Programmierbare, interaktive 3D-Karten zur Kommunikation raumbezogener Information

Jürgen Döllner, Oliver Kersting, Klaus Hinrichs[1]

Abstract

Raumbezogene Objekte und Prozesse beschreiben den Zustand und die Veränderung in der Umwelt und spielen damit eine zentrale Rolle in der Umweltinformatik. Zur Durchführung von Maßnahmen des Umweltschutzes und der Umweltgestaltung ist es notwendig, diese Objekte und Prozesse zu visualisieren und zu steuern. Dieser Beitrag stellt ein Konzept für programmierbare, interaktive 3D-Karten vor, die die Präsentation, Exploration und Manipulation räumlicher und raumzeitlicher Objekte und Prozesse ermöglichen. Eine 3D-Karte basiert auf einem Geländemodell, in dessen Kontext raumbezogene Informationen durch Kartentexturen und geometrische Kartenobjekte visualisiert sind. Die Interaktivität der Kartenobjekte wird mit Hilfe einer integrierten Skriptsprache spezifiziert, die außerdem die Konfiguration bestehender und die Konstruktion neuer Kartenobjekte einer 3D-Karte durch den Benutzer ermöglicht. Derart programmierbare, interaktive 3D-Karten können zum Bau von Werkzeugen zur effizienten Kommunikation von Umweltinformation eingesetzt werden.

1. Einführung

Raumbezogene Objekte und Prozesse beschreiben den Zustand und die Veränderung in der Umwelt. Ihre Modellierung und Visualisierung spielt eine zentrale Rolle in der Umweltinformatik, denn ohne ein Verständnis dieser Objekte und Prozesse lassen sich Maßnahmen des Umweltschutzes oder der Umweltgestaltung kaum durchführen.

Um Einsicht in raumbezogene Daten zu gewinnen, müssen Werkzeuge zu ihrer Präsentation und Exploration bereitgestellt werden (Kraak 1994). Karten sind derartige Werkzeuge, denn sie ermöglichen die effektive Kommunikation raumbezogener Daten und Prozesse mit Hilfe kartographischer Abstraktion. Die Integration von Karten in virtuelle Umgebungen ermöglicht neue Formen von Karten und Kartennutzung. Karten in virtuellen Umgebungen zeichnen sich durch die Benutzer-

[1] Institut für Informatik, Universität Münster, Einsteinstr. 62, 48149 Münster.
Email: {dollner, kerstio, khh}@uni-muenster.de

Karten-Interaktion, die Benutzer-Umgebung-Immersion, die Variabilität der Informationsstärke und die Intelligenz der Kartenobjekte aus (MacEachren et al. 1999b). Im Vergleich zu Papierkarten stellen diese Karten also weiterführende Mittel zur Kommunikation raumbezogener Daten bereit. Deshalb müssen insbesondere die Methoden der kartographischen Abstraktion erweitert werden, um den Entwurf interaktiver Karten mit dynamischem Karteninhalt zu ermöglichen.

Dieser Beitrag beschreibt ein Konzept für programmierbare, interaktive 3D-Karten, die als visuelle Schnittstellen zu raumbezogenen Objekten und Prozessen verstanden werden. Die Software-Architektur definiert eine Sammlung von Bausteinen zur Konstruktion von 3D-Karten, die sich in drei Hauptkategorien einteilen lassen: *Visuelle Kartenobjekte* repräsentieren geometrische Objekte und ihre Erscheinung, *strukturelle Kartenobjekte* ordnen Kartenobjekte hierarchisch und *verhaltengebende Kartenobjekte* definieren die Interaktivität und Dynamik einer 3D-Karte. Eine konkrete 3D-Karte wird durch eine Menge von Kartenobjekten, die miteinander in Beziehung stehen, beschrieben.

3D-Karten nutzen zur Implementierung eine Skriptsprache, die es dem Benutzer erlaubt, Kartenobjekte zur Laufzeit zu konstruieren und zu manipulieren. Mit Hilfe von Skripten lassen sich zum Beispiel Reaktionen auf Zustandsänderungen und Benutzerinteraktionen spezifizieren. Die Skriptsprache in Verbindung mit einem leistungsfähigen Kartentexturenkonzept ermöglicht die Implementierung von 3D-Karten mit dynamischem Karteninhalt. Die Echtzeit-Darstellung wird durch Multiresolutionsmodelle für die Kartengeometrie und für die Kartentexturen gewährleistet.

Programmierbare, interaktive 3D-Karten als visuelle Schnittstellen zu raumbezogenen Objekten und Prozessen finden als visuelle Werkzeuge insbesondere bei der Planung, der Analyse, der Simulation und der Überwachung von Umweltdaten Einsatz und lassen sich als Software-Komponenten in Umweltanwendungen integrieren.

2. Verwandte Arbeiten

Kartographische Abstraktion als effektives Werkzeug zur Exploration und Manipulation räumlicher und raumzeitlicher Information wird seit vielen Jahren im Bereich der virtuellen Geo-Umgebungen eingesetzt (MacEachren et al. 1999b). Virtuelle Geo-Umgebungen, die versuchen, das menschliche Wahrnehmungssystem und kognitive Fähigkeiten auszunutzen (MacEachren et al. 1999a), beruhen auf digitalen Karten als grundlegende computergraphische Primitive. Die Hauptanforderungen, die an solche Karten gestellt werden, sind die Echtzeit-Darstellung durch auflösungsabhängige Geländemodelle, die Gewährleistung von Interaktivität zwischen dem Benutzer und der Karte und eine ausreichend hohe visuelle Qualität der Kartendarstellung (Terribilini 1999). In unserem Ansatz wird dem durch den Einsatz

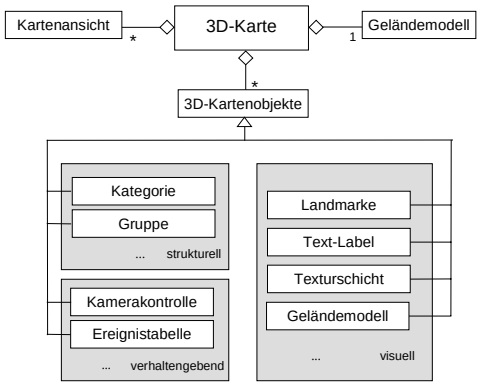

Abbildung 1 Konzeptionelles Modell der 3D-Kartenkomponenten.

eines hybriden Multiresolutionsmodells (Baumann et al. 1999) und durch texturbasierte Gestaltung des Karteninhalts (Döllner et al. 2000) Rechnung getragen.

Der Entwurf von 3D-Karten wurde von Haeberling (1999) untersucht, der drei wesentliche Aspekte bei der Gestaltung identifiziert: Bestimmung der Benutzergruppe, Spezifikation von Benutzeranforderungen und Ableitung von Anforderungen an die Kartensymbolik. Diese Aspekte lassen sich für 3D-Karten durch dynamischen Karteninhalt realisieren. Ervin (1993) stellt eine Kartengestaltung vor, die insbesondere die schematische, abstrakte Darstellung von raumbezogenen Daten betont. Dieser Ansatz lässt sich direkt auf der Grundlage der vorgestellten Sammlung von Kartenobjekten umsetzen.

Computergraphische Systeme, wie z.B. Java3D und OpenGL, stellen allgemeine computergraphische Funktionalität bereit, unterstützen jedoch nicht direkt die Visualisierung raumbezogener Daten. Die Bedürfnisse einer geospezifischen Visualisierung umfassen u.a. auflösungsabhängige Geländemodelle und georeferenzierte Texturen. Der hier vorgestellte Ansatz definiert geospezifische Software-Komponenten, die auf der Grundlage des Renderingsystems OpenGL implementiert wurden.

3. Konzeptionelle Elemente von 3D-Karten

Interaktive 3D-Schnittstellen für GIS sind unmittelbar mit heutigen Visualisierungs- und Graphiksystemen nur schwer zu realisieren, denn diese Systeme stellen meist Primitive bereit, die weder eine passende Abstraktion noch effiziente Algorithmen für die speziellen Anforderungen der Geo-Visualisierung besitzen. Unsere Motivati-

Abbildung 2 Landmarken als Unterstützung zur Wegfindung (links). Label mit interaktiver Landmarke (Mitte). Geländemodell mit Schattierungstextur und Thematiktexturen (rechts).

on war es, eine Sammlung von 3D-Kartenkomponenten zu identifizieren, die es Entwicklern erlaubt, effiziente visuelle Schnittstellen für raumzeitliche Daten zu konstruieren.

Wir stellen ein konzeptionelles Modell für interaktive 3D-Karten vor, das Komponenten für das Geländemodell, den Karteninhalt, die Kartendynamik und die Karteninteraktivität bereitstellt. In diesem Modell wird eine konkrete 3D-Karte durch ein Geländemodell, eine oder mehrere Kartenansichten und eine Sammlung von Kartenobjekten zur Spezifikation des Karteninhalts repräsentiert; Abbildung 1 zeigt das konzeptionelle Modell der 3D-Kartenkomponenten. Wir unterscheiden zwischen visuellen, strukturellen und verhaltengebenden Kartenobjekten.

3.1 Geländemodell

Ein fundamentales Primitiv für 3D-Karten ist das Geländemodell. Um interaktive Bildwiederholraten zu sichern, kommen i.d.R. Multiresolutionsmodelle für die Geländegeometrie zum Einsatz, z.B. Multitriangulation (De Floriani et al. 1998), Progressive Meshes (Hoppe 1998) oder Restricted Quad-Tree Triangulation (Pajarola 1998). Diese Verfahren sind meist auf eine reine Repräsentation der Geometrie beschränkt; eine Repräsentation der Geländetexturdaten wird oftmals nicht unterstützt. Texturen sind aber unverzichtbar, um thematische Daten in 3D-Karten in hoher Qualität und dazu hardwarebeschleunigt darzustellen.

In unserem Ansatz verwenden wir den *Approximation-Tree* (Baumann et al. 1999), ein Multiresolutionsgeländemodell, das Level-of-Detail-Mechanismen sowohl für Geometrie- als auch für Texturdaten bereitstellt. Der Approximationsbaum erlaubt die exakte Repräsentation der Geländemorphologie durch die Kombination von regulären Gitterdaten mit TIN-Daten, die Regionen von besonderem Interesse verfeinert darstellen. Der komplementäre *Texture-Tree* (Baumann et al. 2000) imp-

lementiert ein Multiresolutionsmodell für 2D-Geländetexturen. Mehrere Texturbäume lassen sich mit einem Approximationsbaum assoziieren. Auf diese Weise erhalten 3D-Karten mehrfache, mitunter überlappende Texturen. Abbildung 2 (rechts) zeigt eine 3D-Karte mit mehrfachen Texturen, konkret einer Schattierungstextur, einer Überflutungstextur und einer thematischen Textur.

3.2 Visuelle Kartenobjekte

Visuelle 3D-Kartenobjekte repräsentieren sichtbare, georeferenzierte 3D- und 2D-Objekte, die über graphische Attribute, wie z.B. Farbe, Größe, Form und Orientierung, verfügen. Ein spezielles visuelles 3D-Kartenobjekt ist das Geländemodell. Verglichen mit Primitiven von 3D-Graphiksystemen sind die visuellen 3D-Kartenobjekte auf die Bedürfnisse der 3D-Karten abgestimmt und weisen eine komplexere Struktur sowie ein komplexeres Verhalten auf. Visuelle Kartenobjekte umfassen:

- *Landmarken* sind georeferenzierte 2D- oder 3D-Symbole, die z.B. zur Markierung diskreter Geo-Objekte und –Positionen, zum Aufzeigen von Wegen und Richtungen (Elvins et al. 1997) sowie zur Repräsentation thematischer Informationen eingesetzt werden (Abbildung 2 links).
- *Labels* sind textuelle Beschreibungen, die mit Geo-Positionen in der 3D-Karte assoziiert werden (Abbildung 2 Mitte). Labels werden im Allgemeinen in der Sichtebene positioniert und automatisch zum Betrachter ausgerichtet (Billboarding-Technik).
- *Texturschichten* repräsentieren 2D-Bilddaten, die auf das Geländemodell projiziert werden. Texturschichten können das gesamte Gelände oder nur Teile davon überdecken; ihre Lage wird über Geo-Koordinaten festgelegt. Texturschichten werden miteinander in einer 3D-Kartenansicht visuell kombiniert (Abbildung 2 rechts).

3D-Karten und Kartenobjekte können interaktiv konfiguriert werden und ermöglichen es damit, benutzerspezifische und aufgabenspezifische Informationen zu integrieren (siehe Abschnitt 5).

3.3 Strukturelle Kartenobjekte

Strukturelle 3D-Kartenobjekte werden eingesetzt, um 3D-Kartenobjekte konzeptionell zu ordnen und um Beziehungen zwischen ihnen auszudrücken; sie erlauben dem Benutzer große, individuelle Sammlungen von Kartenobjekten zu klassifizieren und zu organisieren. Strukturelle 3D-Kartenobjekte umfassen:

- *Kategorien* erlauben die schematische Klassifikation für Kartenobjekte, d.h. sie assoziieren Metainformation zu Kartenobjekten. Zum Beispiel kann eine thematische Textur, die Daten über den Waldbestand repräsentiert, mit der Kategorie „Wald“ und der Kategorie „Bestandsaufnahme-1950“ in Beziehung gesetzt werden.
- *Gruppen* ordnen Kartenobjekte in anwendungsspezifische Einheiten an. Gruppen selbst bilden eine eigenständige Hierarchie; sie lassen sich ineinander schachteln. Ein Kartenobjekt kann einer oder mehreren Gruppen angehören. So kann zum Beispiel eine Sammlung von „Haus“-Landmarken in einer „Stadt“-Gruppe zusammengefasst werden.

3.4 Verhaltengebende Kartenobjekte

Verhaltengebende 3D-Kartenobjekte können auf Ereignisse und Veränderungen durch den Aufruf von (anwendungsspezifischen) Aktionen reagieren und definieren dadurch die Interaktivität und Dynamik von 3D-Karten. Um Kartenobjekten ein von ihrer Klasse unabhängiges Verhalten zuordnen zu können, werden verhaltengebende 3D-Kartenobjekte als separate Objekte modelliert. Aus diesem Grund wird das anwendungsspezifische Verhalten nicht durch die Kartenobjekt-Klassen festgelegt, sondern es wird durch die Assoziation von Kartenobjekten und verhaltengebenden Kartenobjekten definiert.

Ereignistabellen sind generische verhaltengebende Kartenobjekte, die Ereignisse mit Aktionen assoziieren. Ereignisse umfassen Geräteereignisse (z.B. Mausklick oder –bewegung), Zeitereignisse und Statusänderungen von Kartenobjekten (z.B. Sichtbarkeitsänderung von Objekten).

Aktionen werden ausgeführt, falls das mit ihnen assoziierte Ereignis eintritt. Eine Aktion ist als Skript implementiert, das durch die integrierte Skriptsprache Tcl interpretiert wird. Skriptbasierte Aktionen ermöglichen es dem Entwickler und den Kartenbenutzern, 3D-Karten zur Laufzeit zu programmieren (siehe Abschnitt 5).

Unterschiedliche Kartenobjekte können ein gemeinsames Ereignistabellen-Kartenobjekt besitzen. Dieses Konzept erlaubt es, Kartenobjekte unterschiedlicher Klassen mit einheitlichem Verhalten auszustatten.

Constraint-Kartenobjekte sind eine weiter Kategorie von verhaltengebenden Kartenobjekten, die Parameter von Kartenobjekten einschränken bzw. überwachen. Das *Kameraanimations-Kartenobjekt* zum Beispiel bewegt und orientiert die Kamera entlang eines definierten Pfades durch die 3D-Karte, indem es auf Zeitereignisse reagiert. Das *Kamerakontroll-Kartenobjekt* stellt sicher, dass benutzerdefinierte Beschränkungen und Forderungen an die Kameraparameter fortlaufend überprüft werden. Zum Beispiel können dadurch die maximale Kamerahöhe oder die Freiheitsgrade während der interaktiven Navigation beschränkt werden.

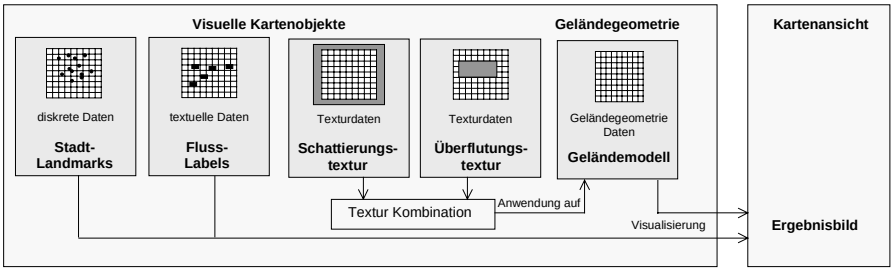

Abbildung 3 Beispiel einer konkreten 3D-Karte und ihrer Kartenobjekte.

3.5 Kartenansichten

Kartenansichten visualisieren 3D-Karten in einem Graphikfenster auf der Basis des Renderingsystems OpenGL (Woo et al. 1999). Für eine gegebene 3D-Karte kann der Benutzer eine beliebige Zahl an Kartenansichten erzeugen und unabhängige Kameraeinstellungen spezifizieren.

Mehrere Kartenansichten können die gleiche 3D-Karte visualisieren, zum Beispiel in einer Hauptansicht mit einer perspektivischen Projektion und eine Überblicksansicht mit einer orthogonalen Projektion, die die Navigation und Orientierung in der Hauptansicht unterstützt. Abbildung 3 zeigt den Aufbau einer konkreten 3D-Karte und ihrer Kartenobjekte.

4. Dynamik von und Interaktion mit 3D-Karten

Animierte und interaktive Visualisierung ist ein Mittel für die bessere und intuitive Wahrnehmung von raumzeitlichen Daten. Kartographische Animation untersucht insbesondere das Potential der dynamischen Kartenrepräsentation zur Kommunikation von Geo-Prozessen und -Phänomenen. Um animierte und interaktive Visualisierungsstrategien zu unterstützen, muss es zur Laufzeit möglich sein, 3D-Karten zu konfigurieren und 3D-Kartenobjekte zu modifizieren.

4.1 Level-of-Detail-Gruppen

Thematische Informationen einer 3D-Karte sollten angepasst an Benutzerbedürfnisse und Bildauflösung mit einer angemessenen Informationsdichte visualisiert wer-

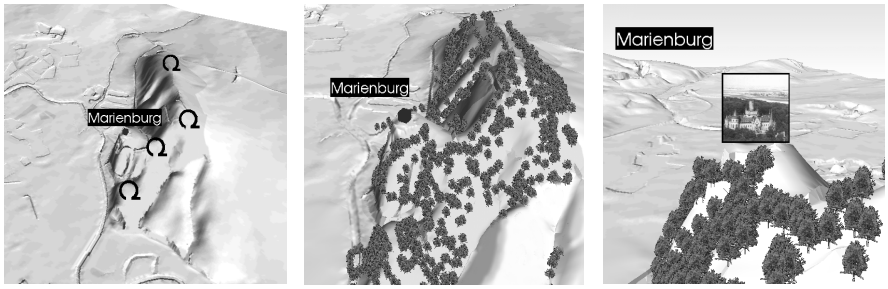

Abbildung 4 LOD-Darstellung eines Geo-Objektes und eines Geo-Feldes.

den. Andernfalls kann die kognitive Belastung für den Benutzer einer 3D-Karte zu hoch werden (MacEachren et al. 1999).

In einer 3D-Karte repräsentiert eine *Level-of-Detail-Gruppe (LOD)* thematische Informationen in unterschiedlicher Intensität und ist damit ein Mittel zur Umsetzung kartographischer Generalisierung. Die LOD-Gruppe enthält eine Sammlung von distanzabhängigen, möglicherweise inhomogenen Kartenobjekten, die thematische Informationen in unterschiedlichen Auflösungsstufen beschreiben. Für jedes Kartenobjekt in der LOD-Gruppe ist ein Sichtbarkeitsintervall angegeben, das festlegt, bei welchem Abstand zur Kamera das Objekt dargestellt werden soll. Die Sichtbarkeitsintervalle von Kartenobjekten können sich überlappen, um Verblendungen zwischen Auflösungsstufen zu ermöglichen.

Abbildung 4 zeigt eine auflösungsabhängige Darstellung von Geo-Objekten und Geo-Feldern. Das Geo-Objekt, eine Burg, wird durch ein Label, durch eine Box und durch ein Photo repräsentiert. Das Geo-Feld, ein Landnutzungsdatensatz für die Thematik Wald, wird durch Glyphen und eine thematische Textur visualisiert.

4.2 Texturfolgen

Texturfolgen können zur Kodierung von zeitvarianten thematischen Daten verwendet werden. Jede Textur einer Texturfolge spezifiziert ein Schlüsselbild einer Animation. Während der Animation werden zwei aufeinanderfolgende Schlüsselbilder mit passenden Gewichtungen interpoliert. Es muss kein Zwischenbild erzeugt werden (z.B. durch rechenintensive Bildoperationen), weil die Texturen im Bildraum während des Renderings kombiniert werden. Damit lassen sich sogar hochaufgelöste Texturen in Echtzeit animieren.

Das Texturfolgen-Kartenobjekt ist eine spezielle Texturschicht, die eine Folge von Texturen speichert und die Gewichtungen der Texturen zu einem gegebenen Zeitpunkt errechnet. Abbildung 5 zeigt die Überflutung in einer Landschaft; die

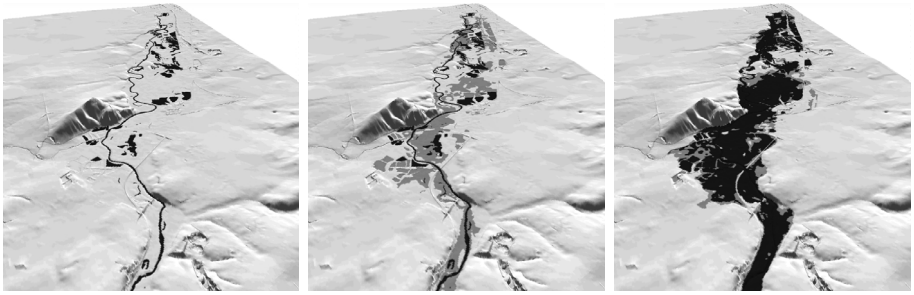

Abbildung 5 Visualisierung einer Überflutungsprozesses mittels Texturfolge.

Wasserausbreitung wurde (zu diskreten Zeitpunkten) als Textur berechnet und in einer Texturfolge abgelegt.

4.3 Texturlinsen

Texturlinsen sind spezielle Texturschichten, die die Sichtbarkeit und Kombination von thematischen Texturschichten in einer 3D-Karte modifizieren. Auf diese Weise können Informationen in 3D-Kartenansichten dynamisch fokussiert und beschränkt werden.

Eine *Luminanz-Texturlinse* modifiziert die Helligkeit einer anderen thematischen Texturschicht. Durch die Animation einer Luminanz-Texturlinse können Anwender durch 3D-Karten geleitet werden (z.B. Wegbeschreibung).

Eine *Filter-Texturlinse* beschränkt die Sichtbarkeit anderer thematischer Texturschichten. Sie kann zur Modifikation der visuellen Repräsentation verwendet werden, z.B. indem thematische Daten aus einer bestimmten Texturschicht innerhalb der Linse aus- bzw. eingeblendet werden.

Texturlinsen erlauben es, Informationen in der Kartenansicht nur dort zu visualisieren, wo sie vom Benutzer benötigt werden, d.h. das Bild der 3D-Karte enthält nur die graphischen Elemente, die im momentanen Kontext von Bedeutung sind. Dadurch lässt sich die Komplexität des Karteninhalts reduzieren.

5. Programmierung von 3D-Karten

Um in Anwendungen 3D-Karten flexibel einsetzen zu können, müssen sie zur Laufzeit konfiguriert und individualisiert werden können. Dazu stellen 3D-Karten Mechanismen bereit, Kartenobjekte zur Laufzeit zu erzeugen, zu manipulieren, zu zerstören und mit anderen Kartenobjekten zu assoziieren.

Abbildung 6 Interaktive Analyse der Bodenbelastung mit Hilfe einer Filter-Texturlinse (links). Aufhellung von Bereichen besonderen Interesses mittels Luminanz-Linse (rechts).

5.1 Verwaltung von 3D-Kartenobjekten

Eine Möglichkeit bestünde darin, 3D-Karten auf der Ebene der Systemprogrammiersprache C++ zu programmieren. Jedoch setzt dies exakte Kenntnisse der Systemprogrammiersprache und des objektorientierten Systementwurfs voraus. Technisch sind darüber hinaus Kompilation und Bindung von Anwendungen erforderlich. Daher ist aus der Sicht eines Kartenentwicklers die Systemprogrammiersprache keine gute Wahl zur Konfiguration und Individualisierung von 3D-Karten.

In unserem Ansatz benutzen wir die integrierte Skriptsprache Tcl (Ousterhout 1994), um 3D-Karten zu konfigurieren und zu individualisieren. Die C++-Implementierung der Kartenklassen stellt für Tcl geeignete Schnittstellen bereit, sodass Kartenobjekte mit Hilfe der Skriptsprache erzeugt und manipuliert werden können. Da Tcl eine interpretierte Sprache ist und über eine leicht lernbare Syntax verfügt, ist es im Vergleich zu Systemprogrammiersprachen einfacher und schneller, Anwendungen zu erstellen (Ousterhout 1998). Eine große Anzahl von Tcl-Erweiterungen, wie z.B. Constraint-Solver und Datenbankschnittstellen (Harrison 1997), sind verfügbar und können die Funktionalität von 3D-Karten ausbauen.

In Tabelle 1 sind die Befehle aufgelistet, um ein Label-Kartenobjekt zu erzeugen, zu zerstören und zu individualisieren. Tabelle 2 zeigt eine Liste aller Label-Eigenschaften, die über die Skriptsprache zur Laufzeit konfiguriert werden können.

5.2 Spezifikation von Interaktivität für 3D-Kartenobjekte

3D-Kartenobjekte können auf Statusänderungen (z.B. Sichtbarkeits-, Distanzveränderung) und Ereignisse (z.B. Benutzerselektion, Zeitereignisse) durch benutzerdefi-

Operation	Tcl-Befehl
Erzeugung	`create LABEL myLabel`
Zerstörung	`delete myLabel`
Modifikation	`configure myLabel –color {1 0 0}`
Anfrage	`info myLabel –color`

Tabelle 1 Tcl-Befehle für Label-Kartenobjekte.

nierte Aktionen reagieren. Dies erlaubt dem Benutzer, jedes Kartenobjekt mit einem individuellen Verhalten auszustatten. Selektiert z.B. der Benutzer ein Objekt mit der Maus, wird die Intersektions-Aktion aufgerufen. Die Aktionsparameter sind in diesem Fall der 3D-Intersektionspunkt, der Name und der Typ des selektierten Objekts. Beispiele verfügbarer Aktionen für 3D-Kartenobjekte sind in Tabelle 3 aufgeführt.

Aktionen werden durch Tcl-Skripte spezifiziert, wobei die folgenden speziellen Variablen bereitgestellt werden: `%t` beinhaltet den Kartenobjekttyp (z.B. `LABEL`), `%n` den Objektnamen (z.B. `myLabel`) und `%v` den Ergebniswert aus dem Wertebereich (z.B. `true`).

Beispiel 1: Das folgende Programmfragment spezifiziert zwei Aktionen für ein Label-Kartenobjekt. Die erste Aktion (*visibleCall*) blendet automatisch ein Informationsfenster ein, wenn das Label in der aktuellen Kartenansicht sichtbar ist. Die zweite Aktion (*distanceCall*) legt die Beschreibung, die im Informationsfenster angezeigt werden soll, in Abhängigkeit vom Abstand des Labels zur Kamera fest: Ist

Property	Domain	Meaning
`-billboard`	`true` or `false`	automatische Label Orientierung
`-categories`	`{c1 ...cN}`	assoziierte Kategorien
`-color`	`{r g b a}`	Labelfarbe
`-distanceCall`	`tclScriptD`	Distanz-Aktion
`-enabled`	`true` or `false`	Enabled Flag
`-intersectCall`	`tclScriptI`	Intersektions-Aktion
`-llf`	`{x0 y0 z0}`	lower-left-front der Boundingbox
`-offset`	`{xo yo zo}`	Labeloffset zur Sichtebene
`-position`	`{xi yi zi}`	Labelposition
`-selectCall`	`tclScriptS`	Selektions-Aktion
`-size`	`size_value`	Labelgröße
`-text`	`{labeltext}`	Labeltext
`-urb`	`{x1 y1 z1}`	upper-right-back der Boundingbox
`-viewcoordinates`	`true` or `false`	Koordinatensystem
`-visibleCall`	`tclScriptV`	Sichtbarkeits-Aktion

Tabelle 2 Eigenschaften des Label-Kartenobjekts.

Typ	Wertebereich	Auslöser
visibleCall	`true` or `false`	Veränderung von Kamera oder Boundingbox
distanceCall	`[0,INF]`	Veränderung von Kamera oder Boundingbox
intersectCall	{x y z}	Objektselektion durch Benutzer (z.B. Maus)
resultCall	objektabhängig	objektabhängig

Tabelle 3 Aktionen für 3D-Kartenobjekte.

das Label weit entfernt, wird eine kurze Beschreibung, andernfalls eine detaillierte Beschreibung eingeblendet. Das Beispiel zeigt, wie beliebige, benutzerspezifische Tcl-Prozeduren die Funktionalität von 3D-Karten erweitern können.

```
proc showInfoBox {visible} {
    if {$visible} {
        # show info window
        wm deiconify .infowin
    } else {
        # hide info window
        wm withdraw .infowin
    }
}

proc setInfoText {distance} {
    if {$distance > 1000} {
        # use short description
        .infowin.text configure -text $shortDescription
    } else {
        # use long description
        .infowin.text configure -text $longDescription
    }
}

# add callbacks to myLabel
configure myLabel -visibleCall {showInfoBox %v} \
    -distanceCall {setInfoText %v}
```

Beispiel 2: Im folgenden Codebeispiel wird eine interaktive Texturlinse spezifiziert. Selektiert der Benutzer das Geländemodell, wird der 3D-Intersektionspunkt errechnet und an die Intersektions-Aktion übergeben. Diese Aktion setzt das Zentrum der Linse auf den errechneten Schnittpunkt. Dadurch kann der Benutzer die Linse interaktiv über das Gelände bewegen. Zusätzlich wird ein Schieberegler in die graphische Benutzerschnittstelle integriert, um die Größe der Linse zu steuern.

```
# create lens map object
create LENS myLens \
    -mixer "lens.rgb" \
    -texture1 "complexthematic.rgb" \
    -texture2 "streets.rgb" -size 1000

# bind lens position to mouse position
```

```
configure myTerrain -intersectCall \
    {configure myLens -center %v}


# create slider to modify lens size
scale .mySlider -from 1 -to 1000 -resolution 5 \
    -command {configure myLens -size}

# make slider visible
pack .mySlider
```

Der hybride Ansatz bestehend aus einer Systemprogrammiersprache zur Implementierung von 3D-Kartenklassen und einer Skriptsprache zur Konfiguration und Individualisierung von 3D-Kartenobjekten bildet unserer Erfahrung nach eine leistungsfähige Kombination. Obwohl die meisten Befehle durch die Skriptsprache ausgelöst werden, werden die zeitkritischen Operationen letztlich durch C++-Code ausgeführt. Aus diesem Grund gibt es keine signifikanten Effizienzeinbußen.

6. Zusammenfassung

Die vorgestellten programmierbaren, interaktiven 3D-Karten zeichnen sich als Werkzeug für die Präsentation und Exploration raumbezogener Daten durch ihr großes Maß an Gestaltbarkeit, Konfigurierbarkeit und Individualisierbarkeit aus. 3D-Karten ermöglichen den unmittelbaren, visuellen Zugang zu raumbezogenen Daten, der u.a. bei Umweltplanungs- und Umweltsimulationsaufgaben notwendig ist. Schließlich ergeben sich aus den vielfältigen Darstellungsmöglichkeiten Chancen für eine zielgruppenorientierte und aufgabenzentrierte Visualisierung raumbezogener Daten. Die leistungsfähige Kartentexturverwaltung schafft die Voraussetzung zur Visualisierung zeitvarianter raumbezogener Daten und Prozesse (z.B. in Form von Texturanimation). Der Einsatz mehrfacher Kartentexturen, die in Echtzeit visuell kombiniert werden können, ermöglichen die Konstruktion leistungsfähiger Explorationswerkzeuge (z.B. Filterlinsen, Detaillinsen).

Der hohe Grad an Abstraktion im konzeptionellen Aufbau der 3D-Kartenobjekte vereinfacht deutlich die Erstellung eigener 3D-Karten-basierter Anwendungen im Vergleich zu allgemeinen Graphiksystemen und beinhaltet darüber hinaus geospezifische computergraphische Funktionalität. Als Software-Komponenten lassen sich 3D-Karten direkt in Anwendungsprogramme integrieren.

Literaturverzeichnis

K. Baumann, J. Döllner, K. Hinrichs, O. Kersting (1999): A Hybrid, Hierarchial Data Structure for Real-Time Terrain Visualization, Computer Graphics International CGI '99, S. 85-92

K. Baumann, J. Döllner, K. Hinrichs (2000): Integrated Multiresolution Geometry and Texture Models for Terrain Visualization, Proceedings of the Joint EUROGRAPHICS and IEEE TCGV Symposium on Visualization, S. 157-166

L. De Floriani, P. Magillo, E. Puppo (1998): Efficient Implementation of Multi-Triangulations. Proceedings IEEE Visualization '98, S. 43-50

T. Elvins, D. Nadeau, D. Kirsh (1997): Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments, Proceedings of UIST 97, S. 21-30

J. Döllner, K. Baumann, K. Hinrichs (2000): Texturing Techniques for Terrain Visualization. Proceedings IEEE Visualization 2000, im Druck

S. Ervin (1993): Landscape Visualization with Emaps, IEEE Computer Graphics & Applications, 13(2), S. 28-33

C. Haeberling (1999): Symbolization in Topograhic 3D Maps: Conceptual Aspects for User-Oriented Design, 19th International Cartographic Conference, S. 1037-1044

M. Harrison (1997): Tcl/Tk Tools, O'Reilly & Associates, Chapter 11, S. 453-473

H. Hoppe (1998): Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering, Proceedings IEEE Visualization '98, S. 35-42

M.J. Kraak (1994): Interactive Modelling Environment for Three-dimensional Maps: Functionality and Interface Issues, In: A. M. MacEachren, D. R. F. Taylor (Eds.), Visualization in Modern Cartography, Pergamon, S. 269-285

A.M. MacEachren R. Edsall, D. Haug, R. Baxter, G. Otto, R. Masters, S. Fuhrmann, L. Qian (1999a): Exploring the Potential of Virtual Environments for Geographic Visualization, Association of American Geographers

A.M. MacEachren, M.J. Kraak, E. Verbree (1999b): Cartographic Issues in the Design and Application of Geospatial Virtual Environments, 19th International Cartographic Conference, S. 657-665

J. Ousterhout (1994): Tcl/Tk, Addison-Wesley

J. Ousterhout (1998): Scripting: Higher Level Programming for the 21st Century. IEEE Computer, 31(3), S. 23-30

R. Pajarola (1998): Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation, Proceedings IEEE Visualization '98, S. 19-26

A. Terribilini (1999): Maps in transition: development of interactive vector-based topographic 3D-maps, 19th International Cartographic Conference, S. 993-1001

M. Woo, J. Neider, T. Davis, D. Shreiner (1999): OpenGL Programming Guide - 3rd Edition, Addison-Wesley