Erstellung eines dreidimensionalen Höhleninformationssystems

Vertiefungsblock in Kartographie
WS 1999/2000

Professor: L. Hurni
Betreuer: A. Neumann, M. Heller
Studenten: Michael Müller, Christoph Schmid

Inhalt

1 Einleitung 3
2 Anforderungen an ein Höhleninformationssystem 3
3 Projektbeschrieb 4
3.1 Datenstruktur 4
4 Programmiertechnisches 6
4.1 VRML 6
4.2 JAVA 6
4.3 EAI (External Authoring Interface) 7
4.4 JAVASCRIPT 7
5 Cave Information System Bedienungsanleitung 8
5.1 Das VRML-PlugIn 8
5.2 Die Übersichtskarte 9
5.3 Das Cave Information System Hauptmenü 9
6 Aufgetretene Probleme 13
7 Erweiterungsmöglichkeiten 19
8 Schlusswort 19
9 Internet-Adressen 20

1 Einleitung

Die Höhlenforschung ist ein grosses Forschungsfeld, das jedoch einer kleiner Forschergruppe vorbehalten ist, vor allem, weil es wenig wirtschaftlichen Ertrag verspricht. Aus diesem Grund existieren auch kaum professionelle Lösungen für speläologische Probleme. Eines dieser Probleme ist die Vermessung und die Kartierung einer Höhle. Die Vermessung einer Höhle ist eine anspruchsvolle Aufgabe, wie man auch bei Tunnelbauten beobachten kann. Die anschliessende Visualisierung der Vermessungsdaten wirft das Problem der Dreidimensionalität auf. Eine Höhle ist oft ein komplexes Durcheinander von Gängen, und kann demnach zweidimensional nur ungenügend oder ausschnittsweise wiedergegeben werden. Die Herausforderung ist es nun, eine dreidimensionale Visualisierung mit einem Informationssystem zu verknüpfen, und so die Höhle auch für den Laien verständlich darzustellen. Gerade die neuen Computertechniken bieten Möglichkeiten, auf diesem Gebiet neue Ansätze zu realisieren.

2 Anforderungen an ein Höhleninformationssystem

Der Benutzer eines Höhleninformationssystems wird in der Regel aus dem Gebiet der Höhlenforschung stammen, also sicher nicht zwangsläufig ein Computerexperte sein. Das Informationssystem sollte deshalb einfach zu bedienen sein. Der Nutzer wird erwarten, dass das System die gesamte Höhle als Vermessungswerk kennt, und demnach Geometrie und Topologie aufzeigen kann. Die Geometrie eines Höhlensystems fordert geradezu nach einer dreidimensionalen Darstellung, zweidimensionale Risse vermögen bloss einen Überblick zu verschaffen. Das System sollte die gängigen und bekannten Bezeichnungen und Merkmale aller Höhlenteile gespeichert haben und den Elementen zuweisen können. Diese Anforderungen stellen die Basis für ein Höhleninformationssystem dar.
Der Nutzer möchte nun möglichst einfachen und doch flexiblen Zugriff auf alle verfügbaren Daten haben. Das sollte einerseits über die grafische Darstellung möglich sein und andererseits als Abfragemenü in übersichtlicher Form realisiert sein. So können grundlegende Abfragen wie "wie heisst dieser Gang?" oder "zeige mir Gang X" durchgeführt werden. Der interessierte Nutzer möchte jedoch alle vorhandenen Informationen in übersichtlicher Form abfragen und sogar kombinieren können. Aus dem Attribut 'Ganglänge' wird dann beispielsweise eine Abfrage "Wie weit bin ich vom nächsten Ausgang entfernt?", welche bereits einen grossen praktischen Wert haben kann. Diese Art von Abfragen nutzt aus den Basisdaten abgeleitete Informationen.
Zusätzliche Anforderungen an das System ergeben sich aus dem Fachgebiet des Nutzers. Ein Geologe beispielsweise möchte geologische Attribute grafisch darstellen lassen, um daraus Rückschlüsse über die Entstehung der Höhle Erkenntnisse zu gewinnen. Der Höhlenforscher möchte vielleicht die Höhle weiter unterteilen und diese Gruppierung darstellen und speichern.
Die Liste mit möglichen Anforderungen könnte beliebig weitergeführt werden, man sieht bereits an diesen wenigen Beispielen, wie vielfältig die Anforderungen sein können. Man wird nie alle Ansprüche befriedigen können, das Höhleninformationssystem wird ein Kompromiss zwischen dem Nutzen einer Option und deren technischer Aufwand sein müssen.

3 Projektbeschrieb

Im Vordergrund des Projekts steht die 3D-Visualisierung der Höhle. Man soll die Höhle frei im dreidimensionalen Raum betrachten können. Dazu wird der gesamte Untergrund sozusagen invertiert, das Erdreich verschwindet, die Höhlenteile werden zu Materie. Die zusätzliche Darstellung der Topographie der Umgebung soll die Orientierung erleichtern. Die Visualisierung bildet, wie bereits erwähnt, den Kern des Informationssystems. Alle weiteren Programmmöglichkeiten stehen demnach im Hintergrund und werden nur auf Verlangen aktiviert.

3.1 Datenstruktur

Die Daten sind extern gespeichert und streng vom Programm getrennt. So ist das System flexibel, und man kann die Daten ersetzen. Es können aktualisierte Daten einfach übernommen werden oder sogar ein komplett anderer Datensatz eingesetzt werden. Die Daten werden weiter unterteilt in die Geometriedaten (i.a. Gänge oder Gangteile, in der Folge als Geometrieelemente bezeichnet), deren Attribute und Strukturierung.

Geometriedaten:

Im VRML 2.0 – Format, werden von TOPOROBOT[1] extrahiert. Für jedes Geometrieelement wird ein IndexedFaceSet verwendet. In dieser Datei sind auch alle vom Programm benötigten VRML-Protos und Knoten bereits enthalten. Diese Datei wird nur vom VRML-Plugin verwendet. Die Topographie ist unabhängig von der Höhle in einer zweiten VRML-Datei gespeichert.

Attribute:

In der Datei *statistic.txt* im Textformat gespeichert. Zu jedem Geometrieelement sind Informationen wie Name, Länge etc. gespeichert. Der Aufbau dieser Datei ist einfach erweiterbar.

Hier ein Auszug aus der statistic.txt Datei der Schachernhöhle:

```
Schachernhoehle
Guest
20.1.2000
V 1

     1      4.80     1    0     1    2     2  Eingang
     2      4.60     1    1     2    2     2  Mueckenschluf
```

Über diese Attribute können verschiedenste Abfragen getätigt und gleich mit der Geometrie verknüpft werden.

[1] TOPOROBOT ist ein Programm zur Erfassung, Verwaltung und Manipulation von Höhlenmessdaten inkl Exportfilter (http://www.geo.unizh.ch/~heller/)

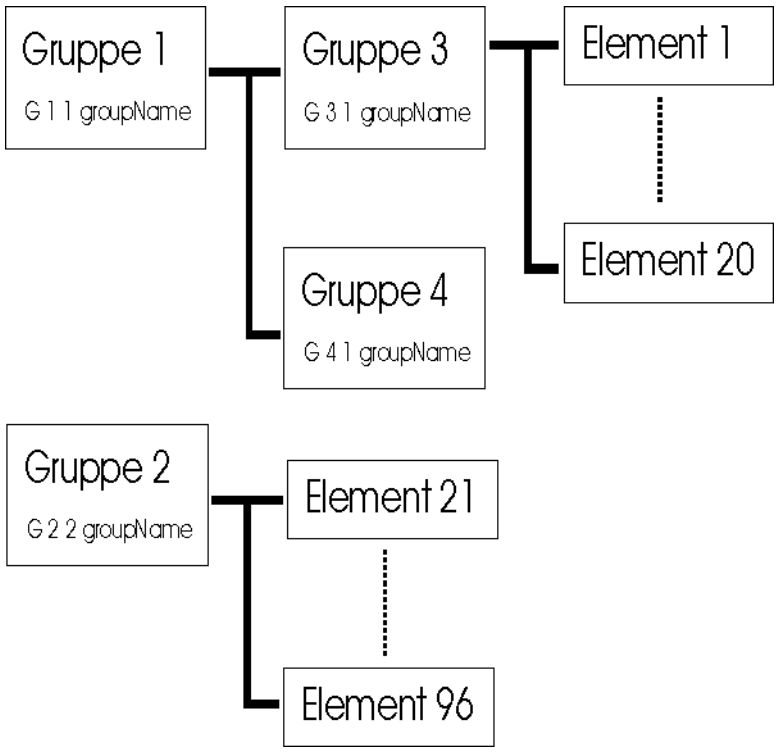
Strukturierung:

In der Datei *dependencies.txt* im Textformat gespeichert. Zunächst werden Gruppen definiert und danach jedes Geometrieelement einer solchen Gruppe zugeteilt.

Hier ein Auszug aus der Strukturierungs- Datei der Schachernhöhle:

```
# G  Group Father Name
G    1    1    Alter Teil
.
.
# A  Series Station Group Color
A    1    0    4    1
A    2    0    4    1
```

Ist der Father einer Gruppe gleich der Gruppennummer, so ist die betreffende Gruppe eine Hauptgruppe und hat keinen Vorgänger. Die Serie- und Stationsnummer eines Elements ergeben eine eindeutige Identifikationsnummer.
Hier ein einfaches Beispiel, wie eine solche Hierarchie aussehen könnte:



Die Gruppen 1 und 2 sind hier die Hauptgruppen ohne Vorgänger. Die Gruppe 1 ist weiter unterteilt in zwei Untergruppen, erst in diesen sind die Geometrieelemente zu finden. Anhand der Identifikationsnummer eines Elements, z.B. eines Ganges, ist es jetzt einfach, die Gruppenzugehörigkeit zu ermitteln.

4 Programmiertechnisches

In diesem technischen Teil, werden kurz die drei von uns verwendeten Programmiersprachen, sowie ihre Vor- und Nachteile beschrieben. Eine detaillierte Einführung zu jeder Sprache ist nicht vorgesehen, im Internet existieren dazu massenhaft Seiten mit Tutorials und Unterlagen (Links dazu auf unserer Vertiefungsblock Homepage).
Der Nutzen dieses Kapitels liegt in der Sensibilisierung des Lesers auf die Probleme, die während der Programmierung aufgetreten sind. Durch die Lösungsansätze werden zukünftige Programmierarbeiten in Richtung VRML-JAVA erleichtert werden.

4.1 VRML

VRML. Virtual Reality Modeling Language. Skriptsprache zur Darstellung dreidimensionaler virtueller Welten in einem VRML-Browser, welcher als Plug-In in einem Internetbrowser (In unserem Fall Netscape Navigator) integriert ist. Bewegungen des Avatars und andere grundsätzliche Eingriffmöglichkeiten des Users in die virtuelle Realität werden direkt im VRML-Browser gesteuert.
Wir arbeiteten in unserem Vertiefungsblock mit VRML Version 2.0. Die endgültige Version des Höhleninformationssystems läuft sowohl im CORTONA wie auch im COSMO-PLAYER Vrml-Browser.
Vorteile von VRML:

- Virtuelle Realität im Internet
- Einfach zu erlernen
- Im Gegensatz zu Java3D, mehrheitlich standardisiert

Nachteile:

- Relativ langsam
- Verschiedene VRML-Browser reagieren zum Teil unterschiedlich auf die Befehle der Progammiersprache.

4.2 JAVA

Java ist DIE Internetprogrammiersprache, obwohl damit nicht nur Internetanwendungen geschrieben werden können.
Die meist verbreiteten Internet-Java-Anwendungen sind sogenannte Java-Applets, kleine Programme, die zusammen mit einer HTML-Seite in den Browser geladen werden.
Java ist nicht zu verwechseln mit Javascript, Javascript ist eine reine Interpretersprache, wo kein Code kompiliert werden muss.
Vorteile von Java:

- Java ist für C++-Programmierer relativ einfach zu erlernen
- Java ist rein objektorientiert
- Java ist portierbar
- Java unterstützt Multithreading

Nachteile:

- Grundsätzlich kein Zugriff auf lokale Informationen von einem Applet aus
- Der compilierte Java-Bytecode muss vor der Ausführung von der lokalen Java Virtual Machine interpretiert werden, dies führt zu Leistungseinbussen.

4.3 EAI (External Authoring Interface)

EAI bezeichnet die Schnittstelle zwischen einem Java-Applet und dem VRML-Browser. Um diese Schnittstelle zu nutzen, ist es notwendig, die mit dem VRML-Browser gelieferten Java-Klassen in den Java-Code zu importieren. Ausserdem muss ein Applet, welches die VRML-Welt manipulieren will, während der Laufzeit eine Referenz auf den VRML-Browser holen. Dann kann man mit EAI direkt auf VRML-Knoten zugreifen, die mit DEF bezeichnet wurden.
→Beispiele folgen in *Aufgetretene Probleme*

4.4 JAVASCRIPT

Die Skriptsprache JavaScript ist eine gemeinsame Entwicklung von Netscape und Sun. Sie ist an Java angelehnt, daher bestehen grosse Aehnlichkeiten.
Unterschiede zu Java:

- Mit JavaScript ist es nicht notwendig den Code zu kompilieren, er kann direkt in ein HTML-File geschrieben werden.
- JavaScript ist viel einfacher konzipiert als Java. Alle Variablentypen werden z.B. einfach mit *var* deklariert.
- In JavaScript gibt es keine Klassen, Methoden und Schnittstellen.

Vorteile von JavaScript:

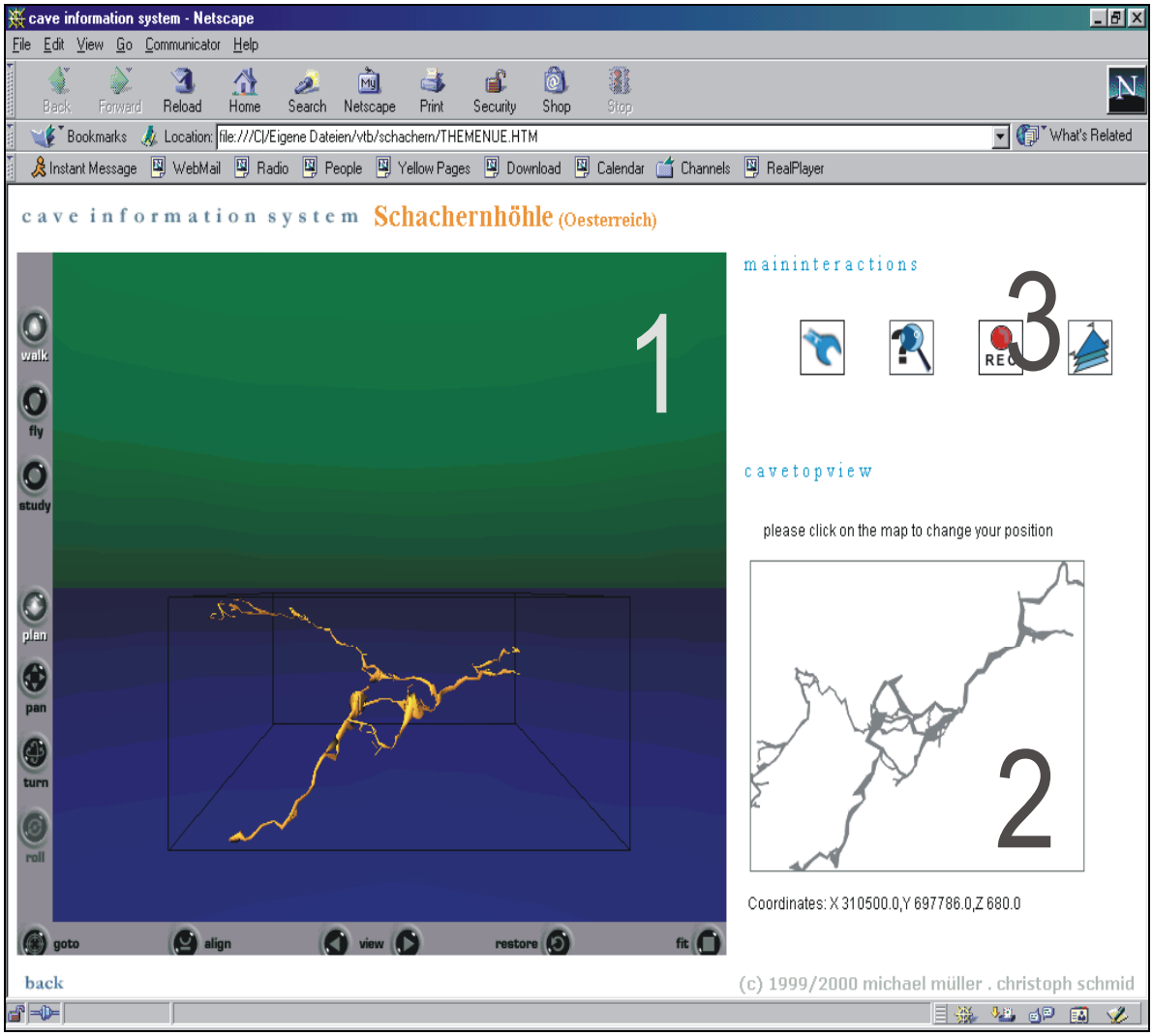
- Code kann einfach analysiert und verbessert werden, ohne neu kompilieren zu müssen
- Sehr einfach zu erlernen
- Kann auf Java-Klassen Methoden zugreifen, die mit Public deklariert wurden

Nachteile:

- JavaScript stösst bei komplexen Problemen schnell an seine Grenzen

5 Cave Information System Bedienungsanleitung

Das C.I.S. lässt sich in drei generell unterschiedliche Bereiche unterteilen:



1: 3D- Darstellung im VRML – PlugIn
2: Übersichtskarte der Höhle (Aufsicht)
3: Cave Information System Hauptmenü

Im folgendenden werden die einzelnen Teile näher beschrieben.

5.1 Das VRML-PlugIn

In diesem Bereich wird die Höhle visualisiert. Dazu muss im Browser ein VRML2.0-PlugIn installiert sein. Die Navigation innerhalb des PlugIns variiert je nach Hersteller und ist den jeweiligen Dokumentationen zu entnehmen. In allen VRML-Playern kann man auch zwischen verschiedenen Navigationsmodi wechseln, das Navigationstempo festlegen und vieles mehr.
Jedes dargestellte Element ist mit einem Berührungssensor versehen und kann angeklickt werden. Es öffnet sich darauf ein Fenster mit Informationen des betreffenden Elementes.

5.2 Die Übersichtskarte

Die Übersichtskarte zeigt eine Aufsicht der Höhle. Ist man als Betrachter innerhalb des dargestellten Bereichs, so erscheint ein schwarzer Punkt, der die Beobachterposition markiert. Unterhalb der Karte wir die aktuelle Position auch noch numerisch als Koordinatentripel für X, Y und Z ausgegeben.

Position neu setzen: Um seine Position neu zu setzen, braucht man bloss einmal auf die betreffende Stelle auf der Übersichtskarte zu klicken. Ein roter Punkt erscheint an diesem Ort und die Position im VRML-Fenster aktualisiert. Der Mauszeiger wird zu einem Kreuz, und man wird aufgefordert, die Blickrichtung durch erneutes Markieren auf der Karte zu definieren. Ist dies getan, so wird im VRML-Fenster die Blickrichtung entsprechend ausgerichtet.

5.3 Das Cave Information System Hauptmenü

Das Hauptmenü besteht aus vier Icons. Jedes davon öffnet ein neues Fenster mit dem entsprechenden Menüpunkt.

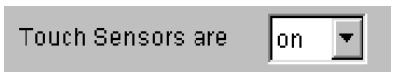
Das Optionenmenü (World Settings)

Im Optionenmenü können verschiedene Umgebungsparameter verändert werden.

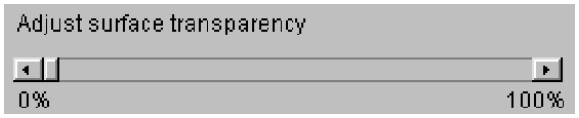
Die Topographie über der Höhle kann zugeschaltet werden, um die Orientierung zu unterstützen.

Das Rechteck um die Höhle, ebenfalls eine Orientierungshilfe, kann weg- und zugeschaltet werden.

Die Navigationsgeschwindigkeit kann eingestellt werden. Diesen Parameter kann man auch im VRML-PlugIn ändern.



Die Berührungs-Sensoren können mit diesem Menüpunkt ein- und ausgeschaltet werden. Bei ausgeschalteten Sensoren ist keine Abfrage über die 3D-Darstellung mehr möglich.



Die Transparenz der Topographie-Darstellung kann variiert werden, um die Sichtbarkeit der Höhle zu optimieren.

Das Abfragemenü (Queries Menu)

Im Abfragemenü können Informationen über Gruppen und Elemente abgerufen werden, Auswahlen können aktiviert sowie deaktiviert oder koloriert werden. Die Aktionen wirken sich teilweise direkt auf die 3D-Darstellung aus.

Panels: Zunächst fallen die drei grossen Panels auf. Sie repräsentieren drei Hierarchiestufen der Höhle. Auf der ersten Fläche sind immer die Hauptgruppen zu sehen. Oberhalb des Panels ist die Zugehörigkeit der aktuellen Ansicht zu sehen, "root" ist das Hauptverzeichnis. Vor jedem Eintrag steht ein + oder ein -, welches anzeigt, ob das Objekt momentan in der 3D-Darstellung angezeigt wird. Ist ein Objekt mit einer speziellen Farbe markiert, erschient ein * vor dem Eintrag.
Öffnen einer Gruppe: Durch Doppelklick auf eine Gruppe wird diese im nächsten Panel geöffnet.
Auswählen der Objekte: In jeder Ansicht, egal ob es sich dabei um Gruppen, Untergruppen, Gänge usw. handelt, kann man die einzelnen Objekte durch einfaches anklicken selektieren. Durch wiederholtes anklicken wird das Objekt wieder aus der Auswahl genommen. Mit den beiden Icons  und  können direkt alle Objekte eines Panels an- und abgewählt werden.
Unterhalb jedes Panels befinden sich die **Aktion-Icons**. Sie haben folgende Funktionen:

 Alle ausgewählten Objekte werden in der 3D-Darstellungangezeigt, falls das nicht schon der Fall ist. Vor den Einträgen erscheint ein +.

 Alle ausgewählten Objekte werden in der 3D-Visualisierung nicht mehr angezeigt. Jeder ausgewählte Eintrag zeigt jetzt ein -.

 Alle ausgewählten Objekte werden in der aktuellen Markierfarbe koloriert. Vor jedem selektierten Eintrag erscheint ein *.

 Die Farbe aller Objekte der Auswahl wird zurückgesetzt.

 Zeigt Informationen zur aktuellen Auswahl an.

 Selektiert alle Objekte des betreffenden Panels.

 Deselektiert alle Objekte des betreffenden Panels.

Im unteren Teil des Menüs befinden sich noch weitere Funktionen:

Die aktuelle Markierfarbe kann gewählt werden. Sie wird aber erst mit  auch angewandt.

Der VRML link bewirkt, dass die Aktionen auch an das VRML-PlugIn weitergegeben werden. Ist er deaktiviert, so wirken sich alle Aktionen nur innerhalb des Abfragemenüs aus. Diese Funktion dient eigentlich bloss Testzwecken.

Ein Fenster mit einer Gesamtübersicht über alle Geometrieelemente erscheint.

Der Flugrecorder (Flight Recorder)

Mit dem Flugrecorder kann in vorgegebenen Zeitabständen die aktuelle Position im 3D-Raum ausgeschrieben werden. Diese Werte können dann in anderen Programmen (z.B. einem Renderer) weiterverwendet werden.

Der Aufnahmevorgang wird gestartet. Das Lämpchen leuchtet auf. Die Werte werden im Textfeld im oberen Teil des Fensters ausgegeben. Die erste Zahl ist die Zeit seit Aufnahmestart in Millisekunden. Danach folgen die drei Koordinaten X Y Z. Die Aufschrift des Knopfes wechselt bei Aufnahmebeginn in 'stop recording'. Bei erneutem Drücken wird der Aufnahmevorgang wieder angehalten.

Das Ausgabefeld wird gelöscht und die Zeit wieder auf Null gesetzt.

Das Zeitintervall, in welchem die Koordinaten registriert werden, kann entweder direkt in Millisekunden eingeben oder mit dem Schieberegler variiert werden.

Das Zeichen, mit welchem die vier Zahlen getrennt werden sollen, kann ausgewählt werden.

Wenn dieses Feld aktiviert ist, so wird nur dann ein Wert herausgeschrieben, wenn sich die Position seit dem letzten ausgeschriebenen Wert geändert hat.

Das Ebenenmenü (Plane Window)

Das Ebenenmenü bietet die Möglichkeit, eine Ebene in den 3D-Raum zu legen. So können zum Beispiel einfache geologische Schichtgrenzen oder etwa der Wasserspiegel simuliert werden.
Öffnet man das Menü, so wird man zunächst aufgefordert, die drei ebenendefinierenden Eckpunkte auf der Übersichtskarte grob zu plazieren. Ist dies getan, so erscheinen drei farbige Kugeln und die durch sie definierte Ebene im VRML-Fenster. Auch in der Übersicht werden die drei Kugeln dargestellt:

Es öffnet sich nun ein anderes Fenster mit verschiedenen Steuerelementen.

Man wählt die Eckkugel aus, auf welche sich die Steuerung auswirken soll.

Mit den drei Schiebereglern X, Y und Z kann die aktive Kugel nun in diesen Koordinaten verschoben werden. Diese Verschiebungen sind relativ zur aktuellen Position und springen deshalb immer wieder auf Null zurück. Die Ebene wird sich den neuen Eckpunkten angepasst.

Die Transparenz der Ebene beliebig von 0% bis 100% variiert werden.

Die Ebene sowie die Eckpunkte werden aus der Darstellung entfernt.

Die Ebene kann von Grund auf auf der Übersichtskarte neu definiert werden.

6 Aufgetretene Probleme

Wie schon erwähnt, wollen wir nachfolgend einige Probleme, die im Zusammenhang mit VRML und Java entstanden sind auflisten, und mögliche Lösungsansätze beschreiben. In den Lösungen steht immer nur der wirklich relevante Programmcode, den Rest sollte man aber mit ein wenig Programmiererfahrung ergänzen können.
Vor jedem Problem steht das Attribut Problemart, welches den technischen Ursprung des Problems spezifiziert.

Wie kann ich EAI und Java-Swing-Klassen zusammen verwenden?
Problemart: Internetbrowser, EAI

Erklärung: Zu diesem Problem konnten wir trotz erheblicher Anstrengungen keine Lösung finden. Die Vermutung liegt nahe, dass es für die heutige Browsergeneration (Netscape Navigator 4.7, Internet Explorer 5.0) gar keine Lösung existiert. In einer VRML-Newsgroup waren auch andere „Experten“ dieser Meinung.
Das Problem liegt darin, dass die Internetbrowser die Java-Swing Klassen nur mit einem speziellen Plug-In von Sun interpretieren und ausführen können. Applets mit Swing-Klassen können nicht mit dem APPLET-Tag in HTML eingebunden werden, sondern mit EMBED (Navigator) und OBJECT (Explorer).
Der Vorteil der Swing-Klassen liegt in der Vielfalt ihrer UI-Komponenten. Was unter den AWT(Abstract Windowing Toolkit)-Klassen selber programmiert werden muss, existiert häufig schon in Swing.
Mit Swing wäre es für unsere Arbeit erheblich einfacher gewesen, intelligent konzipierte Menüs zu kreieren.
Vielleicht wird in der nächsten Generation der Browser Swing schon integriert sein.....

Wie baut man ein Hintergrundbild in eine Java-Panel-Komponente ein?
(Bemerkung: Mit Java-Swing wäre dies ohne weiteres möglich).
Problemart: Java

Problem: Um die UI-Komponenten unseres Informationssystems ansprechend zu gestalten, war es notwendig, Panels mit GIF-Bildern zu hinterlegen um kleine Icons zu entwerfen.

Lösung: Man programmiert eine eigene Panel Art, die von Panel abgeleitet wird. Den Konstruktor definiert man so, dass bei der Erzeugung eines ImagePanels das Bild gleich übergeben werden kann.
Die Klassendeklaration:

```
public class ImagePanel extends Panel { .........
```

Der Konstruktor:

```
public ImagePanel(Image panelIm) {
  panelImage = panelIm;
  try  {
         jbInit();
  }
  catch(Exception ex) {
  ex.printStackTrace();
}
```

Zusätzlich muss die paint-Methode überschrieben werden.
Die neue paint-Methode:

```
public void paint(Graphics g) {
   g.drawImage(panelImage,0,0,this);
}
```

Wie kann man gleichzeitig mehrere Prozesse laufen lassen und ihre Priorität verändern?
Problemart:Java

Problem: Es kann sein, dass gleichzeitig mehrere Prozesse laufen, z.B. der Prozess, der die Bewegung des Avatars in der VRML-Welt überwacht. Manchmal möchte man vielleicht auch einem anderen Prozess Vorrang geben, damit er vom Computer zuerst durchgeführt wird.

Lösung: Multithreading. Multithreading ist in Java voll integriert. Man kann Methoden synchronisieren, damit sie nicht gleichzeitig auf Daten zugreifen. Prioritäten von Threads können verändert werden. Threads lassen sich stoppen oder einschläfern usw.
Ein eigener Thread kann generiert werden, indem man eine Klasse von der Klasse java.lang.Thread ableitet und die Schnittstelle Runnable implementiert.
Beispiel:
Deklaration eines neuen Threads:

```
Thread paintcoord = new GetCoords(500);
```

Implementierung des Thread GetCoords:

```
public class GetCoords extends Thread implements Runnable
{
   GetCoords(int t){
   this.time=t;
   }
   public void run() {
         while (true){
                     try {
                                 sleep(time);
                                 go();
                     }
                     catch (Exception e) { }
               }
   }
   private int time;
}
```

Durch den Aufruf:

```
paintcoord.start();
```

wird der Thread gestartet und alle 500ms die Funktion go() aufgerufen.
Ausserdem kann die Priorität des Threads verändert werden (0-10):

```
paintcoord.setPriority(10);
```

Wie kann man VRML-Code während der Laufzeit mit Java generieren?
Problemart:EAI,VRML

Problem: Es ist manchmal von Vorteil, wenn man einen VRML-Knoten, der nicht im VRML-Code vorkommt, direkt aus Java mit einem String generieren kann. Die Ebene, die wir in unserem Höhleninformationssystem darstellen, lässt sich nirgends im VRML-Code finden. Man kann dadurch beliebige Variablen des Applets in einen String umwandeln und in den Knoten einbauen.

Lösung: Knoten, die man generiert, werden einem bestehenden Knoten als Children angehängt. Als erstes bastelt man irgend einen Gruppen-Knoten (z.B. Transform), der bis dahin noch keine Children besitzt, diesen kopiert man ins wrl-File (VRML-Code).

```
DEF help Transform
{
        translation 0 0 0
        children
        []
}
```

Im nächsten Schritt holt man im Applet eine Referenz auf den help-Knoten:

```
Node root=KoordApplet.browser.getNode("help");
```

Man kreiert den String:

```
Node[] plane1 = browser.createVrmlFromString(
      "Shape{appearance Appearance
          {material Material{diffuseColor 1.0 0.0 0.0 "+
          "transparency "+trans+"}}"
          +"geometry IndexedFaceSet {"
          +"coord Coordinate { point ["+pos1String1+"
          "+pos1String2+" "+pos1String3
          +","+pos2String1+" "+pos2String2+" "+pos2String3+","+
          pos3String1+" "+pos3String2+" "+pos3String3+"] }"
          +"coordIndex [0,1,2] solid FALSE}}"
);
```

und fügt ihn als Children dem help-Knoten hinzu:

```
EventInMFNode addChildren =(EventInMFNode)
root.getEventIn("addChildren");
addChildren.setValue(plane1);
```

analog dazu funktioniert das Entfernen des Knotens:

```
EventInMFNode removeChildren=(EventInMFNode)
root.getEventIn(„removeChildren);
removeChildren.setValue(plane1);
```

Wie können zwei Applets miteinander kommunizieren?
Problemart:Java

Problem: Es existieren zwei Applets in zwei verschiedenen Frames. Eines dieser Applets will eine public-Methode des anderen Applets aufrufen.

Lösung1: RMI Remote Method Invocation Interface. RMI bietet die Möglichkeit , Java-Klassen, welche auf einer anderen virtuellen Maschine laufen, anzusprechen. Dabei ist es egal, ob die virtuelle Maschine lokal vorhanden ist, oder irgendwo im Internet ausgeführt wird. In Java werden zur Umsetzung des RMI-Konzeptes vier Packages verwendet:

```
java.rmi, java.rmi.dgc, java.rmi.registry, java.rmi.server
```

→diese Lösung war in unserem Fall zu aufwendig, da beide Applets auf dem gleichen Computer in der gleichen Virtual Java Machine laufen.

Lösung2: Man könnte im ersten Applet eine statische öffentliche Instanz von sich selber erzeugen, die vom zweiten Applet angesprochen werden kann, da sie, weil statisch, nur einmal an einem ganz bestimmten Ort im Speicher vorkommt.
→obwohl diese Methode programmiertechnisch nicht „schön“ ist, versuchten wir ohne Erfolg, sie zu implementieren. Den Fehler konnten wir bisher nicht finden.

Lösung3: Dieser letzte Lösungsversuch stellt einen Umweg über JavaScript dar. Er ist einfach und funktioniert. Vom ersten Applet wird eine JavaScript-Funktion startFrame() im Frame mapApplet aufgerufen:

```
try {
   getAppletContext().showDocument
   (new URL("javascript:parent.mapApplet.startFrame()"));
}
   catch (MalformedURLException me) {
   System.out.println("url?");
}
```

die wiederum ruft die Funktion start_planeFrame() des zweiten Applets auf.

```
function startFrame() {
   document.KoordApplet.start_planeFrame();
}
```

Wie liest man Daten aus einem File?
Problemart:Java

Problem: Ein Applet soll ein Textfile einlesen, das eine grosse Anzahl wichtiger statistischer Daten enthält. Durch die restriktiven Sicherheitsmassnahmen von Java, ist es Java-Applets grundsätzlich nicht erlaubt, auf lokale Ressourcen zuzugreifen.

Lösung: Wenn sich das Textfile (oder jedes andere File) innerhalb der Klassenressourcen befindet, z.B. im komprimierten jar-Archiv, kann es trotzdem von Applets gelesen werden. Dazu existiert eine spezielle Funktion getResourceAsStream(String filename):

```
InputStream in=getClass().getResourceAsStream("statistic.txt");
```

Zusätzlich erstellt man eine Variable, die den Input-Stream aufnehmen kann.

```
BufferedReader reader = new BufferedReader(new
                            InputStreamReader(in));
```

Um z.B. eine einzelne Zeile einzulesen, schreibt man:

```
String oneline=reader.readLine()
```

Wie holt man eine Referenz des VRML-Browsers?
Problemart:Java

Problem: EAI funktioniert nicht, weil beim Start des Applets keine Referenz zum VRML-Browser vorhanden ist →Null Pointer Exception

Lösung1: Man baut in das HTML-File, welches das Applet aufruft, eine JavaScript Funktion ein, welche den Browser beim Laden der HTML-Seite an das Applet übergibt.

```
function notifyApplet2() {

document.KoordApplet.setBrowser(parent.vrml.document.embeds[0]);

}
<body bgcolor="white" text="#0099cc" onload="notifyApplet2()">
```

Dazu muss im Applet eine Funktion setBrowser existieren.

```
public void setBrowser(Browser b) {
   browser=b;
   isConnected=true;
}
```

Lösung2: Eine reine Java-Lösung:
VRML-Browser und Applet sind im Beispiel in unterschiedlichen Frames plaziert, der Browser ist im Frame „vrml“.
In der start Methode des Applets fügt man folgenden Code hinzu:

```
browser = Browser.getBrowser(this,"vrml",0);
```

Orientierung und Position eines Viewpoints stimmen nicht, wenn ich ihn über EAI setze.
Problemart:VRML, Java

Problem: Sobald der Avatar in der VRML-Welt nicht mehr horizontiert ist, kann man die Orientierung nicht mehr ohne weiteres als Drehung um die z-Achse (y-Achse in VRML) angeben.

Lösung: Bevor man in Java den Avatar an die neue Position und Orientierung bindet, bindet man ihn zuerst an einen (von uns) sogenannten Dummy-Viewpoint, der den jump Parameter auf FALSE gestellt hat. Die darauffolgende Bindung an die neuen Werte sollte dann kein Problem mehr sein.
VRML-Dummy-Viewpoint:

```
DEF Dummy Viewpoint { orientation  0 1 0 0 position 0 0 0 jump
                      FALSE}
```

Die Java-Befehle zum Holen der Knoten:

```
Node n = KoordApplet.browser.getNode("CaveViewpoint");
pos = (EventInSFVec3f) n.getEventIn("set_position");
ori = (EventInSFRotation) n.getEventIn("set_orientation");
bindview = (EventInSFBool) n.getEventIn("set_bind");
Node n2= KoordApplet.browser.getNode("Dummy");
binddummy=(EventInSFBool) n2.getEventIn("set_bind");
```

und zum Setzen der neuen Position und Orientierung:

```
binddummy.setValue(true);
bindview.setValue(false);
pos.setValue(realpos1);
ori.setValue(realori);
```

Welcher Gang wurde angeklickt?
Problemart: VRML

Problem: Wenn ein Gang mit der Maus angeklickt wird, wie weiss man welcher es ist?
Lösung: Man entwickelt einen Prototypen eines TouchSensors, und kreiert eine Instanz davon für jeden Gang. Wenn der Gang aktiviert wurde, leitet er die Gang-ID an die entsprechende Variable der Instanz eines weiteren Prototypen weiter.
Der Variableninhalt kann mit Java über EAI abgefragt werden.

Die beiden Prototypen:
Prototyp für den TouchSensor:

```
PROTO segTouch [
  field SFString segID "t1_0"
  exposedField SFBool touchActive TRUE
  eventOut SFString segtouched]
{
  DEF galleryTouch TouchSensor {
        enabled IS touchActive
  }
  DEF touchOut Script {
        eventIn SFBool touched
        eventOut SFString segtouched IS segtouched
        field SFString segID IS segID
        url      ["javascript:      function      touched(active)
{segtouched=segID;}"]
  }
  ROUTE galleryTouch.isActive TO touchOut.touched
}
```

Prototyp des Empfängers:

```
PROTO whichGallery[
  eventIn SFString segchanged
  eventOut SFString galNr]
{ Script{
        eventIn SFString segchanged IS segchanged
        eventOut SFString galNr IS galNr
        url     ["javascript:     function     segchanged(segname)
{galNr=segname;}"]
        }
}
```

7 Erweiterungsmöglichkeiten

Die Dauer unseres Vertiefungsblockes war viel zu kurz, um alle geplanten Features zu verwirklichen. Das von uns entwickelte Höhleninformationssystem kann jedoch theoretisch beliebig mit neuen Menüs erweitert werden.

Einige Erweiterungsmöglichkeiten sind:

- Zusätzlich zu den Ganginformationen wie Länge und Zugehörigkeit, könnte man ein Foto des Ganges einblenden, wenn man ihn anklickt. Dazu könnte jedem Gang eine URL zugewiesen werden.
- Geologische Informationen. Zum Beispiel in Form von Ebenen, welche die Schichtgrenzen darstellen.
- Markierung der Lage der Höhleneingänge auf der Topographie
- Tool zur Streckenmessung. Wenn man 2 Punkte der Höhle anklickt, erscheint der kürzeste Weg zwischen diesen Punkten und seine Länge.
- Gerade, die bei einem Mausklick durch den Ort des Mausklicks auf der Höhle und die z-Achse definiert ist. Der Schnitt der Geraden mit der Topographie zeigt die Klickposition.
- Menü zum Suchen und Markieren von Gängen. Suche nach Namen oder anderen Attributen.
- Verbessertes Ebenenmenü mit mehreren Ebenen und Informationen zur Lage der Ebenen (Steigung, Neigung).
- Neue Implementierung des Systems mit Java-Swing-Komponenten.

8 Schlusswort

Die Arbeit am *Cave Information System* war spannend und lehrreich. Durch die Kombination verschiedener Werkzeuge konnten wir einen Einblick in die weite Umgebung der Internetprogrammierung gewinnen. Viele der gewonnenen Erkenntnisse können wieder in anderen Gebieten eingesetzt werden. Gerade durch die Tatsache, dass auf dem Gebiet der Höhlenvisualisierung keine professionellen Vorbilder existieren, gewann das Projekt an Attraktivität.
Das am Ende des Vertiefungsblockes vorliegende Programm ist keineswegs fertig, es müsste in Bezug auf Stabilität und Kompatibilität noch optimiert werden. Die Programmstruktur sollte Erweiterungen einfach ermöglichen, einzelne Teile könnten bei einer Neuimplementierung (z.B mit JAVA Swing) weiterverwendet werden.

Wir danken an dieser Stelle Andreas Neumann und Martin Heller, welche unsere Arbeit sehr aufmerksam betreut haben und ihre Erfahrungen weitervermitteln konnten.

9 Internet-Adressen

Das **Cave Information System** mit der Schachernhöhle als Beispiel befindet sich bei http://www.karto.ethz.ch/~schmidc/

Verschiedene Tutorials
http://karto.ethz.ch/

vrml
http://karto.ethz.ch/vrml97/vrml97.htm *VRML-tutorial*
http://www.vrml.ch/
http://o2c.fh-potsdam.de/vrml/pages/newtechnologies/eai/external_authoring_interface/eai_faq.html
http://www.vrmlsite.com/
http://www.wiley.com/compbooks/vrml2sbk/toc/toc.htm Online-Beispiele

java
http://java.sun.com/ *Alles über JAVA*
http://mindprod.com/converterNative.html *Typenkonvertierung*

Höhlenforschung
http://www.speleo.ch/ *Schweiz. Gesellschaft für Höhlenforschung*
http://www.karto.ethz.ch/~an/ *Homepage Andreas Neumann*
http://www.geo.unizh.ch/~heller *Homepage Martin Heller*
http://www.geo.unizh.ch/~heller/toporobot/index.html *TOPOROBOT*

newsgroups
news:comp.lang.vrml
news:de.comp.lang.java
news:comp.lang.java.programmer