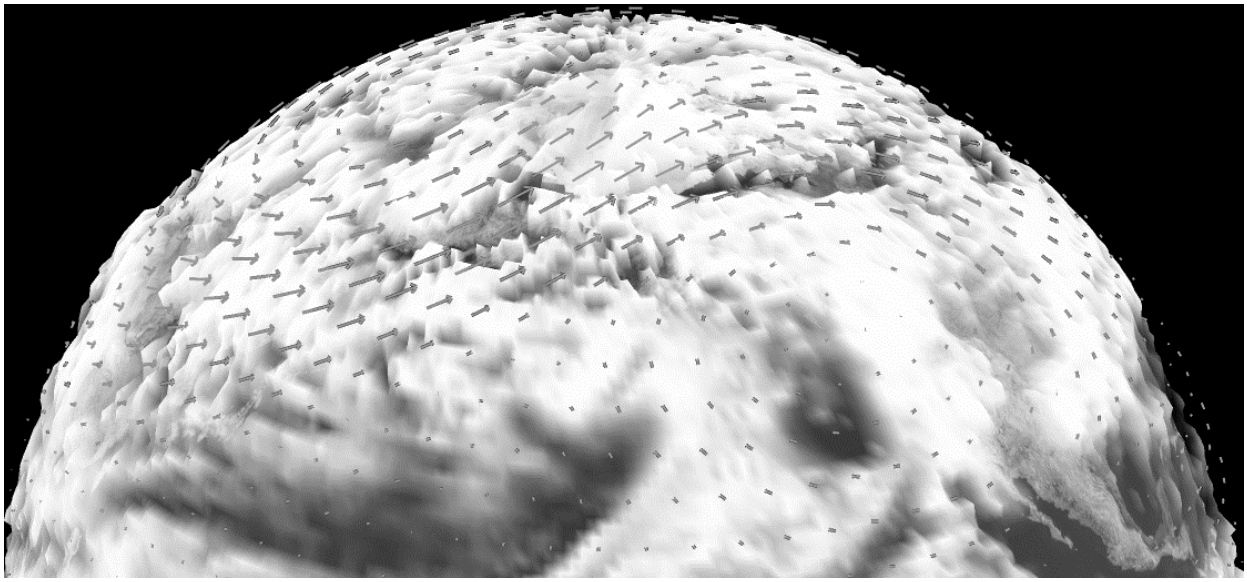

Abbildung 1: Kombinierte Darstellung von Bewölkung und Wind

Einzelne GRIB-Files überdecken zumeist nur Teile der Erdoberfläche (etwa Europa) und müssen geeignet kombiniert dargestellt werden, um den Großteil der Erdoberfläche abzudecken. Für die unterschiedlichen Wetterdaten wurden jeweils geeignete Visualisierungstechniken gewählt. Die einzelnen Datentypen können auch kombiniert dargestellt werden (siehe Abb.1).

Bislang existiert kein vergleichbarer Ansatz, der die grundlegenden Funktionalitäten von Google Earth bietet (Wahl der Kameraposition, selektives Laden der Daten je Bildausschnitt und Zoomstufe) und mit Animationen über die Zeit kombiniert. Ferner lässt die plattformunabhängige Implementation als Java-Applet die Einbettung in eine Website zu. Der aktuelle Stand der Implementation ist unter http://project.informatik.uni-osnabrueck.de/hwenke/WorldWeather3D.html zu betrachten.

1 Daten

1.1 Ausgangsdaten

Die verwendeten georeferenzierten Daten lassen sich in statische (Erdtextur, Küstenlinien, Ländergrenzen,...) und dynamische (Wetter, Klima) Daten einteilen. Da der Fokus auf der Animation liegt, wird an dieser Stelle lediglich auf die dynamischen Daten eingegangen.

1.2 Rasterdaten

Die Wetterdienste weltweit übertragen ihre Daten in der Regel mittels GRIB (Gridded Binary) Files. In GRIB-Files werden Wetterdaten in einem regelmäßigen rechteckigen Gitter gespeichert, welches sich einer bestimmten geographischen Region zuordnen lässt. Auch Klimadaten und andere Daten werden in GRIB-Files abgespeichert. Die eigentlichen Daten werden innerhalb eines GRIB-Files in einzelnen Records strukturiert. Ein Record beinhaltet eine bestimmte Kenngröße (Temperatur, Regenmenge, etc.) zu einem festgelegten Zeitpunkt. Ein GRIB-File kann aus beliebig vielen Records bestehen.

1.3 Datenstruktur

Ein zentrales Problem des Projekts liegt in der begrenzten Bandbreite, mit der die Daten nachgeladen werden können. Zur folglich wünschenswerten Minimierung der zu ladenden Datenmenge kommen im Wesentlichen die beiden folgenden Mechanismen zum Einsatz:

- Aktualisierung ausschließlich von sichtbaren Bereichen der Erdoberfläche
- Auswahl der Daten in einer in Abhängigkeit von der Zoomstufe optimalen Auflösung

Dazu werden die in den GRIB-Files enthaltenen Daten in kleinere Rechtecke eingeteilt und in verschiedenen Auflösungen als Blob-Object in komprimierter Form in eine MySQL-Datenbank eingefügt. Bei einer Suchanfrage werden alle Rechtecke einer bestimmten Auflösung geladen, die zumindest teilweise im Sichtfenster liegen und den angefragten Zeitpunkt repräsentieren. Überlappen sich mehrere GRIB-Files so wird nur das jeweils am weitesten oben liegende Rechteck geladen. Diese Einteilung in atomare Rechtecke erlaubt effizientes Auslesen von zeitlichen und räumlichen Teilmengen der Daten bei geringer Beanspruchung der Datenbank. Der Ansatz ermöglicht es, die Menge der zu übermittelnden Daten in etwa konstant zu halten, indem diese in einer Dichte angefordert werden, die umgekehrt proportional zur Größe des sichtbaren Bereichs ist. Folglich resultiert eine höhere Maximalauflösung auf dem Server nicht in einer gestiegenen Beanspruchung der Bandbreite, sondern ermöglicht weitere Zoomstufen. Ein herkömmlicher DSL 1000 Anschluss ist für eine vollständig flüssige Animationsgeschwindigkeit bereits mehr als ausreichend.

2 Darstellung

Die von der aktuellen Version des Applets darstellbaren Wetterdaten sind Luftdruck, Temperatur und Wind. Allerdings ist die Winddarstellung noch nicht ausgereift, weshalb sie hier nicht erläutert wird.

2.4 Luftdruck

Luftdruck wird im Allgemeinen durch Isolinien dargestellt. In diesem Kapitel wird ein Verfahren zu deren effizienter Erzeugung vorgestellt.

2.4.1 Marching Square Algorithmus

Das Marching Square Verfahren (Lorensen et al., 1987) dient zur Visualisierung zweidimensionaler Daten mittels Isolinien. Die Daten müssen als zweidimensionales skalares Punktfeld vorhanden sein. Aus jeweils vier Punkten wird ein Quadrat definiert, wobei an den vier Ecken der Box die skalaren Werte eingetragen sind. Für einen gegebenen Isowert wird untersucht, welche der Eckpunkte über und unter dem Isowert liegen. Liegt bei einer Kante ein Wert über und einer unter dem Isowert, so existiert ein Schnittpunkt, der durch lineare Interpolation bestimmt wird. Mit diesen Schnittpunkten werden die einzelnen Linien gebildet. Insgesamt werden 16 verschiedene Anordnungen unterschieden (Abbildung 2). Wenn alle Gitterzellen für einen Isowert der Reihe nach abgearbeitet wurden, erhält man

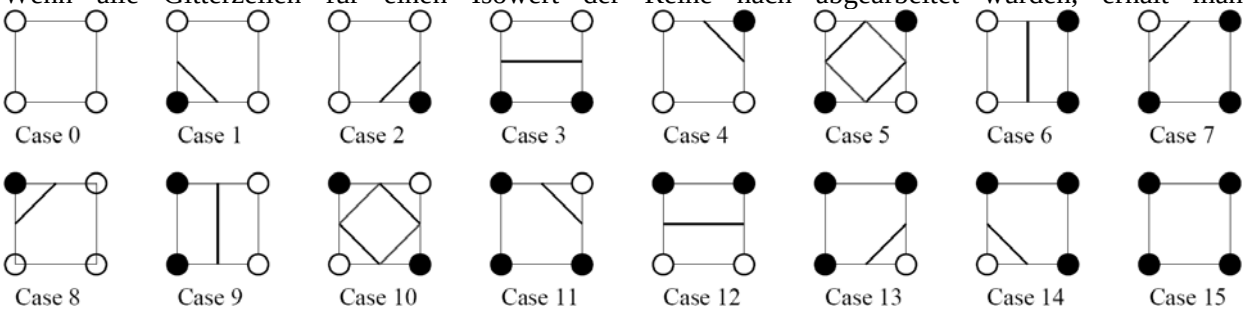

Abbildung 2: 16 Fälle beim Marching Square Verfahren

einzelne unzusammenhängende Liniensegmente der zugehörigen Isolinien. Aufgrund der lokalen Arbeitsweise ist der Marching Squares Algorithmus sehr effizient, sowohl in der Zeichengeschwindigkeit als auch in der Speicherbelastung. Es werden in der Ausgabe geschlossene Konturlinien gerendert, die allerdings nur aus Liniensegmenten bestehen und keine zusammenhängenden geometrischen Objekte darstellen. Aus diesem Grund können die Isolinien nicht geglättet und beschriftet werden.

2.4.2 Marching Square SE

Um die beschriebenen Defizite des Marching Square Algorithmus zu beheben, ist im Rahmen dieser Arbeit ein Verfahren entwickelt worden, mit dessen Hilfe echte Polygone auf Basis des Marching Square Algorithmus gebildet werden können.

Arbeitsweise
Ausgangssituation der Erweiterung des Marching Square Algorithmus sind die Boxen mit zusammenhängenden Liniensegmenten, die der Marching Square Algorithmus für einen Isolevel erzeugt hat. Diese enthalten einen Index für den Boxtyp, zwei zunächst nicht initialisierte Linien und pro Kante einen ggf. undefinierten Schnittpunktindex. Jede Box hat vier Kanten und damit vier direkte Nachbarn (siehe Abb.3 links). Des Weiteren können Methoden definiert werden, die in Abhängigkeit des Boxtyps für jede in der Box enthaltene Linie bestimmen, in welchen Nachbarboxen sie fortgesetzt wird (Abb.3 Mitte). Aufgrund dieses Wissens entfällt eine aufwändige Suche nach einer Fortsetzung der Linie. Der Algorithmus verarbeitet die Boxen zeilenweise von oben nach unten. In den einzelnen Zeilen werden die Boxen von links nach rechts betrachtet. Bedingt durch diese Verarbeitungsrichtung sind die Boxen links und über

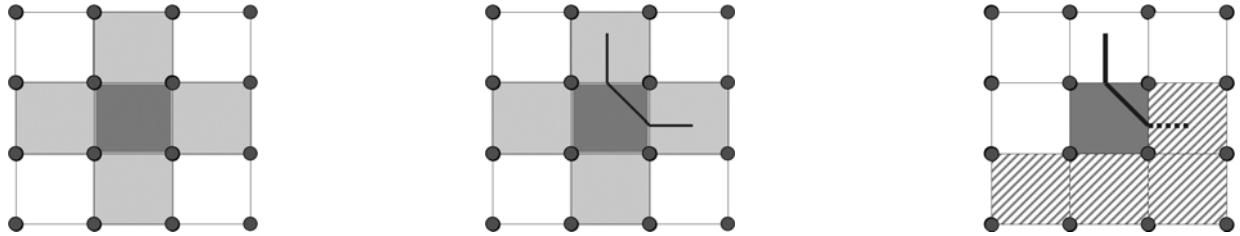

Abbildung 3: Nachbarschaftsbeziehungen der aktuellen Box (dunkelgrau).Links: Die vier direkten Nachbarn (hellgrau). Mitte: Rekonstruktion des Linienverlaufs anhand des Boxtyps. Rechts: Die bereits fertigen Boxen (weiß) und die noch nicht betrachteten (schraffiert).

der jeweils aktuellen Box zu diesem Zeitpunkt bereits fertig verarbeitet (Abb.3 rechts). Aus diesem Grund werden Vergleiche nur mit diesen Vorgängerboxen durchgeführt. In Abhängigkeit des Boxtyps der jeweils aktuellen Box wird für jede enthaltene Linie eine der folgenden Aktionen durchgeführt:

- **Neue Linie**

  Eine neue Linie muss immer dann erzeugt werden, wenn eine Box ein Liniensegment enthält, welches keine gemeinsame Kante mit einer bereits verarbeiteten Box hat und folglich nicht an eine existierende Linie angehängt werden kann (siehe Abb.4 links).

- **Id einhängen**

  Enthält eine Box ein Liniensegment, welches genau eine gemeinsame Kante mit einer bereits bearbeiteten Box hat, so kann es an eine Linie in dieser Box angehängt werden (siehe Abb.4 Mitte).

- **Linien vereinigen**

  Enthält eine Box eine Linie, welche zwei bereits untersuchte Boxen verbindet, so muss hier der Linienzug geschlossen werden(siehe Abb.4 rechts).

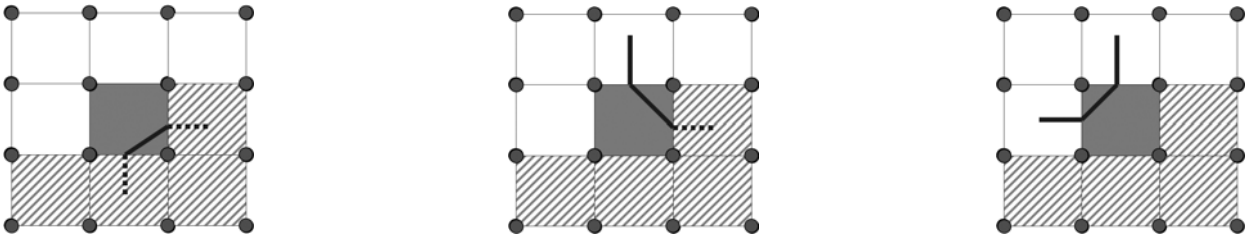

Abbildung 4: In Abhängigkeit des Boxtyps wird entschieden, wie mit der Linie der aktuellen Box (dunkelgrau) zu verfahren ist. Durchgezogene Linien aus fertigen Boxen (weiß) existieren bereits, gestrichelte Linien aus noch nicht verarbeiteten Boxen (schraffiert) existieren noch nicht. Links: Erzeugen einer neuen Linie. Mitte: Anhängen an bestehende Linie. Rechts: Schließen einer Linie.

Datenstruktur zur Linienrepräsentation

Beim Einfügen eines Elements oder Anhängen einer anderen Linie kann das richtige Linienende nicht in Abhängigkeit von der Boxstruktur festgelegt werden. Daher muss diese Aufgabe von der Linie selbst übernommen werden. Dazu wird ausgenutzt, dass zusammengehörige Liniensegmente benachbarter Boxen in einem Punkt enden, dessen Index bereits beiden Boxen bekannt ist. Außerdem muss die Linie unabhängig von der Reihenfolge sein, mit der die Indices eingefügt wurden, damit bei der Vereinigung zweier unterschiedlich gerichteter Linien keine elementweise invertiert werden muss. Die Lösung ist eine doppelt verlinkte ungerichtete Liste ungerichteter Knoten (siehe Abb.5). Darin hat, von den Randknoten

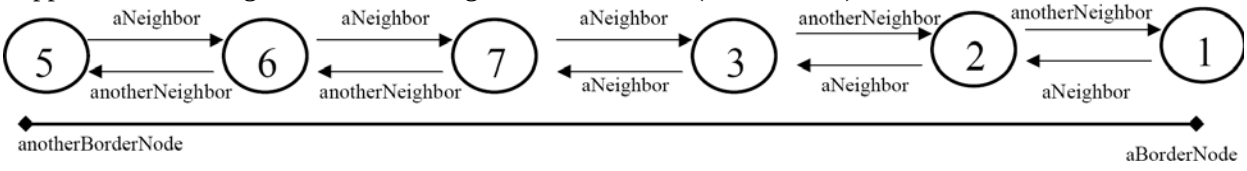

Abbildung 5: Beispiel einer doppelt verlinkten ungerichteten Liste ungerichteter Knoten.

abgesehen, jeder Knoten zwei gleichwertige Nachbarknoten. Durch die doppelte Verlinkung kann die Liste an beiden Enden an eine andere Liste angehängt werden. Das wäre bei doppelt verlinkten gerichteten Knoten zwar auch möglich, jedoch sind nach dem Anhängen an eine andere Liste eventuell Vorgänger und Nachfolger aller Knoten vertauscht. Wird dagegen auf eine Reihenfolge verzichtet, muss nur der Auslesemechanismus angepasst werden, indem willkürlich an einem Ende angefangen wird und das jeweils nächste Element daran erkannt wird, dass es nicht das Vorherige ist. Vor einer Verknüpfungsoperation zweier Listen wird zunächst jeweils das Ende mit dem gemeinsamen Index ausgewählt (siehe Abb.6 links). Ein Knoten, der zuvor auf den Knoten der eigenen Liste mit dem gemeinsamen Index verwiesen hat, verweist nun auf den Knoten der anderen Liste mit dem gemeinsamen Index (siehe Abb.6 Mitte). Nachdem der Knoten mit dem gemeinsamen Index einen Rückverweis auf diesen erhalten hat und die Ränder der Liste neu definiert worden sind, ist die Operation beendet.

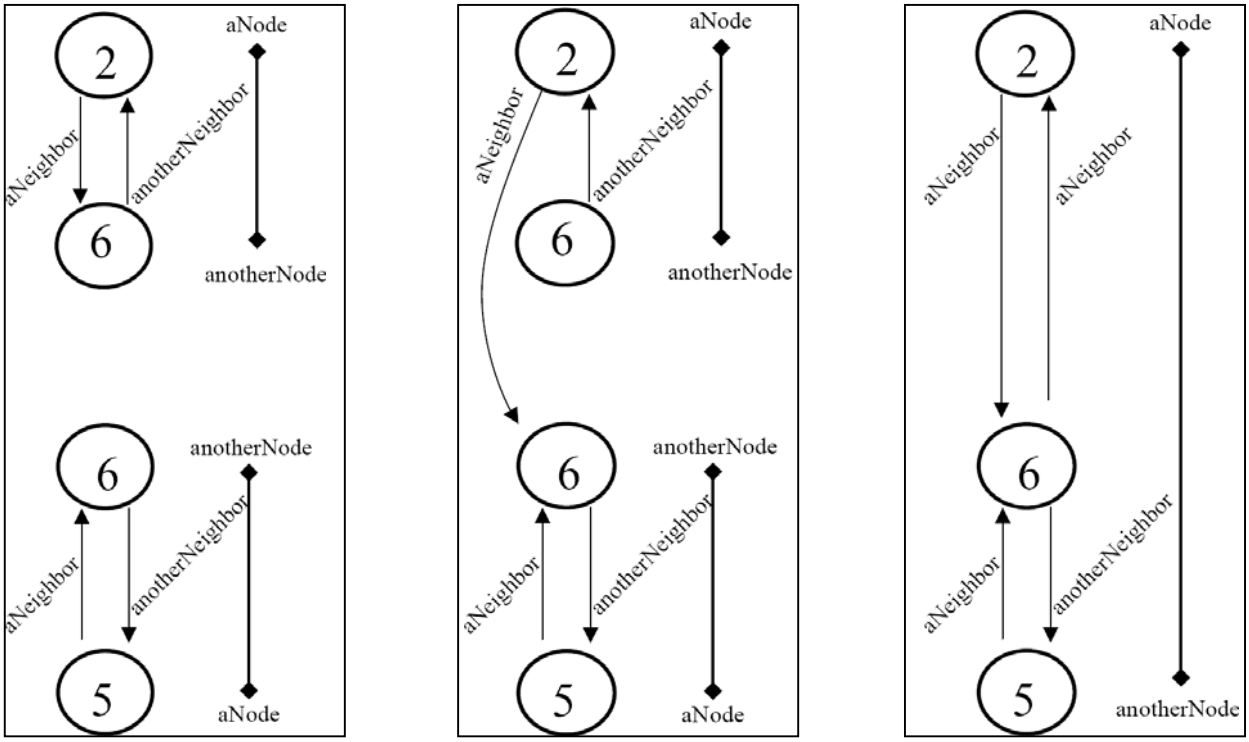

Abbildung 6: Beispiel einer Verbindungsoperation zweier Listen ungerichteter Knoten. Die Zahlen in den Knoten stehen für den enthaltenen Schnittpunktindex.

Bewertung

Alle Operationen, die in einer Box durchgeführt werden können, verursachen konstante Kosten. Da der Algorithmus alle Boxen genau einmal betrachtet, verhält sich die Laufzeit für einen Isolevel linear zur Anzahl der Boxen bzw. Messwerte. Isolinien zu verschiedenen Isoleveln können keine gemeinsamen Punkte haben, weshalb die Berechnungen zu verschiedenen Isoleveln nicht voneinander profitieren können. Die Laufzeit in Abhängigkeit der Boxenzahl n und Isolevelzahl m ist demnach: $O(n \cdot m)$. Da jeder Isolevel jeder Box mindestens einmal betrachtet werden muss, ist die Laufzeit des Algorithmus nicht mehr verbesserbar. Somit ist das asymptotische Verhalten mit dem des einfachen Marching Square Algorithmus identisch und auch in der Praxis verringert sich die Animationsgeschwindigkeit kaum. Insgesamt kann festgehalten werden, dass der erweiterte Marching Square Algorithmus (Marching Square SE) aufgrund seiner linearen Laufzeit lediglich vernachlässigbare Auswirkungen auf das Laufzeitverhalten hat und folglich zur Anwendung in dieser Echtzeitumgebung bestens geeignet ist. Durch die so ermöglichte Glättung und Beschriftung der Isolinien konnte die Qualität des Programms entscheidend verbessert werden. Außerdem liefert das Verfahren die Grundlage zum Einfärben von Isoflächen und lässt sich direkt auf andere Contourplot Verfahren übertragen.

2.4.3 Ergebnis

Abbildung 8 zeigt die durch den Marching Square SE Algorithmus erzeugten geglätteten und beschrifteten geschlossenen Linienzüge.

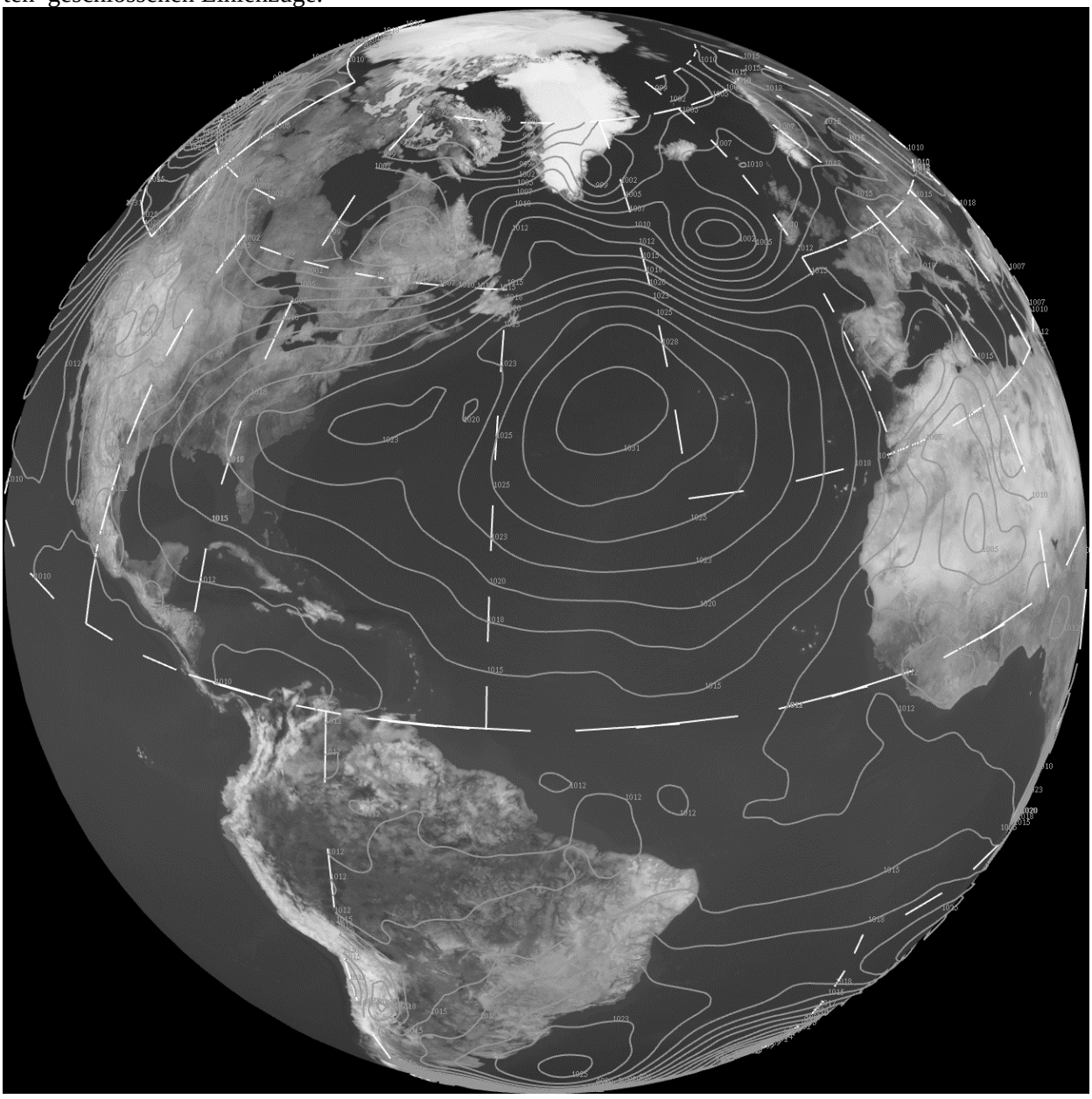

Abbildung 8: Die Isoliniendarstellung des Luftdrucks

2.4.4 Temperatur

Wie der Luftdruck ist auch die Temperatur ein skalarer Wert. Zur Darstellung von unterschiedlichen Temperaturen werden in der Regel Farben verwendet, da der Mensch von je her bestimmten Temperaturen eine Farbe zuordnet. So ist eine hohe Temperatur meist rötlich, die kalten Temperaturen werden meist mit Blau assoziiert. Dies sollte auch bei der Darstellung der Temperaturwerte berücksichtigt werden. Isolinien würden sich prinzipiell dazu eigenen eine Temperatur darzustellen. Allerdings wären unterschiedlich eingefärbte Isolinien nur schwer erkennbar. Daher wird die gesamte Erdoberfläche eingefärbt. Eine Möglichkeit der Oberflächeneinfärbung ist eine Abbildung aller Messwerte auf ein Farbspektrum. Anschließend werden die Flächen zwischen den Punkten mit Farbverläufen eingefärbt. Vorteil des

Verfahrens sind eine genaue Wiedergabe der Messwerte und durch die Ausführung auf der Grafikkarte hervorragende Performance. Bei gröberen Auflösungen können sich benachbarte Messwerte stärker unterscheiden. Die Folge sind wenig klare Farbverläufe, welche für den Betrachter nur schwer zu interpretieren sind (siehe Abb.9 links). Dieses Problem kann umgangen werden, indem bestimmte Temperaturintervalle auf jeweils eine Farbe abgebildet werden. Die so erzeugten Isoflächen wirken auf den Betrachter wesentlich übersichtlicher (siehe Abb.9 rechts). Dies ist allerdings mit einem Genauigkeitsverlust verbunden, da im Inneren einer Fläche nicht mehr der genaue Wert festgestellt werden kann.

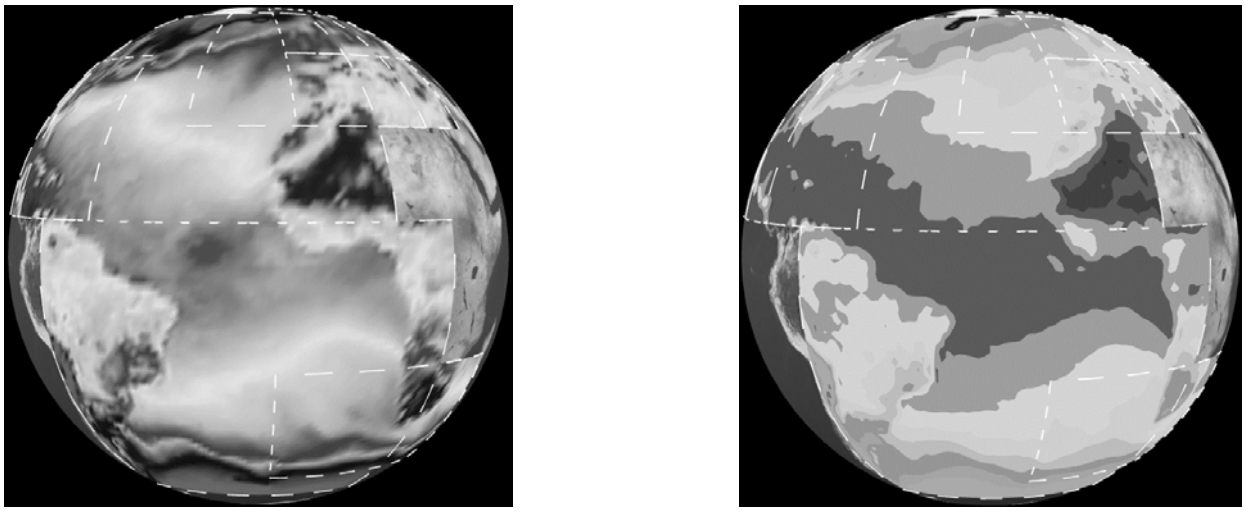

Abbildung 9: Vergleich der Temperaturdarstellung durch Farbverläufe (links) und durch Isoflächen (rechts).

Derzeit ist eine Glättung der Isoflächen nicht möglich und auch die Performance ist noch nicht optimal, weshalb die Isoflächendarstellungstechnik in der aktuellen Version noch nicht zum Einsatz kommt.

2.5 Animation

Die Dynamik des Klimas kann nicht durch statische Bilder, sondern nur durch Animation auf äußerst eindrucksvolle Weise vermittelt werden. Da keine vorberechneten Wetterkarten sondern die Ausgangsdaten selbst zum Client übertragen werden, kann eine Animation zwischen zwei Zeitpunkten durch lineare zeitliche Interpolation der Werte erzeugt werden. Für jeden so erzeugten Zwischenschritt wird die Darstellung unabhängig berechnet. Parallel dazu werden bereits im Hintergrund die für den nächsten Zeitpunkt benötigten Daten geladen, um eine gleichmäßige Animationsgeschwindigkeit zu ermöglichen. Die Entwicklung effizienter Algorithmen zum Laden und Visualisieren der Daten ist folglich von zentraler Bedeutung für ein Gelingen des Projekts, da sonst eine ausreichend flüssige Darstellung des zeitlichen Verlaufs unmöglich wäre.

3 Eigenschaften der verwendeten Visualisierungstechnik

In diesem Kapitel werden die wesentlichen Eigenschaften des Projekts erläutert und denkbare Nutzungsmöglichkeiten aufgezeigt.

Räumlichkeit
Die Möglichkeit der 3D-Darstellung des Klimas auf der gesamten Erde ist keine technische Spielerei sondern absolut unverzichtbar. Schließlich macht das Klima nicht vor Länder- und Kontinentgrenzen halt sondern kann nur global verstanden werden. Außerdem erleichtert gerade die räumliche Darstellung dem Betrachter die Orientierung und damit auch die Navigation.

Interaktivität
Bei vorberechneten Datenvisualisierungen (etwa Fernseh- oder Zeitungswetterberichte) müssen stets gestalterische Entscheidungen getroffen werden, welche visuelle Umsetzung geeignet ist und welche Aspekte besonders hervorgehoben werden sollen. Ein generelles Problem ist folglich die damit unweigerlich verbundene Interpretation der Ausgangsdaten. Da in diesem Projekt dagegen die Daten zum Client übermittelt werden und die Darstellung dort dynamisch berechnet wird, kann der Benutzer in den Visualisierungsprozess eingreifen und die für ihn interessanten Eigenschaften der Daten besonders herausheben. Auf diese Weise erreicht das Programm einen bislang unübertroffenen Interaktionsgrad.

Vielseitige Verwendbarkeit
Der hier beschriebene Ansatz ist nicht ausschließlich auf die Visualisierung von Klima- bzw. Wetterdaten festgelegt, sondern kann zur Darstellung beliebiger georeferenzierter Daten dienen. Dazu zählen u.a. Arbeitslosigkeit, Demografie, CO2-Ausstoß, Bevölkerungsdichte, Straßenkarten oder Wahlergebnisse.

Allgemeine Verfügbarkeit
In diesem Projekt werden ausschließlich freie und plattformunabhängige Werkzeuge verwendet. Demnach genügt zur Verwendung des Programms ein einfacher internetfähiger Rechner. Eine 3D-Grafikkarte verbessert zwar die Animationsgeschwindigkeit und Bildqualität, wird aber nicht vorausgesetzt. Außerdem muss nichts gesondert installiert werden, da alles im Applet läuft.

4 Fazit

In der heutigen Zeit gewinnt die Präsentation von Produkten, Ideen aber auch von wissenschaftlichen Ergebnissen zusehends an Bedeutung. Insbesondere gilt dies für Forschungszweige, in denen es große Datenmengen zu überblicken gilt. Hier können durch grafisches Darstellen der Messwerte selbst kleine Besonderheiten oder auch Fehler unmittelbar erkannt werden. „Ein Bild sagt mehr als 1000 Worte“ heißt es nicht umsonst im Volksmund. Außerdem kann auf diese Weise auch ein Laie in einem Maße Einblick in die aktuelle Forschung nehmen, der ihm sonst aufgrund der Komplexität der Materie verwehrt bliebe. Das entstehende Programm kann durch seinen besonders anschaulichen und interaktiven Ansatz einen kleinen Teil dazu beitragen, dass Forschungsergebnisse wie etwa die Entwicklung des Klimas von größeren Teilen der Menschheit verstanden werden. Um eine interaktive Anwendung zu entwickeln, in der sich die zu visualisierenden Daten ständig ändern, sind effiziente Algorithmen notwendig. Der in diesem Paper vorgestellte Marching Square SE ist ein wesentlicher Beitrag für die effiziente Berechnung von Isolinien und möglicherweise auch für Isoflächen. Trotz der eingeführten Veränderungen ist die Laufzeit des Algorithmus nur durch einen konstanten Faktor minimal vermindert, wodurch er sehr gut für die Echtzeitberechnung derartiger Linien geeignet ist. Nicht zuletzt angesichts der unaufhaltsamen Verbreitung des Internets steigt täglich die Zahl derer, die sich mit diesem Programm über aktuelle Erkenntnisse informieren können. Somit stellt diese Arbeit keine technische Spielerei dar, sondern ist auf einen wirklich sinnvollen Einsatz neuer Technologien ausgerichtet.

5 Literaturverzeichnis

Google Inc., Google Earth, 2007, http://earth.google.com

JOGL, The JOGL API Project, 2007, https://jogl.dev.java.net

Lorensen et al., 1987, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics (Proceedings of SIGGRAPH 87), Vol. 21, Nr. 4, S. 163-169.

OpenGL, The OpenGL API, 2007, http://www.opengl.org/