SANDBOX: A VIRTUAL REALITY INTERFACE TO SCIENTIFIC DATABASES

by

ANDREW JOHNSON

DISSERTATION

Submitted to the Graduate School
of Wayne State University,
Detroit, Michigan
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

1994

MAJOR: COMPUTER SCIENCE (Database)

Approved by:

____

Adviser   Date

____

____

____

Dedication

I dedicate this to my family,

and especially to my father,

who would have understood.

Acknowledgements

I would like to thank my adviser Dr. Farshad Fotouhi for his insightful comments and discussions. I also want to thank my other committee members: Dr. Bill Grosky for his thoughts and his vast library of journals and proceedings, Dr. Ishwar Sethi and Dr. Frank Westervelt for their valuable contributions. I would also like to thank Dr. Robert Reynolds and Dr. Vipin Chaudhary for their help.

I would like to thank the members of the Electronic Visualization Laboratory at the University of Illinois at Chicago for their help with the CAVE; Narendra Goel, John Norman, and Don Strebel for their insight into the FIFE database; and Larry Cathey for loaning me the one GL manual that made my implementation possible.

I would like to thank Peter Gabriel, Sarah McLachlan, Kitaro, and REM for providing the background music for this work.

I would like to thank H. G. Wells, Jules Verne, Ray Harryhausen, Inoshiro Honda, Eiji Tsubaraya, Akira Ifukube, Gene Roddenberry, Patrick McGoohan, Ray Bradbury, Arthur C. Clarke, Robert Heinline, and Larry Niven for inspiring me in my youth.

I would like to thank the two Steves for creating the Apple][- a truly marvelous toy, and a great place to start hacking.

I would like to thank all the friends along the way: Preston, Tamaki, Kristen, Rob, Lee, Gary, Dave, Bill, Lorna, Ray, Abad, Elena, Imran, Colin, Dawn, Jonathan, Ward, Mike, Greg, Bob, Rick, Chris, and many others.

I would especially like to thank my friend Jason Leigh.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years there has been much work devoted to collecting and analyzing large amounts of data from scientific experiments. This trend will continue into the future with even larger amounts of data being collected, stored, and analyzed. A principle problem facing users of such information systems is finding appropriate data without detailed knowledge of the structure of the stored data.

Relational databases are designed to deal with limited ranges of data on specific topics. The form of the data is known ahead of time and the database tables and their relationships are clearly defined before the data is entered. Scientific databases [Mic91] contain a much larger amount of data on many different, but related topics. The form of the data is not known ahead of time as the data is collected by investigators from a wide range of disciplines.

Scientific data is often stored using flat files or relational databases. This huge amount of interrelated data is forced into the rigid table structure of a relational database which can not adequately model the necessary relationships [Kim90]. Unlike typical relational databases, scientific databases store much larger amounts of information, have very few updates, need to maintain the data for longer periods of time, and contain more complex data. Each discipline sees different relationships between sets of data so each discipline stores data in its own way, making it very difficult for investigators in other fields to access the data.

The user of a scientific database is interested in obtaining certain specific information as quickly and painlessly as possible. My research involves improving access to scientific databases by using two methods: using clustering to make the structure of the database malleable, and using a virtual reality interface to hide the database completely from the user.

The first solution involves clustering the tables of a scientific database based on the user's queries. Instead of forcing the users to conform to the structure of the database, we can mold the structure of the database to the needs of the individual users, and thereby reduce their confusion when interacting with the database. Visualizing the structure of the scientific database and then making that structure malleable allows each user to mold the scientific database to their own needs.

The user declares what topic they are interested in, and then interacts with the database as usual. The system keeps track of which parts of the database are accessed. In this method, the content of the tables is irrelevant, only the fact that the user sees a relationship between the tables is important. This information can then be used to cluster the tables of the database. Users from different backgrounds can use these clusters to quickly index into the database by seeing which areas of the database are likely to contain information on their topic. This flexibility will be increasingly important as the size, breadth, and accessibility of scientific databases increases.

The second solution involves using virtual reality and scientific visualization to hide the database from the user. Typically virtual reality and scientific visualization are used after data has been extracted from a scientific database, but they can also be used to make retrieving data from the database easier, and more intuitive.

Much of the data that is stored in scientific databases was collected through experimentation. Using virtual reality an investigator can 'recreate' the original experiment, collecting data from the scientific database in much the same way that the original data was collected. The user places virtual instruments into a virtual reenactment of the original experiment and retrieves data from the database without ever directly interacting with the database. I call this interface the SANDBOX: Scientists Accessing Necessary Data Based On eXperimentation.

The rest of this dissertation is organized as follows: Chapter 2 discusses previous related work. Chapter 3 discusses my first solution - clustering. Chapter 4 discusses my second solution - the SANDBOX. Chapter 5 discusses my implementation of the SANDBOX. Chapter 6 discusses my conclusions and plans for future work in these areas.

Chapter 2

Related Work

2.1 Introduction to Related Work

This chapter on previous work related to my dissertation is divided into four main parts. Section 2.2 discusses related work in the field of scientific databases. Section 2.3 discusses related work in the field of hypertext. Section 2.4 discusses related work in the field of adaptive algorithms. Section 2.5 discusses related work in the field of virtual reality.

2.2 Scientific Databases

Scientific databases contain very large amounts of information. They tend to have very long life spans and very few updates. Some of the disciplines which are currently creating scientific databases include: the earth sciences [HS89, SSAE93, SCN+93], engineering [LLL89], medicine [ACF+93, FLS89], and physics [BG91].

It is difficult to visualize the overall structure of scientific databases due to the large number of files or tables containing the data. A researcher may need information from several distinct tables in the database, but can not find which tables contain the required information. Once the current tables have been found, the user must write an appropriate query linking all these tables together to extract the needed information. This becomes even more difficult if the user needs to access image data [Jai92]. There is simply too much information in a scientific database and its users get lost among all the columns and tables. There is too much information overhead.

Scientific databases also contain a large amount of meta-data - data about the data (notes, drawings, etc.) It is very difficult to include this information in current databases, yet this meta-data can be a great benefit to those trying to access the data. It is important to give the user convenient access to this information.

Database management systems provide information using a query-response methodology where the information is stored in a set of tables. The user asks for information using a query language (such as SQL) which performs operations on the tables (combining tables together, selecting specific columns from a table, etc.) and returns the requested information to the user in a new table. The user is required to learn the database's specific query language in order to extract information. The researcher is not interested in learning a query language. The researcher simply wants to retrieve information from the database. Allowing the user to build up a query without a query language means one less obstacle to retrieving the necessary data.

Scientific databases contain an overwhelming amount of information that needs to be used by a wide variety of researchers. Much of the difficulty in accessing data in scientific databases comes from the enormous amount of data that is involved; but the organization of this data is also a large problem. Users from various scientific communities see different relationships between sets of data. Certain information is important to certain investigators and certain information is not.

The creator of a scientific database must balance the needs of all the users and compromise on the storage of the information. The database schema is often designed around the efficient storage of the information, rather than the users' concepts of the information. Every user must then adapt to this generic view of the data. Instead of each investigator adapting to this single generic view of the information. It would be better to adapt this view to each user. This way each user sees the same information in their own unique way. When each user can see the data in their own way, it reduces the amount of information overhead in the database.

Scientific databases are accessed by users from a wide range of disciplines, mostly unfamiliar with databases and their associated query languages. These users need to search for specific pieces of data quickly, and browse through related information to see if it is of value to them. They need to relate information from different tables in the database.

For a more thorough introduction to scientific databases and their problems see [FJP90, Mic91]. These user interface issues are not limited to scientific databases, but are also important issues in database systems in general [SAD$^+$93].

2.2.1 Current Solutions

When an interface to a scientific database is designed, its creator typically imposes a generic structure on the database - a hierarchical menu system allowing the user to move through an ordering of the files or tables. This gives researchers from all backgrounds a way to access the data, but each of the researchers must conform to this generic structuring of the data. This approach has several shortcomings: 1) The menu system does not provide enough flexibility for a wide range of researchers, 2) The users may not know enough about the domain to make appropriate choices, 3) It does not help the user with ill-defined queries.

Graphical query languages have been proposed to simplify the interface [KM89, OW93]. Graphical query languages make the database schema more visible, reduce typing, and allow users to rely on recognition rather than memorization. This approach has several shortcomings: 1) The schema of scientific databases are so large, and complicated that the user rapidly runs out of screen real-estate, 2) The graphical metaphor quickly becomes cumbersome for complicated queries.

Ioannidis, et all [ILH92, ILH$^+$93] developed a graphical interface for the management of scientific experiments and data using the Object-Oriented data model MOOSE. The user interacts with the database through the schema. The system makes large schemas more manageable by allowing the user to hide parts of the schema, collapse sections of the schema into nodes, and use reference nodes to eliminate long arcs. While useful for scientists involved in the original experiment, this approach has several shortcomings for users less familiar with the original experiment: 1) The users may not know what data is available, 2) The users may not know enough about the domain to make appropriate choices, 3) It give users a variety of choices without sufficient descriptive material to make that choice, 4) The original schema may not match the relationships seen by all users.

Graphical interfaces to statistical databases have been proposed which represent the attributes of a statistical database in a graphical form [RR90]. The user can then browse through this graphical form. These systems have problems similar to those for graphical interfaces mentioned previously.

Other related work includes replacing current data visualization systems [WRH92] (e.g. AVS [Ups89], Khoros [RY92]) with visualization tools such as Stonebraker's Tioga [SCN$^+$93] built on top of more powerful database components. These systems rely on a data flow visual language to move data through a series of predefined operations. Of course, before the user can visualize the data, the user must find the appropriate data in the database.

Ahlberg, et all [AWS92] experimented with using graphical widgets to formulate database queries. Graphical visualization was used to show the contents of a small database (the periodic table) and the results of the queries. By hiding the database schema and allowing the user to interact directly with the data values, they found that the users gained a faster understanding of the data than with queries based on textual interfaces. Expanding on this, and allowing the user to have a more realistic interaction with the data values of a much larger database should give the user a more intuitive way of accessing their data.

Hypertext has been proposed as a way to give users the capability to browse through the meta-data associated with scientific databases [Ste88]. This gives the users better understanding of the contents and organization of the database. Expanding on this, and allowing the user to browse through the data in the database as well as the meta-data should give the user a better understanding of the relationships among the various data.

2.2.2 FIFE

FIFE is a scientific database. Developing techniques to determine surface climatology from satellite observations is the objective of the ISLSCP (International Satellite Land Surface Climatology Project.) FIFE (First ISLSCP Field Experiment) was undertaken at a 20km by 20km square site near Manhattan, Kansas in 1987 and 1989. Its purpose was to gather enough data to allow the creation and testing of models to develop these techniques [SH89, SNO89].

120 gigabytes of data was collected (300 megabytes textual data, the rest image data including satellite photos and site photographs taken on the ground.) The textual data fills over 100 tables in a relational database. Each experiment is given its own table with the attributes containing the numerical data collected during that experiment. These tables typically have 10-30 attributes each, and over 100,000 rows. The textual data is currently available on-line [SHS$^+$90]. A subset of the textual

and graphical data has been made available on a set of five CD-ROMs from NASA [SLN$^+$91, SLN$^+$92c, SLN$^+$92a, SLN$^+$92b, SNL$^+$93].

2.3 Hypertext

Year by year we are dealing with more and more information. It is not just the storage and retrieval of this information that is important, but also the ability to access related pieces of information. Hypertext systems encourage the user to browse through information as they would browse through a museum, rather than querying a database of facts.

2.3.1 Hypertext Definitions

A hypertext system connects information into a graph structure where related nodes of information are connected by links. Using computer screens to display information allows the user to interactively browse through these nodes of information using many different paths.

A node (or information element) is a piece of information. A node can contain text, a picture, a sound, a piece of animation, or a combination of the above. The amount of information in a node varies with the flexibility of the hypertext system and the author of the hypertext itself. A node contains one or more links to other nodes.

The simplest form of link moves the user from one node to another. A link is traversed by activating an anchor (a word, picture, or special icon in a node.) When the link is traversed a new node appears on the screen either in addition to, or replacing, the previous node. These links connect the nodes into a network (or web, or information-space.)

The author of the hypertext creates the nodes and links. The user uses the hypertext. The author may also be the user of the hypertext. Hypertext systems and applications vary the amount of power given to the user. Some applications (software engineering, writing support) give the user the power of the author to change nodes and links, while others (museum displays) limit the user's power to browsing through, but not modifying, the hypertext.

Hypertext systems encourage browsing: using the links to move node by node through the network until you find the information you are looking for. Many hypertext systems include a limited query feature as an index into the network, but browsing is the main method of finding information. Browsing allows users to find something when they don't know exactly what they are looking for. The user does not have to learn a query language, or learn how to formulate the request in terms a query language can process. The user gains knowledge (facts and relationships) by browsing through the hypertext. This is often described as moving through the information space (or hyper-space.)

Knowledge is stored not only in the individual nodes of information but in the relationships between linked nodes. Each user makes their own path through the information network based on their own interests, allowing for less cumbersome structuring of the information. The user has more direct contact with the information, and is more involved with structuring the information in their own way, compared to a traditional database system.

The term hypertext was coined in the mid 1960s, well before the current interest in multimedia technology. The first implementations of hypertext were limited by their display hardware to linking text. Newer hypertext systems link images, sound, and animation, so the term hypermedia has begun to replace hypertext.

There are several good introductions to hypertext: [Con87, Nie90a, TKM+91].

2.3.2 Hypertext Applications

Hypertext systems are currently being used for a wide variety of tasks:

Hypertext can be used to relate large amount of different types of information in Software Engineering [CFG91, GS87], and computer aided design [DS86].

Hypertext has had the most success in data retrieval: a hypertext medical handbook [Fri88, FC89], chemistry journals [ELK+91], an electronic form of the Oxford English dictionary [RT88], a hypertext encyclopedia [Sch87, WB85], a departmental information system [JFG+94b], and on-line manuals [Wal87].

Hypertext has changed writing [CFCP92, DL91, Ess91, LK92] and enhanced collaborative work [CBY89, CB88, SCG89].

2.3.3 Hypertext History

Hypertext has had a fairly long history.

In 1945, Vannevar Bush [Bus45] described the memex - a system for managing a person's information. Based on microfilm and photography, the ideas for hypertext were present, but not the necessary technology. In 1965, Theodor Nelson coined the term hypertext and extended the idea of the personal memex to the global Xanadu. Xanadu would be a global information space - all the world's libraries available electronically.

The first hypertext systems were created in the late 1960s: Engelbart's NLS in 1968, Nelson and van Dam's HES in 1968, van Dam's FRESS in 1969. Work would continue through the 1970s with Newell, Akscyn, and McCracken's ZOG, and van Dam's Electronic Document Project.

Hypertext began to expand out of the research labs in the 1980s: Akscyn and McCracken's KMS [AMY88] in 1981, Halasz's Notecards [Hal88, Tri88] in 1983, Delisle and Schwartz's HAM [CG88a, DS86] in 1984, and Meyrowitz's Intermedia [CBY89, YHMD88]. Commercial packages would appear in the late 1980s. Owl's Guide became the first popular commercial hypertext system in 1986. Schniederman's HyperTIES [Sch87] and Apple's Hypercard [WK90] followed in 1987. Internet based hypertext systems appeared in the early 1990s such as WWW and NCSA's Mosaic [And93].

For a comparison of current several hypertext systems, see [SLKB88].

2.4 Adaptive Algorithms

Adaptive algorithms attempt to mimic more 'natural' methods of learning.

Adaptive algorithms have been used successfully in information retrieval [Gor88], classification problems [CG88b, MH88], clustering [AC93, KSB93, Sae90, Sny93] and 'solving' NP problems in a reasonable time [AVL88].

2.4.1 Genetic Algorithms

Genetic algorithms were first developed by John Holland in the mid 1970s [Hol75]. They perform searching in a manner similar to natural selection in nature. By mimicking "survival of the fittest," genetic algorithms try to 'evolve' a solution to a search problem. A population of possible solutions is used. The more fit members of this population (i.e. those that are nearer to the solution) are more likely to mate and produce the next generation than those members of the population who are less fit. As the generations pass, the members of the population should get fitter and fitter (i.e. closer and closer to the solution.)

Genetic algorithms accept their input coded as a finite length string (or chromosome.) Each of the elements in the chromosome is a gene, and each gene has an allele value. The allele values are elements of a finite alphabet. A set of these chromosomes form a population.

The search problem is coded into the number of genes in a chromosome and the possible allele values. The space of possible solutions to the search problem is bounded by the number of permutations of genes that are possible.

The population changes from one generation to the next through reproduction, crossover, and mutation. The number of chromosomes in each iteration remains constant, but their fitness should generally improve. The fitness of a chromosome is determined by a payoff function which depends on the allele values of each chromosome.

The 'goal' of the genetic algorithm is to produce a chromosome with the highest possible fitness; that is, the sequence of allele values that most closely matches the optimal point in the search space. The payoff function converts one entire chromosome into a single number. When the payoff function has been applied to all of the chromosomes their relative fitness can be determined. Each of the chromosomes in the population is initially given random allele values. The more fit chromosomes of the previous generation reproduce to produce the chromosomes of the next generation.

Sometimes when two chromosomes reproduce, they crossover. When chromosomes W and X crossover to form chromosomes Y and Z, a certain set of chromosome W's genes are copied into identical positions in Y. The remaining genes in chromosome W are copied into Z. Genes from chromosome X fill in the remaining positions in Y and Z. That is, each of the children inherits some of the traits of each of their parents. Crossover occurs frequently, typically 50% to 90% of the time. The result of the crossover may be a more fit chromosome, or a less fit chromosome.

Crossover only mixes existing allele values at each gene. Mutation causes random changes in the allele values, allowing for more variation. Mutation happens rarely, typically less that 1% of the time. The result of the mutation may be a more fit chromosome, or a less fit chromosome.

For a more thorough introduction to genetic algorithms see [BBM93a, BBM93b, Gol89, Whi93].

2.4.2 Neural Networks

A neural network is a distributed network of small processing elements. This large number of simple processing units (neurons) are connected together forming a miniature version of the human brain. Neural networks have had a good deal of success in pattern recognition and learning by example.

A self organizing feature map is a neural network developed by Teuvo Kohonen [Koh90, Koh93a, Koh93b, RMS92]. It is a two dimensional array of processing elements. Each element contains a weight vector. The input to the self organizing feature map is a set of weight vectors. The self organizing feature map forms itself into a topological ordering of the input data through unsupervised competitive learning.

In [Koh90] Teuvo Kohonen described his self organizing feature map as follows:

> It is a sheet-like artificial neural network, the cells of which become specifically tuned to various input signal patterns or classes of patterns through an unsupervised learning process.

The weight vector W_{ij} of each element of the M by M array is initialized with random values. An input vector I is chosen at random. The neuron whose weight vector W_{ij} is closest to I is found ($min \mid W_{ij} - I \mid$). This neuron and all the neurons within a certain square neighbourhood (N_c by N_c) of this neuron have their weights adjusted, bringing the weights closer to I. Random inputs are repeatedly provided while the amount of adjustment (α), and the neighbourhood shrink in a form of simulated annealing. The amount of adjustment and the size of the neighbourhood eventually become negligible and the learning ceases.

When the learning stops, the network has organized the set of n-dimensional inputs into a two dimensional array. For each input there is one processing element with the closest matching weight vector. Labeling each neuron with the name of the input vector that it matches most closely gives the topological ordering of the inputs.

For a more thorough introduction to neural networks in general see [Koh88, Nie90b].

2.5 Virtual Reality

Virtual Reality makes use of three dimensional computer generated environments to create an immersive, interactive interface to information. The major difference between virtual reality and traditional display methods is that virtual reality is immersive. The user is not outside, looking in. The user is inside, looking around.

2.5.1 Virtual Reality Definitions

The goal of virtual reality systems is to provide a totally immersive virtual environment. This involves the use of three-dimensional computer graphics, motion, and sound.

In order to create this virtual world there is a great deal of interaction between the user and the environment. The user moves through the environment, and interacts

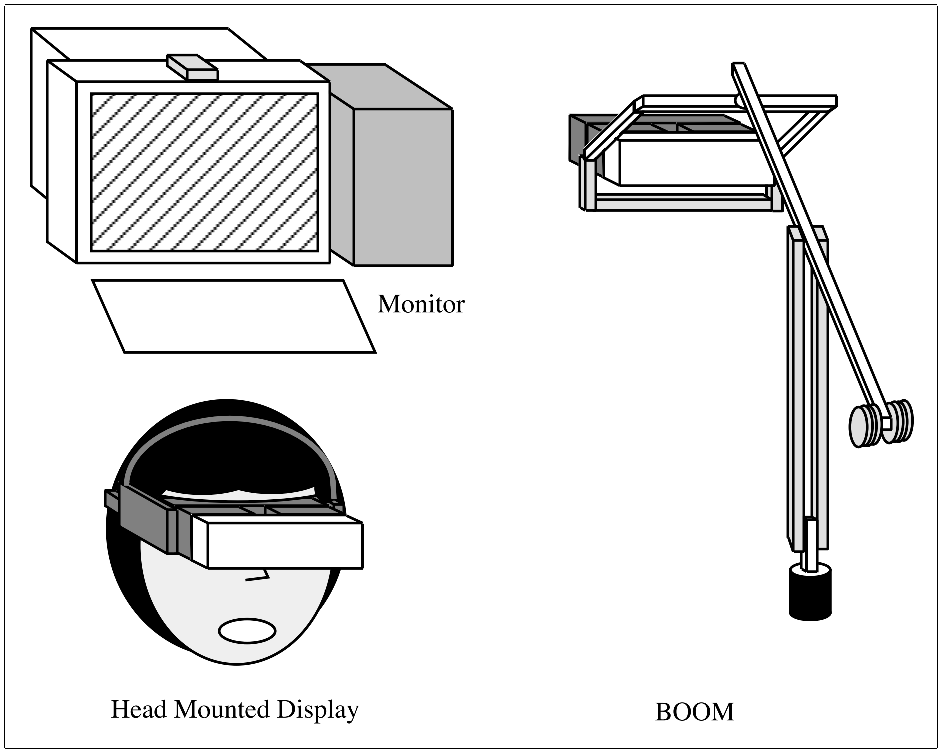


Figure 2.1: Virtual Reality Hardware

with objects in the environment. The system monitors the user's position and actions. The system gives feedback to the user. This feedback can be visual, audible, or tactile.

The most obvious feature of virtual reality is its use of wide angle three-dimensional graphics. Displaying these graphics to the user requires the use of a regular monitor and stereo glasses (fish tank VR [AB93]), or an HMD (Head Mounted Display [Tei90]), or a BOOM mounted display (Binocular Omni-Oriented Monitor [MBP$^+$90]), or a projection based display (CAVE [DSC93]). This display gives the user their perspective in the three-dimensional world. See Figure 2.1.

Vision is only one of our five senses, so acoustic [GSO91, Ast93, BBK$^+$93] and tactile (haptic) [Iwa90] feedback are also being experimented with.

A virtual reality system also needs to know the user's head position, head orientation, and hand location in the virtual world. This tracking can be done mechanically (with a boom mounted display), or through a set of emitters and sensors (acoustic, electromagnetic, inertial, optical, etc.)

The virtual reality system also needs to allow the user to interact with it. This interaction can be done through devices like the Data Glove [ZLB+87] or the GROPE system [BOYBK90].

As we go through life, the world does not flicker. At the theatre, still images pass by our eyes fast enough that we see continuous motion. For virtual reality systems to be useful they need to display at least 15 stereo frames per second [WS82] (meaning 15 frames for each eye.) Thirty stereo frames per second gives most people jitter-free vision. This requires very fast graphics hardware and often parallel computer systems.

There are several good introductions to virtual reality: [AB92, Cru93, Wex93].

2.5.2 Virtual Reality Applications

Virtual reality is currently being used in several applications.

There is much current work in the field of scientific visualization [CRM91, MRC91, Rib91, RMC91, Web93], taking large amounts of scientific data and using sophisticated computer graphics to visualize the structure and patterns of the data. Scientific visualization is an effective means of bringing order to an overwhelming amount of information. However, graphics are not the end product of analysis, they are only a byproduct. Virtual reality enhances scientific visualization by allowing the user to move around and through three-dimensional representations of the data giving the user better interaction with the graphics and a better understanding of what they represent [Koi93] .

Virtual reality is being used to interact with computer generated simulations (e.g. simulating the flow of virtual air around a virtual airplane in a virtual windtunnel

[BL91], simulating virtual weather over a virtual piece of land [HS89], simulating the excitations of neurons in the brain [LDL$^+$93] or simulating the movements of stars and galaxies [SN93].) The computer generates the data from models and then virtual reality is used to interact with the model.

Walkthroughs and virtual design allow a person to walk through a building or see a design before any construction is begun [FB89].

Telepresense makes use of virtual reality to allow a user in a virtual reality environment to control an actual instrument in an actual environment as though the user was in the actual environment (i.e. in an inhospitable place such as at the bottom of the sea, or in outer space.) The sensory feedback that the user receives from the virtual reality hardware gives the user better control over the object than more traditional interfaces [BF90].

Virtual reality educational tools allow students experience remote, inhospitable, or even nonexistent places and to perform experiments that would be inconvenient, expensive, or even impossible to perform any other way [LEB93].

2.5.3 Virtual Reality History

Virtual reality has had a fairly short history. In 1960, Morton Heling created Sensorama where a user can see, hear, smell, and feel several prerecorded experiences. In the mid 1960's Ivan Sutherland proposed "the ultimate display" [Sut65] which he described as follows:

> The ultimate display would, of course, be a room within which the computer can control the existence of matter. ...With appropriate programming such a display could literally be the Wonderland into which Alice walked.

In the late 1960's and early 1970's Ivan Sutherland built the first head mounted display ("the Sword of Damocles") and Frederick Brooks built an early force-feedback system (GROPE II.) In the early 1980's several types of head mounted stereoscopic displays were created at a reasonable cost. In the late 1980's and early 1990's Fake Space Labs developed a boom mounted display (the BOOM [MBP$^+$90]), and the Electronic Visualization Lab developed a virtual reality room (the CAVE [CSD$^+$92].)

2.5.4 The CAVE

The CAVE (Cave Audio Visual Experience Automatic Virtual Environment) is a projection based virtual reality system [CSD$^+$92]. The user enters a 10 foot by 10 foot by 10 foot room where images are projected onto the three of the walls and the floor. When the user dons a pair of lightweight StereoView LCD shutter glasses, the projected images fill the room and surround the user. The user is given the freedom to move around the room reasonably unencumbered, and can walk around or through virtual objects in the CAVE. Since they can see their own bodies, users have a true sense of being inside the virtual environment.

To interact with the virtual objects in the CAVE, the user carries a physical three button wand. The user's position and the wand's position are tracked by Ascension Technology Flock of Birds trackers. One tracking sensor is mounted on top of the StereoView glasses giving the position and orientation of the user's head, and the other on the wand, giving the position and orientation of the wand. The CAVE is controlled by a single SGI Onyx/RE^2.

See Figure 2.2 for a diagram of the CAVE hardware, and Figure 2.3 for a photograph of the CAVE at the Electronic Visualization Laboratory at the University of Illinois at Chicago. There are currently three CAVEs in operation (Argonne National

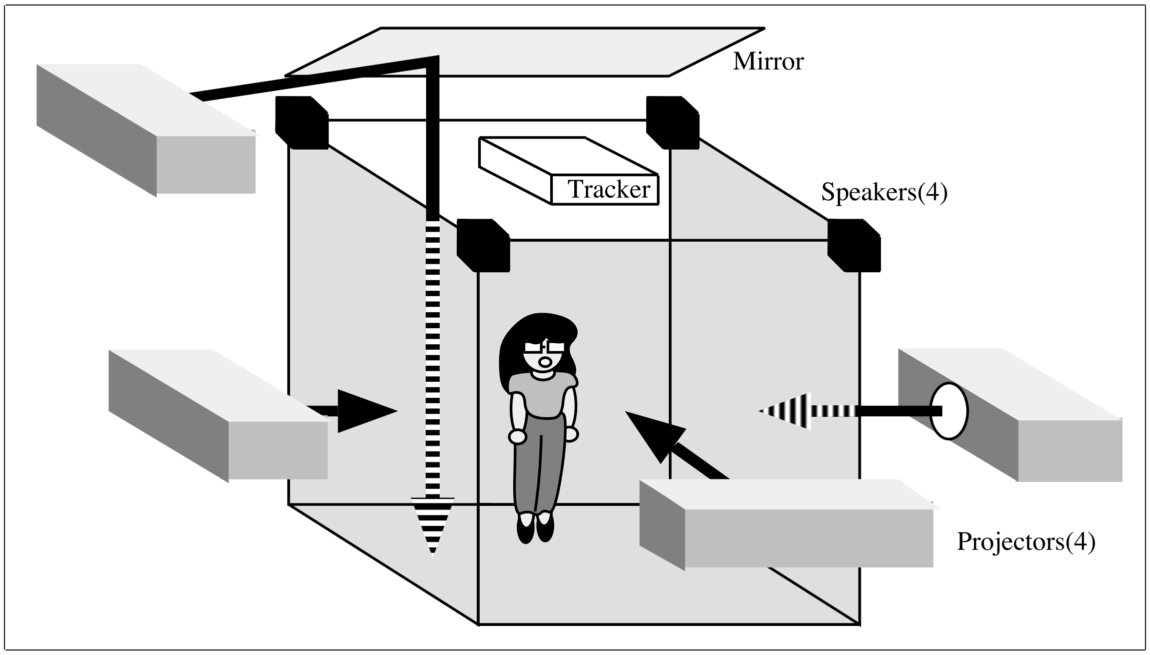


Figure 2.2: Diagram of the CAVE

Laboratory, National Center for Supercomputer Applications, and at EVL) and a fourth being built.

Scientists at Caltech, the University of Minnesota, the University of Chicago, Argonne National Laboratory, the National Center for Supercomputing Applications and the University of Illinois at Chicago use the CAVE in their research [CLB+93].

Given the large volumes of data that may need to be displayed, the CAVE has a great advantage over head mounted display technology. During prolonged data access the virtual environment freezes. This can cause the user of a HMD to become disoriented as the movement of their head is no longer reflected by a change in their 3D view. In the CAVE, while the virtual environment is still frozen, users can see their bodies, and the surrounding CAVE, thereby reducing disorientation and vision induced nausea. Over 10,000 people have been inside the CAVE and only two have experienced enough nausea to complain about it.

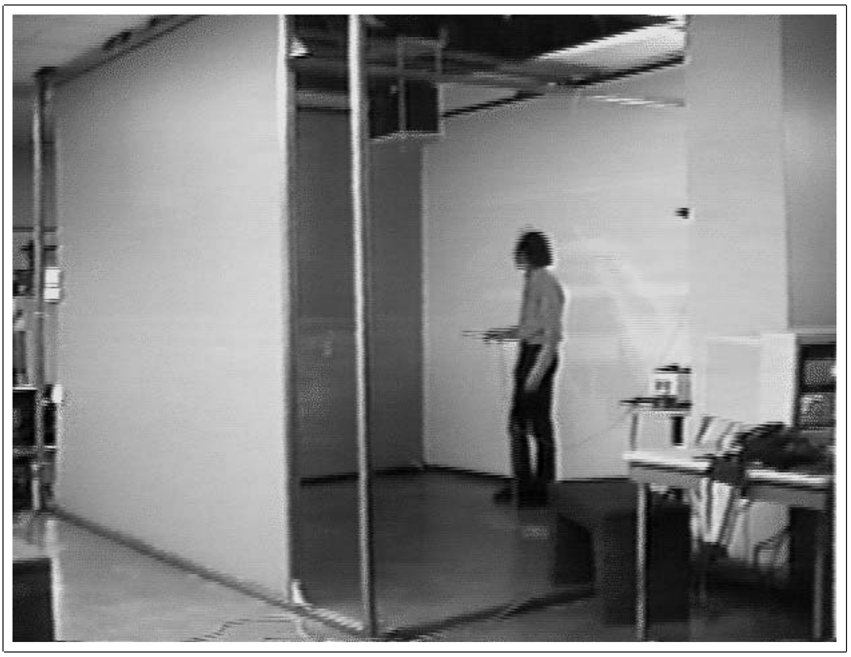


Figure 2.3: Photograph of the CAVE

Chapter 3

Clustering

3.1 Introduction to Clustering

As scientific databases grow larger it becomes more and more important to cluster their tables. Clustering reduces information overhead. It allows for higher level concepts (groups of tables), allows for the breaking of a single large scientific database into appropriate modules, allows for views over the database, and allows the user to make changes at the cluster level without affecting the scientific database itself.

When the database tables have been clustered, the user has the choice of browsing through the existing generic interface, or using a hierarchy of clusters as an index to move quickly to the appropriate tables in the database. The original scientific database remains untouched, however each user now has a personal index into that database. See Figure 3.1.

As new users begin to work with this database, they can choose from the existing clusterings. A new user can choose to look at the database from a biologist's point of view, or a climatologist's point of view. This gives each user a starting point nearer their own needs than the generic interface. Experienced users will also be able to access their data faster and more conveniently because the database will adapt to their interests.

Clustering in databases has typically been used to reduce access times to commonly used records [FGK78, LY77, YSL85]. The database itself is physically altered to decrease access time for the current types of queries being issued. Clustering in

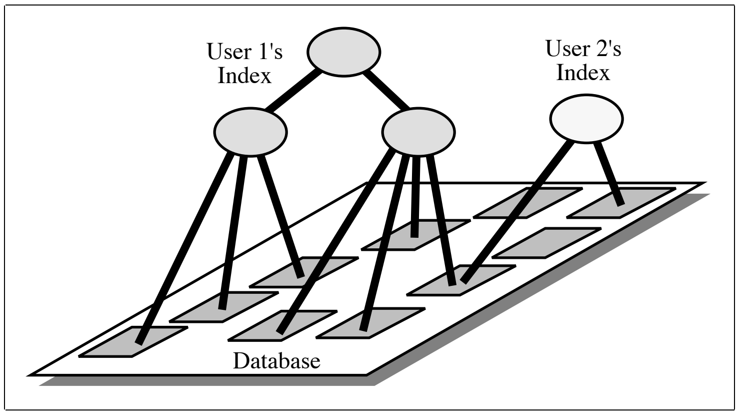


Figure 3.1: Hierarchical Index into the Scientific Database

document retrieval systems is typically used to improve the appropriateness of documents retrieved [FB91, Gor88, ZMSD92].

By monitoring the user's queries into the database we can cluster the scientific database based on the connections the user makes between the tables. As each user generates queries and retrieves relevant information from the scientific database, the system learns what information in the scientific database is important to this individual user. The system learns which tables are important, which columns are important, and what kinds of linkages the user creates. This knowledge is then used to generate clusters in a form of user-oriented clustering [DR86, RD87, OM87, YSLT81].

Those tables that the user has found important are made more visible, and more easily accessible. In commonly used databases this will give the user quick access to important blocks of information, and several starting points for beginning new searches. The clusters can be shared between users. This gives new users a choice of several ways to view an unfamiliar database, and gives regular users a way to see the database from different perspectives.

We can use an adaptive algorithm to delineate the clusters. Finding the optimal clustering of the tables is equivalent to checking every partition of the set of tables. This is an exponential problem (there are over 5 trillion different clusterings of 19 tables) so it is too costly to find the optimal clustering for a database with more than a few tables. Using an adaptive algorithm such as genetic algorithms or neural networks we can find a very good clustering within a reasonable amount of time.

Once the clusters have been generated, they can be used to reduce the overall structure of the scientific database and make it easier to find information. This gives each user simplified, personal access to the parts of the scientific databases of importance to them. Extraneous information is hidden from the user and the important information is made readily accessible.

These clusters can then be used by the same user later on to further ease the job of creating and modifying queries. These clusters can also be used by other researchers in the same field. While the physical scientific database has a generic structure useful to no-one, the logical clusters that each user creates will be very useful to other investigators with similar interests. These clusters can be used to give new users a starting point. Instead of being overwhelmed by this gigantic database, a new user can choose to view the database in a particular way. The new user can now see the scientific database as others in their discipline do.

Section 3.2 discusses how information is collected to generate the clusters. Section 3.3 discusses the two different techniques used to generate the clusters. Section 3.4 discusses my implementation of these two clustering techniques. Section 3.5 discusses how these clusters, once generated, can be used. Section 3.6 discusses how parallelism can be used to reduce the clustering time. Section 3.7 gives some conclusions on this clustering method. Section 3.8 discusses enhancements to these clustering techniques.

Parts of this research were previously published as [JFG94a, JF].

3.2 Information Collection

Users retrieve data from a relational database using a query language like SQL. I am monitoring the user's queries, and storing information about the tables accessed in a list. Since the physical database itself is not being clustered, lists can be used to support multiple sets of clusters over the same database simultaneously. This allows clustering to be based on topic, or on time. Lists created for different topics, or lists created at different times can be used to cluster the tables of the database. Without storing the user information in lists, only a single, current, clustering can be maintained.

Each line of the list contains information on a single table in the database. This line lists all of the columns that have been displayed for that table, and all the columns that have been used to connect this table to other tables in the database. The format of each line of the list is shown below:

table$_i$: {selectCol$_i$,}*:{joinCol$_i$-joinCol$_j$@table$_j$,}*.

where:

- table$_i$ is the table we are interested in.
- selectCol$_i$s are the columns in table$_i$ that have been selected.
- joinCol$_i$s are the columns in table$_i$ that have been joined with columns joinCol$_j$ of table table$_j$.

Three small sample SQL queries are converted into their list form as shown in Figure 3.1.

Query 1	List 1	
SELECT $A.\phi$, $B.\chi$, $C.\epsilon$	A:ϕ,	:α-β@B, α-γ@C,.
FROM A, B, C	B:χ,	:β-α@A,.
WHERE $A.\alpha = B.\beta$	C:ϵ,	:γ-α@A,.
AND $A.\alpha = C.\gamma$;		
Query 2	List 2	
SELECT $C.\rho$, $C.\omega$, $D.\lambda$	C:ρ, ω,	:ρ-σ@D,.
FROM C, D	D:λ,	:σ-ρ@C,.
WHERE $C.\rho = D.\sigma$;		
Query 3	List 3	
SELECT $A.\alpha$	A:α,	:δ-β@B,.
FROM A, B, C	B:	:β-δ@A, β-ρ@C,.
WHERE $A.\delta = B.\beta$	C:	:ρ-β@B,.
AND $B.\beta = C.\rho$;		

Table 3.1: Converting Queries into Lists

When a user begins to search for information in the database they either choose to work with an existing list or create a new empty list. The user can give this new empty list an appropriate name. As the user works with the database, the list keeps track of the tables and columns the user is accessing through their queries.

The mechanism for creating the query (straight query commands, a graphical query language, etc.) is unimportant. Eventually the query in the form of tables and columns is given to the database and this query updates the listing.

A user can create one list for all of their queries, or create multiple lists where each concept the user is interested in is given its own list. The lists can be unioned together. A user can keep their lists private or share them with other users of the scientific database. A list can therefore be one of a user's lists, the union of a user's lists, the union of a group's lists, the union of a site's lists, or the union of all the lists available on the system.

I have used a similar technique to collect information on a user's browsing through

Table	List 1 Percent	List 2 Percent	List 3 Percent	Neighbouring Tables
A	50	0	50	BC
B	50	0	50	AC
C	33	33	33	ABD
D	0	100	0	C

Table 3.2: Converting Lists into Percentages

a hypertext document to cluster the nodes based on the user's usage patterns [JF93]. The scientific database user, like a hypertext user, is faced with an overwhelming amount of interrelated information presented in a generic way. Clustering allows us to add a personal interface layer on top of the generic interface reducing information overhead and speeding up access to the underlying data.

3.3 Clustering

The lists that have been created by the information gathering process are used to cluster the tables of the scientific database. The number of times each table is used in each list, divided by the total over all the lists, is used to generate usage percentages. For each table, every joined table in each of the lists is added onto the roster of neighbouring tables.

The converted versions of the lists created in Table 3.1 are shown in Table 3.2. Since Table A was used once in List 1, and once in List 3, Lists 1 and 3 have usage percentages of 50% while List 2 has a usage percentage of 0%. Table A was joined with tables B and C in the queries so tables B, and C are neighbours of table A. As the number of lists increases, the usage percentages decrease towards zero for the commonly used tables. This decreases the effectiveness of the clustering algorithms as the amount of difference in the usage patterns decreases.

To avoid this pitfall, the percentages are scaled so that the average usage percent-

Table	List 1 Percent	List 2 Percent	List 3 Percent	Neighbouring Tables
A	63	0	63	BC
B	63	0	63	AC
C	50	50	50	ABD
D	0	100	0	C

Table 3.3: Final Inputs to the Clustering Algorithms

age is set to 50% rather than (100/number of lists)%. The final converted versions of the lists shown in Table 3.1 are shown in Table 3.3.

$$Val' = \begin{cases} \frac{Val}{100/number\ of\ lists} \times 50 & \text{if val} \leq (100/\text{number of lists}) \\ \frac{Val-(100/number\ of\ lists)}{100-(100/number\ of\ lists)} \times 50 + 50 & \text{otherwise} \end{cases}$$

These values, the usage percentages and the neighbour roster, are used to cluster the tables.

3.3.1 Genetic Algorithm Clustering

In my approach, I use a population of 10 chromosomes. Each chromosome represents a set of clusters; each gene represents a table in the database; each allele value represents a cluster. Genes with the same allele value are in the same cluster. Each table is a member of at most one cluster, and the number of clusters varies from 1 to n where n is the total number of tables being clustered. Each table can form its own unique cluster in the chromosome, if necessary.

Figure 3.2 shows a sample chromosome that would be used to cluster the data in Figure 3.1. The four genes (A, B, C, D) in the chromosome represent each of the four tables involved in the clustering. Tables A and B are in the same cluster as they both have an allele value of α. Table C forms its own cluster with an allele value of β, and table D forms its own cluster with an allele value of δ. As there are four tables (A, B, C, D), there are four possible clusters (α, β, γ, δ.)

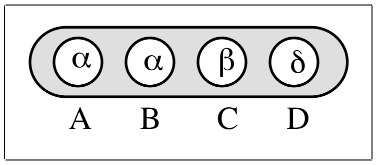


Figure 3.2: Sample Chromosome

As the ordering of the genes is irrelevant in this problem, I use uniform crossover, rather than 1-point crossover. Instead of using the standard genetic algorithm technique of performing crossover by breaking two chromosomes at the same point and swapping the end parts, I generate a random percentage. Each gene has that percentage chance of being swapped between the two chromosomes.

In addition to crossover and mutation I implemented two other operations to alter the chromosomes: fusion and fission. Fusion takes two unique allele values and combines them into a single allele value. Fusion is used to combine two clusters into one. Fission takes a single allele value and gives it a different random allele value. Fission is used to break a cluster apart, where each table in that cluster is randomly assigned to another (possibly empty) cluster.

The percentage possibility of each operation is given below:

$Fusion$	75%
$Crossover$	50%
$Fission$	25%
$Mutation$	2%

These values were found experimentally to yield good overall results for a variety of problem sizes.

The payoff function is composed of two separate partial payoff functions: weight similarity, and neighbourlyness. Each partial payoff function's values range from 0.0 to 10.0.

Weight similarity, computed by Equation 3.1, promotes clusters containing tables with similar weight percentages, indicating similar usage patterns. For every cluster, the difference between each table percentage and the average percentage of the cluster is computed and scaled into the range 0 to 10. The closer the percentages are, the smaller this ratio will be, and the larger the overall payoff will be. Weight similarity pushes the genetic algorithm to create smaller clusters.

$$10 \times (1 - min(\frac{\sum_{tables} \sum_{lists} \mid table\% - aveCluster\% \mid}{\# \ of \ tables}, 1)) \tag{3.1}$$

Neighbourlyness, computed by Equation 3.2, promotes clusters containing neighbouring tables. The roster of neighbours shows which tables have been joined. For each pair of tables that are connected by a join, at least one user must feel these two tables are related because that user joined them. Each connection in the input lists either connects two tables in the same cluster, or two tables in different clusters. Neighbourlyness sums up the number of connections between tables in the same cluster and divides the total by the total number of connections between tables in the lists. Neighbourlyness pushes the genetic algorithm to create bigger clusters.

$$10 \times \frac{\sum_{clusters} \# \ of \ joins \ between \ tables \ in \ cluster}{\# \ of \ joins} \tag{3.2}$$

Neighbourlyness tries to bring all the connected tables together. Weight similarity tries to isolate tables with similar usage patterns. Together, this pulling together and pushing apart generates the clusters. Equation 3.3 shows how the two partial payoff functions are combined.

$$payoff = neighbourlyness \times weightSimilarity^2 \tag{3.3}$$

This results in an overall payoff function value that ranges from 0 to 1000.

Each table is initially put into its own cluster. The genetic algorithm is started. After each new generation is created the most fit chromosome of the previous generation replaces the least fit chromosome of the new generation. This preserves the best set of clusters from one generation to the next. The genetic algorithm stops when the value of the most fit chromosome remains constant over 200 generations, or the number of generations reaches a set maximum value.

If there is more than one resulting cluster, these clusters are given back to the genetic algorithm. The weight percentages, and neighbours of the components of each cluster are used to generate the weight percentage, and neighbours for the cluster as a whole. Each cluster is now treated like a table. The clusters are clustered forming a hierarchy until one global cluster remains.

When the clustering is complete, the genetic algorithm will have placed all of the tables into the leaf nodes of the cluster hierarchy. Now a breadth first traversal is made of the cluster hierarchy from the leaf nodes to the root. Each table is taken in turn. If it is a neighbour of all of the tables in its own node and in each of its siblings' nodes then it is moved up one level in the tree. If this leaves an empty node with no children, then that node is removed. This way common nodes are moved up the tree through the internal nodes, and the depth of the tree is reduced.

3.3.2 Neural Network Clustering

In our clustering algorithm the weight vectors contain the usage percentages for the various usage lists being used to cluster the database. If there are k lists involved in the clustering, the weight vector will contain k values. The size of the M by M array is set to $\lceil \sqrt{number\ of\ tables}\ \rceil$. Each table can form its own unique cluster in the array, if necessary.

If we used a self organizing feature map to cluster the data in Figure 3.1, we would use a 2 by 2 array. Each element of the array would be a weight vector of length three (one for each list.) The set of input vectors would consist of four weight vectors (one for each table) of length three:

$$\begin{array}{c}(0.63, 0.00, 0.63)\\(0.63, 0.00, 0.63)\\(0.50, 0.50, 0.50)\\(0.00, 1.00, 0.00)\end{array}$$

The self organizing feature map orders itself with regards to similarity in the weight percentages and the neighbouring tables.

For each iteration in the learning process, one table is chosen at random. The neuron with the most similar set of weights is identified. That neuron and all the neurons within the current neighbourhood have their weights adjusted bringing them closer to the weights of the chosen table.

The weight vectors are adjusted as follows:

$$W_{ij_{t+1}} = \begin{cases} W_{ij_t} + \frac{\alpha_t}{2} \times (I - W_{ij_t}) & \text{if node is within } N_{c_t} \\ W_{ij_t} & \text{otherwise} \end{cases}$$

One neighbour of the chosen table is selected at random. The chosen table and all the neurons within the current neighbourhood have their weights adjusted bringing them closer to the weights of the selected neighbouring table.

The weight vectors are adjusted as follows:

$$W_{ij_{t+1}} = \begin{cases} W_{ij_t} + \frac{\alpha_t^2}{20} \times (I - W_{ij_t}) & \text{if node is within } N_{c_t} \\ W_{ij_t} & \text{otherwise} \end{cases}$$

The neighbourhood and the amount of adjustment (α) are then adjusted. If α is greater than zero, another iteration is performed.

α is adjusted as follows:

$$\alpha_0 = 1$$

$$\alpha_{t+1} = \alpha_t - 0.0001$$

Abbreviated Table Name	Table Description
AEROLOG	measurements from surface flux group
AIR_FLUX	aircraft flux from flights over the konza
AMS_87	ave from ncar's pams, army corp's dcps
AMS_89	ave from ncar's pams, army corp's dcps
AMS_STAT	# of reports from each ams station
BIOMASS	plant biomass weight, nitrogen content
BRUT	actual radiosonde data observations
CLOUD	cloud estimates from liverpool cameras
FIFE_SITE	reference info. on the collection sites
GRAV	soil moisture readings at 25,75,150mm
NEUTRON	soil moisture with 200cm neutron probe
RAD_FLUX	measurements from surface flux group
RAIN_DAY	daily rainfall data by site and date
SOIL_PROP	soil properties measured historically
SOIL_GAS	no2 flux, co2 from soil respiration
VEG_SPEC	species composition data by site, date
WIND_PRO	noaa lidar wind profile data

Table 3.4: FIFE Table Descriptions

the neighbourhood is adjusted as follows:

$$N_{c_t} = M \times \frac{\alpha_t}{2}$$

These values were found experimentally to yield overall good results for a variety of problem sizes.

After the array has been generated each table is taken in turn. Its set of weight vectors are compared to the elements of the array and the best match is found. Each array element is then labeled with the set of tables claiming it as their best match. Thus all of the tables are mapped to their appropriate spot in the array.

3.4 Clustering Implementation

I tested my approach on a subset of NASA's FIFE scientific database.

Five queries utilizing 17 tables were made to the database showing the kind of

List	Table
Rainfall	FIFE_SITE, AMS_87, AMS_89, AMS_STAT
Meteorological	FIFE_SITE, AMS_87, AMS_89, AMS_STAT, RAIN_DAY, CLOUD
Atmospheric	FIFE_SITE, WIND_PRO, BRUT, AIR_FLUX, RAIN_DAY, RAD_FLUX, AEROLOG
Surface Biophysical	FIFE_SITE, BIOMASS, SOIL_PROP, SOIL_GAS, CLOUD, VEG_SPEC
Soil Moisture	FIFE_SITE, CLOUD, NEUTRON, GRAV, BIOMASS, SOIL_PROP

Table 3.5: FIFE Queries

queries that researchers in different, but related fields would create. Each of these queries was stored in its own appropriately named list. The tables which made up those queries are shown in Table 3.4. The tables used in each of the five queries are shown in table 3.5.

Figure 3.3 shows how the genetic algorithm creates clusters. The 'n', 'w', and 'f' fields show the values of each of the partial payoff functions (neighbourlyness, weight similarity) and the final payoff value for each set of clusters. The 'str' field shows the tables and which clusters they are assigned to (each cluster has a unique ASCII character.) Initially, as each table forms its own cluster, the weight similarity is very high, but the neighbourlyness very low. As the clustering proceeds the overall payoff rises as the tables combine into appropriate clusters.

This figure shows how clusters are generated from the individual tables. Once these clusters are created the process is repeated with the newly formed clusters being clustered. When only a single cluster remains, the second phase of genetic algorithm clustering is run to compact the hierarchy and move common tables toward the root.

Figure 3.4 shows how the two phases of the genetic algorithm clustering relate. Phase 1 results in a very deep tree with all of the individual tables at the leaf nodes. Combining neighbouring tables together where possible collapses the tree and moves

```
Population Report - Generation:   1
   Num    n    w      f                str
   0:     0  100      1   !"#$%&'()*+,-./01
   1:     2   84      1   )"#$%&'()*+,'./01
   2:     0   88      1   !"#!%&'()*+,-./01
   3:     0   88      1   !"#$%&'()*%,-./01
   4:     0  100      1   !"#$%&'()*+,-./01
   5:     0  100      1   !"#$%&'()*+,-./01
   6:     1   81      1   !"#$%&'(&*+,-.'01
   7:     1   94      1   !"#$%&'()*+,'./01
   8:     1  100      1   !"#$%&'()*+,-.001
   9:     1   92      1   !"#$%&'()*+,-./(1
```

```
Population Report - Generation: 200
   Num    n    w      f                str
   0:    53   64    206   &&(((!&!$!!&$!!!&
   1:    42   42     73   &&,,,,&#,#!&,!!#&
   2:    53   64    206   &&,,,!&!+!!&+!!!&
   3:    51   71    251   &&,,,!&!/!!&$!!!&
   4:    51   71    251   &&,,,!&!/!!&$!!!&
   5:    42   64    165   &&,%,!&!$!!&$!!!/
   6:    70   48    162   &&,,,!&!&!!&&!!!&
   7:    39   61    115   /&,,,!&!/!!0$!!!0
   8:    49   60    145   ..,,,!.!,!!.$!!!&
   9:    53   52    136   /&,,,!&!/!!&&!!!&
```

```
Population Report - Generation: 400
   Num    n    w      f                str
   0:    44   70    197   $$111"$"---$("""$
   1:    49   75    229   $$111"$""!!$("""$
   2:    40   82    273   $$111"$"-!!$("""$
   3:    54   62    196   $$111"$"1!!$$"""$
   4:    39   65    127   $$111"$"-!1$("""$
   5:    40   82    273   $$111"$"-!!$("""$
   6:    40   67    181   $$111"$"-!!$!"""$
   7:    44   76    233   $$111"$"-""$(""0$
   8:    40   68    185   $$111"$"$!!.("""$
   9:    44   61    149   ($111"("---((""" (
```

```
Population Report - Generation: 600
   Num    n    w      f                str
   0:    34   82    205   $$111.$."!!$)...'
   1:    40   49     97   $$111.$.+$$$)...$
   2:    40   82    273   $$111.$.+!!$)...$
   3:    54   38     76   $$111+$++11$1+++$
   4:    40   71    201   (())+.(.+'!(+...(
   5:    60   60    217   ((111.(.(!!((...(
   6:    34   82    205   $$111.$.+!!$)...'
   7:    32   60    109   $$1+1.$.+!!$!...'
   8:    28   82    137   $$111.$(+!!$)...)
   9:    32   76    178   $$111.$.)!!$)#.#$
```

```
Population Report - Generation: 800
   Num    n    w      f                str
   0:    40   58    133   $$--!.$.-##$#...$
   1:    40   82    273   $$111.$.-!!$#...$
   2:    56   62    191   $$111.$.-..$1...$
   3:    35   57    100   $$111"$"1##$#.."$
   4:    49   61    153   $$111.$.!!!$1...$
   5:    32   47     67   $)111...1!!$#....
   6:    46   74    221   $$111.$.-!!$1...$
   7:    46   64    169   $$111.$.&&-$1...$
   8:    39   60    109   0$111.$.###$#...$
   9:    39   48     70   $$111.$.-11$1&..$
```

```
The best clustering found is:
     $$111.$.-!!$#...$
```

```
Num: Chromosome Number
n:   Neighbourlyness Payoff
w:   Weight Similarity Payoff
f:   Overall Payoff Function Value
str: Allele values of Chromosome
```

Figure 3.3: Genetic Algorithm Generating Clusters

common tables to the interior nodes.

Figures 3.5 and 3.6 show how the neural network generates clusters. Figure 3.5 shows the actual numeric contents of the array, and then graphically displays those values for each of the lists. Zero is shown as black, ten (A) is shown as white. Figure 3.6 shows where the best match for each table can be found. At the beginning the values are randomized, as are the best matches. As the clustering continues specific areas of the array become attuned to specific sets of tables.

The genetic algorithm takes 25 seconds to return the clusters shown in Figure 3.7. The neural network takes 30 seconds to return the clusters shown in Figure 3.8.

The genetic algorithm and neural network create similar clusterings of the database tables. The genetic algorithm generated 6 clusters and formed these clusters into a hierarchy. The neural network generated 8 clusters and formed these clusters into one overall cluster.

The tables within those clusters are similar. The two clusters in the center of the neural network array (FIFE_SITE and CLOUD) each get a cluster at the top of the genetic algorithm hierarchy. The cluster at the bottom right of the array (WIND_PRO, BRUT, AIR_FLUX, RAD_FLUX, AEROLOG) gets its own cluster in the hierarchy. The cluster at the upper left of the array (VEG_SPEC, SOIL_GAS) gets its own cluster in the hierarchy. The two clusters at the upper right of the array (BIOMASS, SOIL_PROP, and NEUTRON, GRAV) are combined into a single cluster in the hierarchy and the two clusters at the bottom left of the array (AMS_87, AMS_89, AMS_STAT, and RAIN_DAY) are combined into a single cluster in the hierarchy.

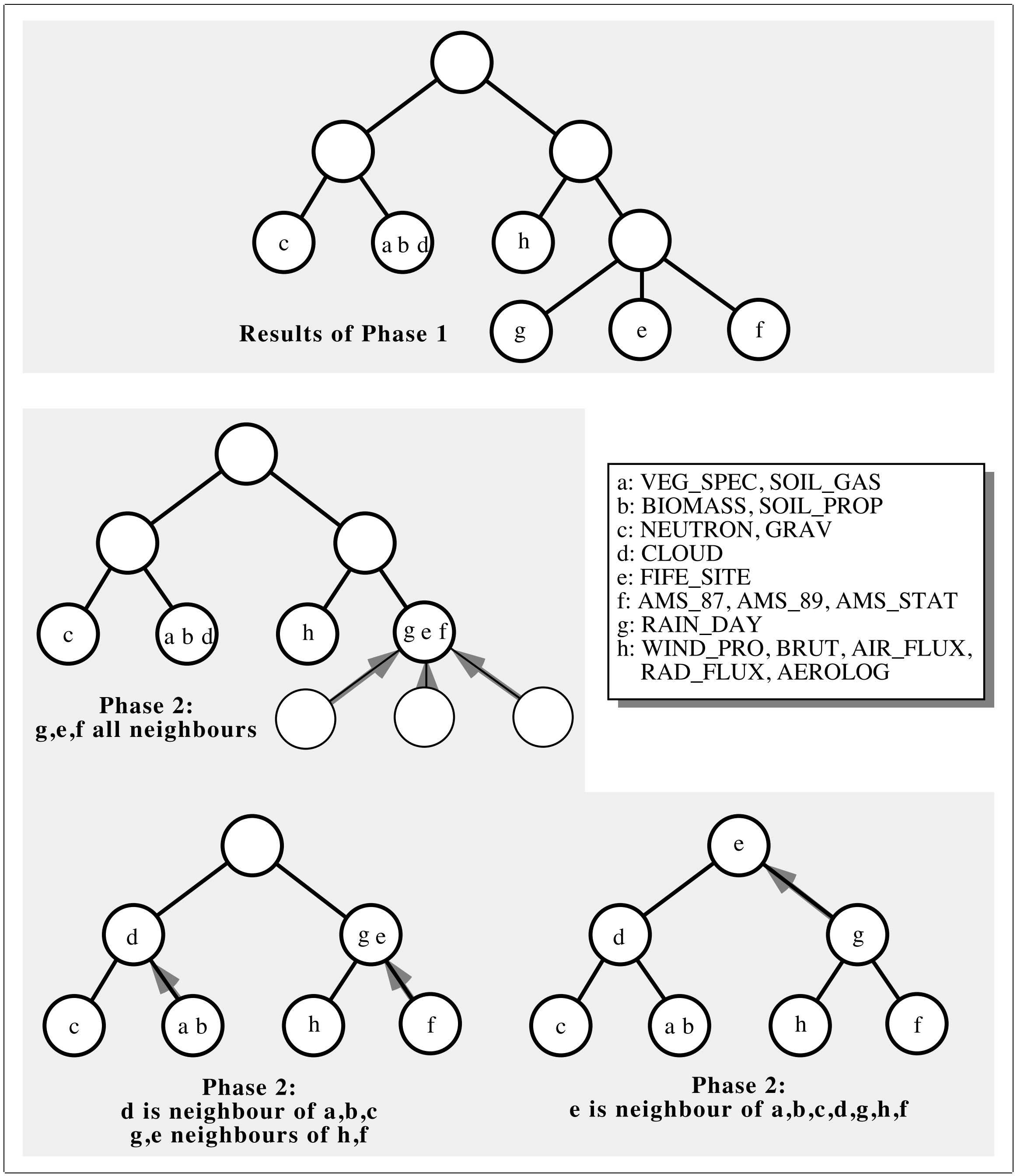


Figure 3.4: Two Phases of the Genetic Algorithm Clustering

```
Start:
36772.86867.57017.63174.85157.
58247.45664.46645.44556.7517A.
44686.66465.44646.56464.49225.
49A68.44464.54554.64645.35136.
32307.9968A.48441.97477.06357.

1/4 Done:
00047.00047.00326.00326.11911.
11147.11147.00326.00326.11911.
22136.22137.00326.00326.01911.
33225.33226.11415.01515.11812.
55422.45323.22711.11812.11812.

1/2 Done:
01091.01083.01066.01048.0001A.
22282.12174.03066.01048.01019.
34336.35355.15255.01712.00802.
56333.46344.16344.01900.01A00.
77000.57100.15600.01A00.01A00.

3/4 Done:
000A0.00092.00077.00039.0000A.
000A0.11192.12166.11148.0001A.
44040.44152.35355.12524.01604.
67000.56222.26444.12811.01A00.
77000.67111.26622.02900.01A00.

Done:
010A1.00092.00077.00039.0001A.
000A0.11192.06066.11148.0000A.
44040.44152.55555.12524.01604.
77000.56222.26444.12811.00A00.
67011.67111.07700.02900.01A00.
```

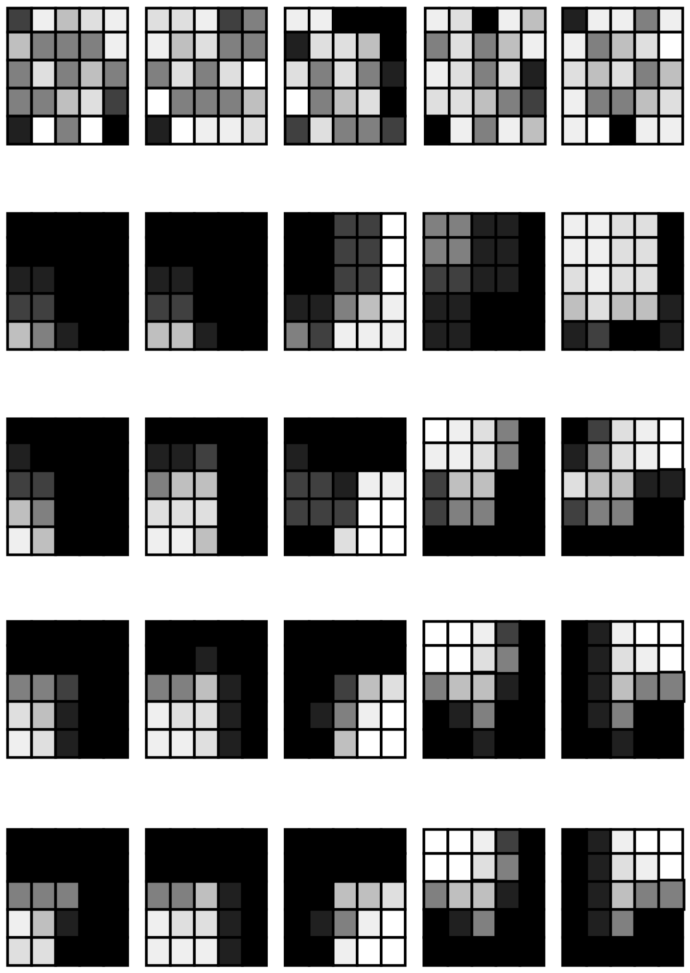
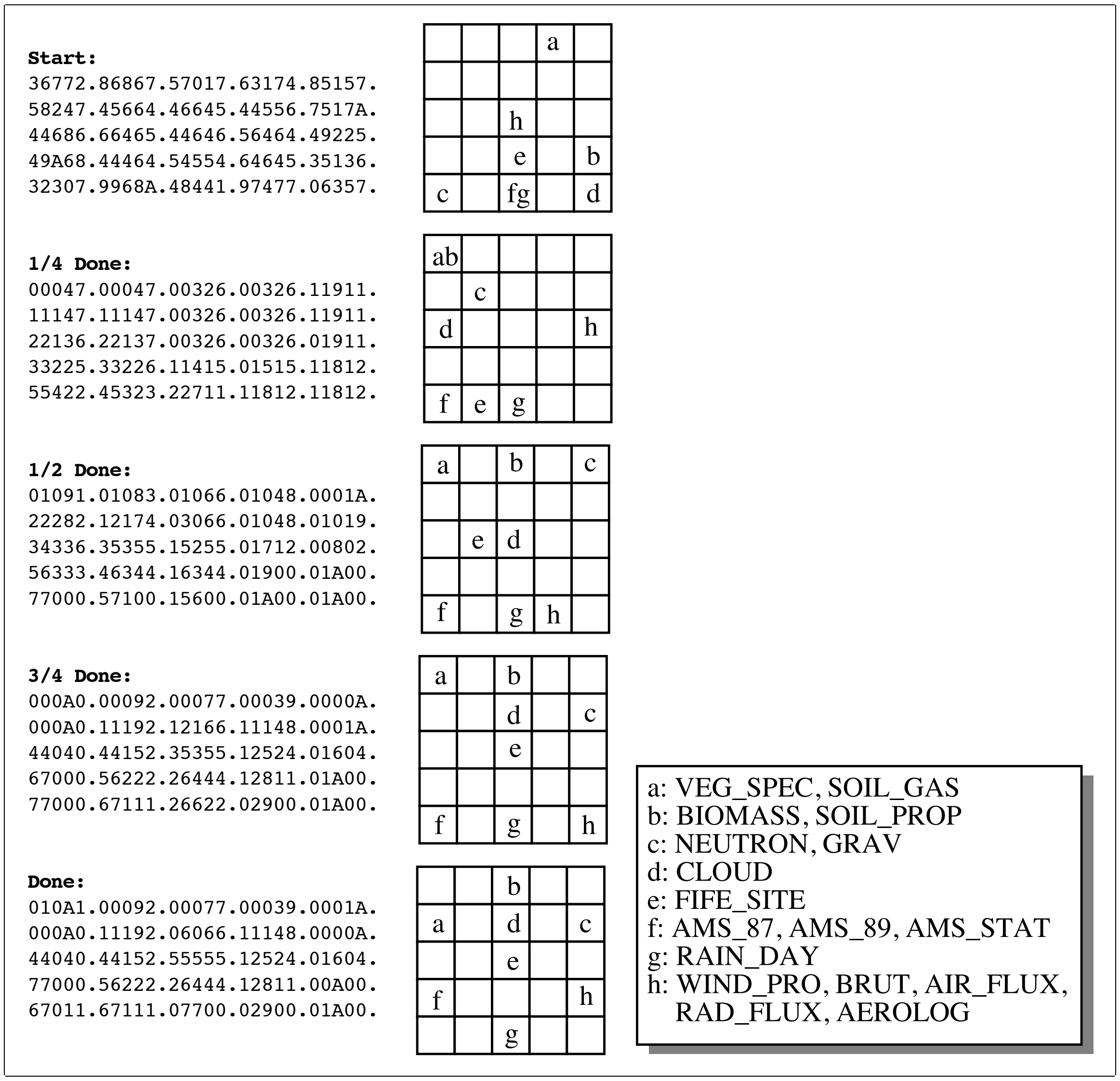


Figure 3.5: Neural Network Generating Clusters



Start:
```
36772.86867.57017.63174.85157.
58247.45664.46645.44556.7517A.
44686.66465.44646.56464.49225.
49A68.44464.54554.64645.35136.
32307.9968A.48441.97477.06357.
```

			a	
		h		
		e		b
c		fg		d

1/4 Done:
```
00047.00047.00326.00326.11911.
11147.11147.00326.00326.11911.
22136.22137.00326.00326.01911.
33225.33226.11415.01515.11812.
55422.45323.22711.11812.11812.
```

ab				
	c			
d				h
f	e	g		

1/2 Done:
```
01091.01083.01066.01048.0001A.
22282.12174.03066.01048.01019.
34336.35355.15255.01712.00802.
56333.46344.16344.01900.01A00.
77000.57100.15600.01A00.01A00.
```

a		b		c
	e	d		
f		g	h	

3/4 Done:
```
000A0.00092.00077.00039.0000A.
000A0.11192.12166.11148.0001A.
44040.44152.35355.12524.01604.
67000.56222.26444.12811.01A00.
77000.67111.26622.02900.01A00.
```

a		b		
		d		c
		e		
f		g		h

Done:
```
010A1.00092.00077.00039.0001A.
000A0.11192.06066.11148.0000A.
44040.44152.55555.12524.01604.
77000.56222.26444.12811.00A00.
67011.67111.07700.02900.01A00.
```

		b		
a		d		c
		e		
f				h
		g		

a: VEG_SPEC, SOIL_GAS
b: BIOMASS, SOIL_PROP
c: NEUTRON, GRAV
d: CLOUD
e: FIFE_SITE
f: AMS_87, AMS_89, AMS_STAT
g: RAIN_DAY
h: WIND_PRO, BRUT, AIR_FLUX,
RAD_FLUX, AEROLOG

Figure 3.6: Neural Network Generating Clusters (another view)

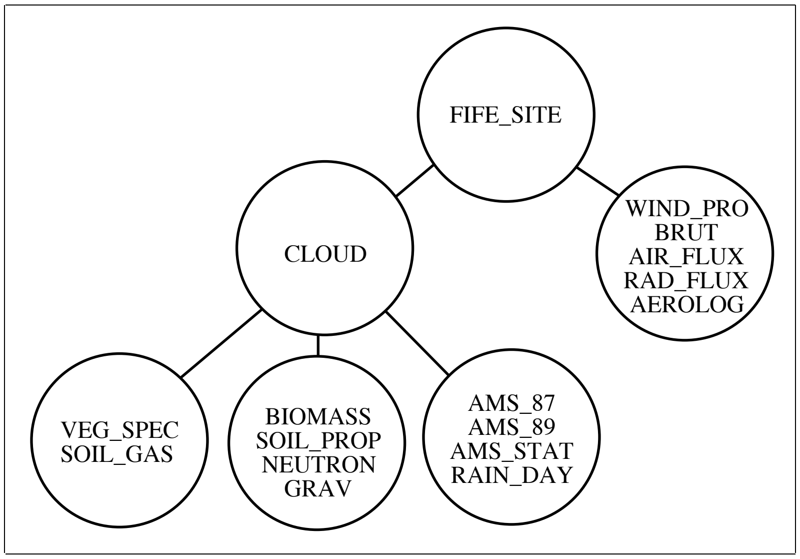


Figure 3.7: Clusters Produced by the Genetic Algorithm

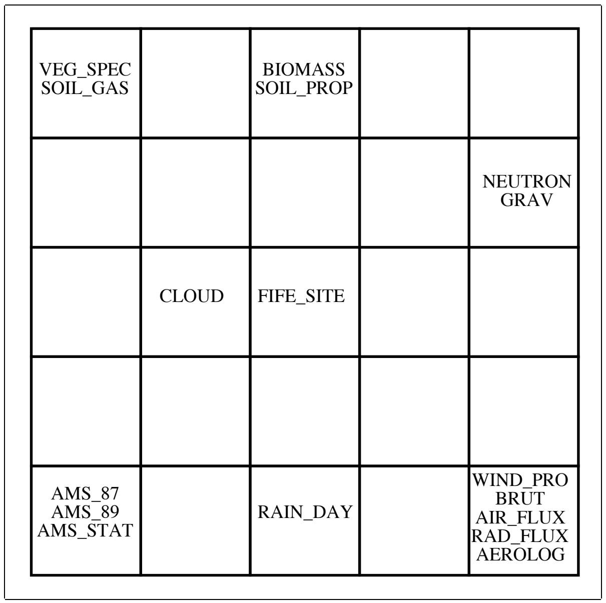


Figure 3.8: Clusters Produced by the Neural Network

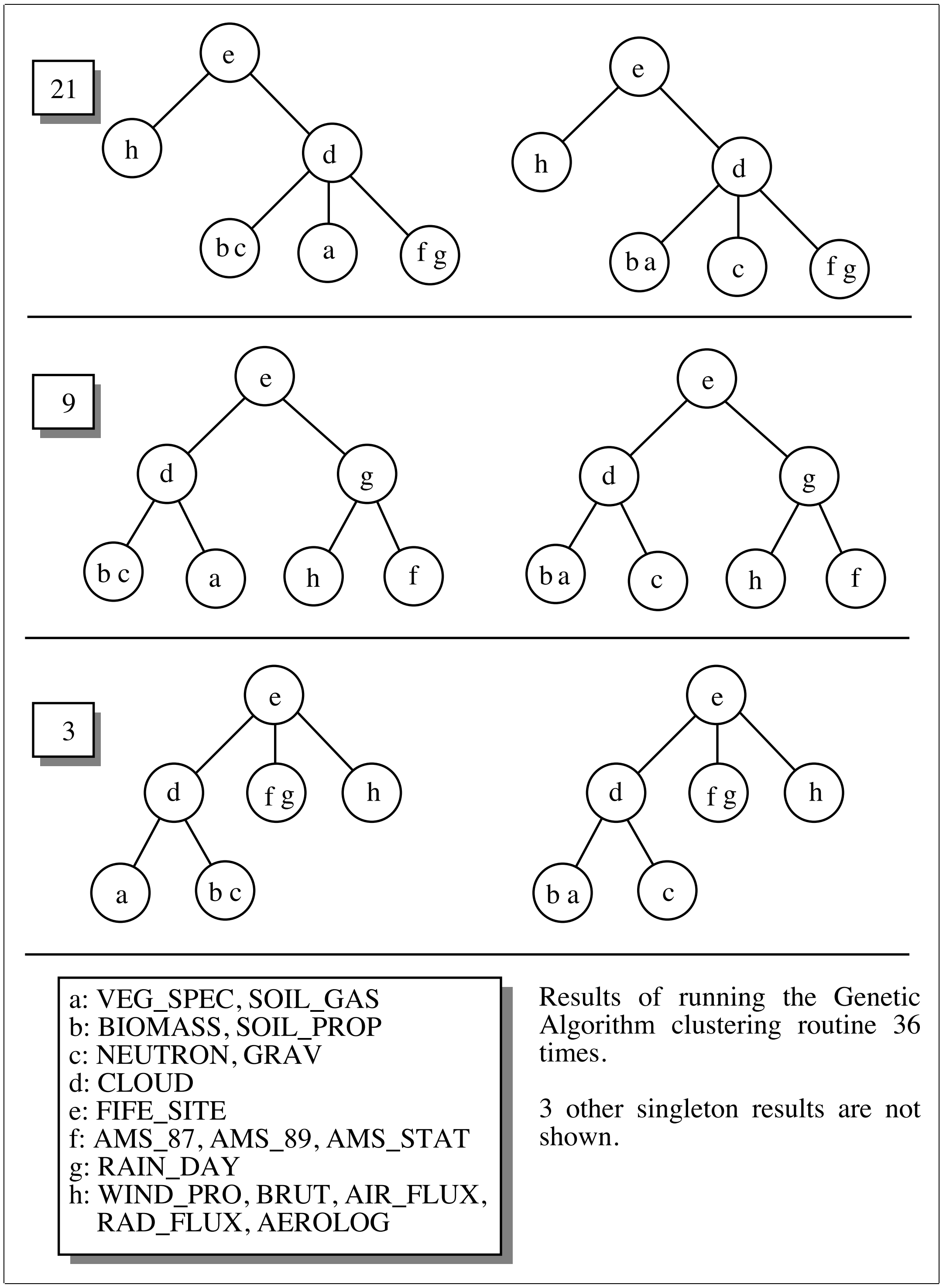
3.4.1 Clustering Stability

Multiple runs were made using each algorithm on the same input data to see how stable the clustering is. The results from 36 runs of the genetic algorithm are shown in Figure 3.9. The results of 60 runs of the neural network are shown in Figure 3.10.

In the genetic algorithm clustering the most common hierarchy of clusters was generated 21/36 times, the second most common 9/36 times, the third most common 3/36 times, and the remaining 3 hierarchies each once. Even among the three major hierarchy types there is a great deal of similarity. The FIFE_SITE table was common to all 5 queries, and a neighbour to all 16 other tables so it is always found at the top of the hierarchy. The CLOUD table was common to 3 queries, and a neighbour to 6 other tables so it is always found at an internal node near the top of the tree.

The final two queries (Surface Biophysical and Soil Moisture) each have 6 tables, 4 tables in common to both queries (FIFE_SITE, BIOMASS, CLOUD, SOIL_PROP) and 2 unique ones (SOIL_GAS, VEG_SPEC and GRAV, NEUTRON respectively.) None of these 4 unique tables are used in any of the other queries, making the two sets of unique tables equivalent with respect to the clustering algorithms. This can be seen in the clustering results where table set 'b' (BIOMASS, SOIL_PROP) was combined with table set 'a' (VEG_SPEC, SOIL_GAS) roughly half of the time and with table set 'c' (NEUTRON, GRAV) roughly half of the time.

In the neural network clustering the most common array of clusters was generated 46/60 times, the second most common 7/60 times, the third most common 3/60 times, and the remaining 4 arrays each once. There is a much wider variety among the actual arrays generated, as there is no specific orientation for the arrays. Rotating, flipping horizontally, or flipping vertically maps several different actual arrays into a single array format. The individual clusters within the array may also shift one array

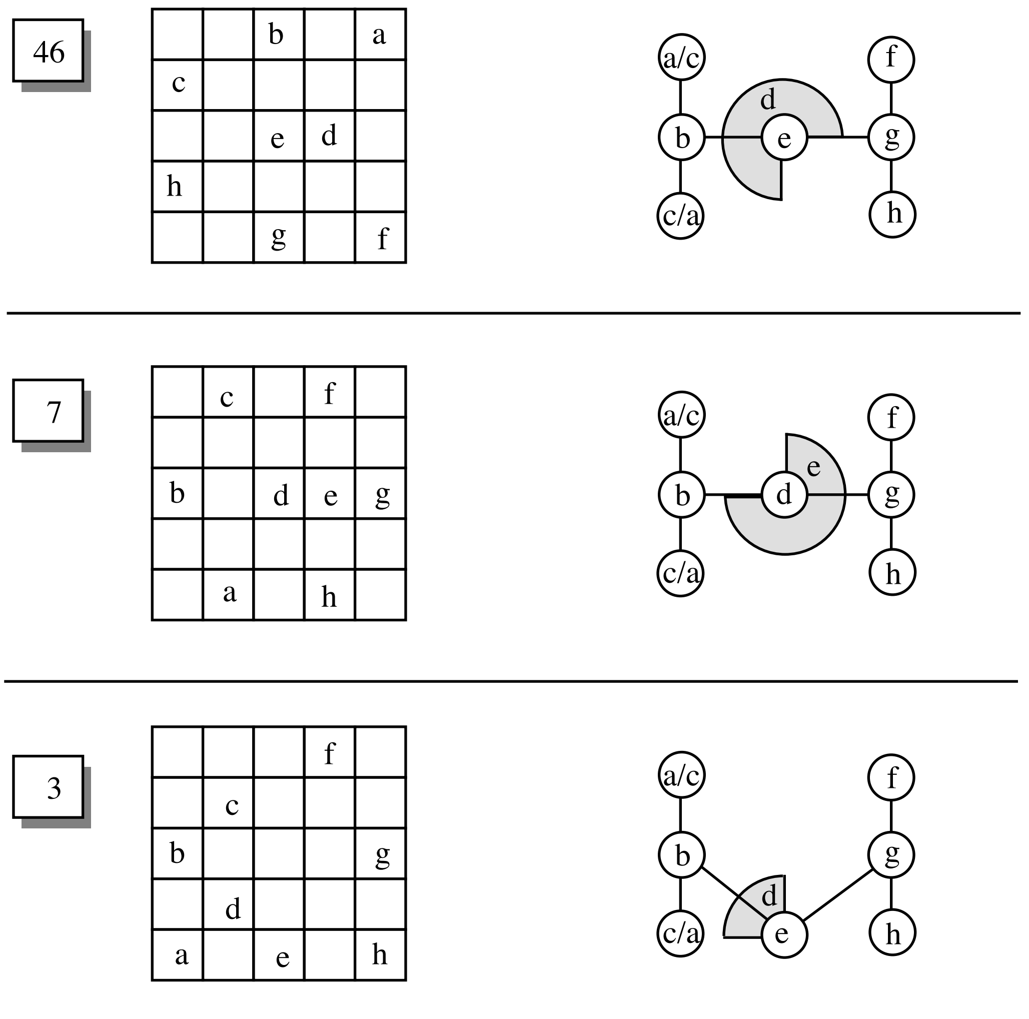


a: VEG_SPEC, SOIL_GAS
b: BIOMASS, SOIL_PROP
c: NEUTRON, GRAV
d: CLOUD
e: FIFE_SITE
f: AMS_87, AMS_89, AMS_STAT
g: RAIN_DAY
h: WIND_PRO, BRUT, AIR_FLUX, RAD_FLUX, AEROLOG

Results of running the Genetic Algorithm clustering routine 36 times.

3 other singleton results are not shown.

Figure 3.9: Multiple Runs of the Genetic Algorithm



a: VEG_SPEC, SOIL_GAS
b: BIOMASS, SOIL_PROP
c: NEUTRON, GRAV
d: CLOUD
e: FIFE_SITE
f: AMS_87, AMS_89, AMS_STAT
g: RAIN_DAY
h: WIND_PRO, BRUT, AIR_FLUX, RAD_FLUX, AEROLOG

Results of running the Neural Network clustering routine 60 times.

4 other singleton results are not shown.

Figure 3.10: Multiple Runs of the Neural Network

position between different runs. Factoring out these differences leads to the three overall most common configurations of clusters generated.

Figure 3.10 shows both an actual sample clustering, and the general array format for the three most common array formats. In the most common format table set ‘e’ was in the center, with table set ‘d’ orbiting about it. Table sets ‘b’ and ‘g’ would be on opposite sides of table set ‘e’. Table sets ‘a’ and ‘c’ are on opposite sides of table set ‘b’ and table sets ‘f’ and ‘h’ are on opposite sides of table set ‘g’. The second most common format was similar to the most common format except table set ‘d’ replaced table set ‘e’ in the center of the array and table set ‘e’ orbited about ‘d’. In the third most common clustering table set ‘e’ was at the center of one of the edges of the array, rather than at the center of the array.

3.4.2 Comparison of Clusters Generated

The genetic algorithm generates a strict hierarchy with an arbitrary number of levels where the relationships between the clusters are shown by their relative positions in the hierarchy. The neural network generates only two levels of clustering but the overview shows more subtle relationships between the clusters. The genetic algorithm generates deeper, more structured indices than the neural network. The neural network gives a better overall picture of how clusters are related. Figure 3.11 shows how the some possible clusterings of tables are represented by the neural network and the genetic algorithm.

The neural network clustering can help a new user see a global picture of the database. With the neural network clustering, tables used most generally will be found near the center of the array. Tables with more specific uses will be found towards the corners. This can be seen in Figure 3.8 where the FIFE_SITE table used

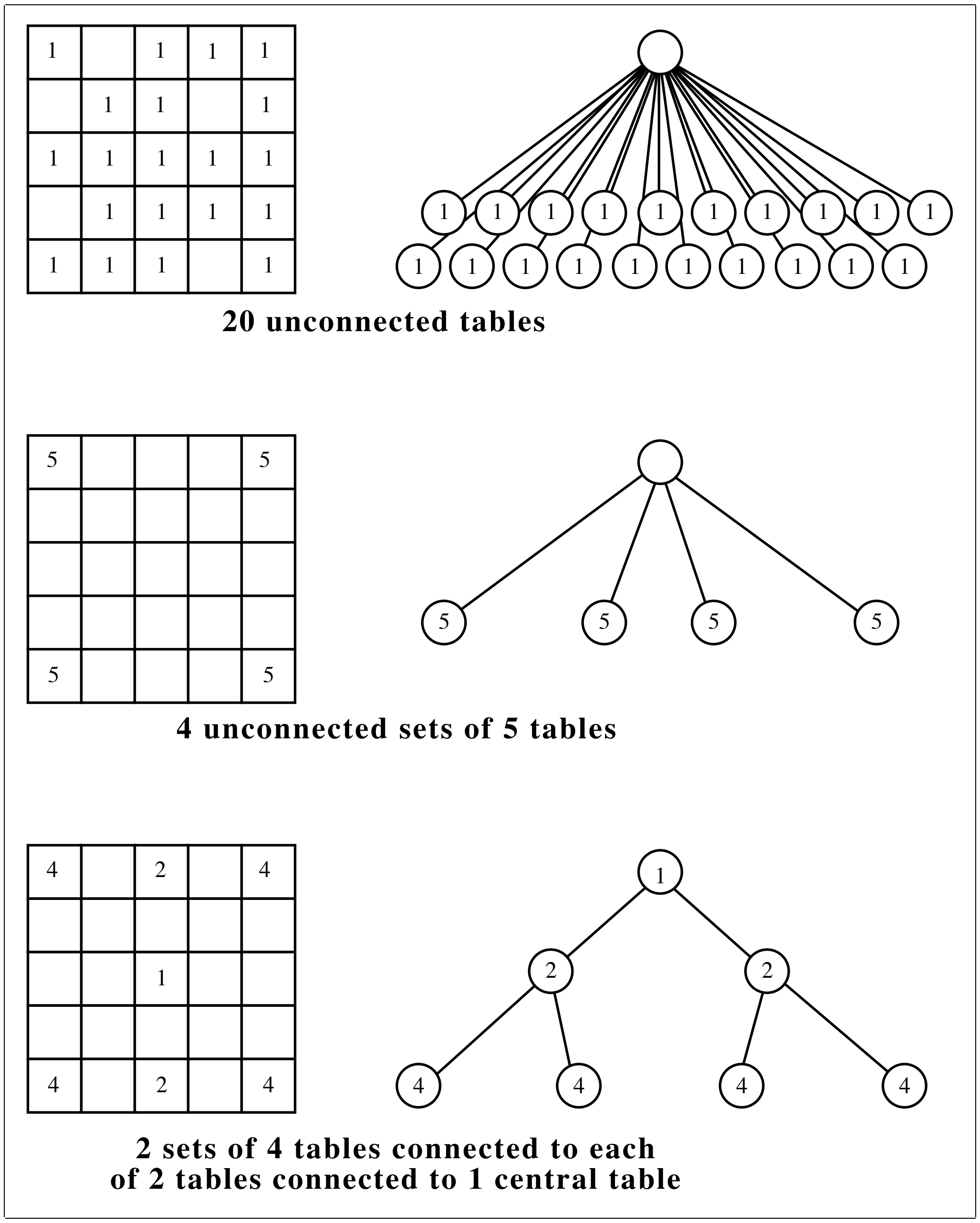

20 unconnected tables
4 unconnected sets of 5 tables
2 sets of 4 tables connected to each
of 2 tables connected to 1 central table

Figure 3.11: Common Clusters in Both Implementations

in all five queries was placed in the center of the array while the VEG_SPEC table used in only one query was placed in the upper left corner.

The genetic algorithm clustering can help an administrator categorize the clusters into a hierarchical relationship. General concepts are found in the interior nodes at the top of the hierarchy while more specific concepts are found at the bottom of the hierarchy, near the actual tables being clustered. This generates a hierarchy similar to the generic hierarchy provided by most statistical and scientific databases, but this one is based on actual user patterns instead of presumed user patterns.

The strict partitioning of the genetic algorithm, like a generic hierarchy, will not completely satisfy each user. There is no half-way point between clusters. The neural network clustering does provide these half-way clusters by generating more, smaller, clusters and using their positions in the matrix to show which clusters are 'close.'

3.4.3 A Larger Example

Using all 106 tables in the FIFE database, and the hierarchy in the existing generic FIFE menu interface we can divide the tables into 8 lists. The generic hierarchy was divided into 8 sections and each of these sections is given its own list. Some tables were found in multiple sections of the hierarchy generating some overlap. All tables in the same section are considered neighbours for this test.

The percentages used are shown in Figure 3.13. Each table in the database is given a number (1-106). The usage percentages are determined by the number of times each table is found in each section of the hierarchy.

Markers are used to make the clusters generated more visible. Table 3.6 shows the labels used in the later figures. Two tables are given the same marker if they have the same neighbours and the same usage percentages.

Marker	Table
a	1
b	2
c	3,6,28,29,32,43
d	4
e	5
f	7
g	8,9,10,15,17,24,33,34,35,37, 40,41,47,48,49,50,51,52,53, 54,55,68,69,71,78,80,82,84, 85,86,93,94,98,99,100,102,103
h	11
i	13,38,74,92,106
j	14
k	16
l	19
m	20
n	21,22,23
o	25
p	12,18,26,36,39,42,45,46, 58,59,60,63,95,96,97
q	30,31,65,76,77
r	44,83
s	56,57
t	61,62
u	66,67
v	70,72
w	75
x	79
y	81
z	27,64,73,87,88,89,90
+	91
-	101
/	104,105

Table 3.6: Labels for Tables in Larger Example

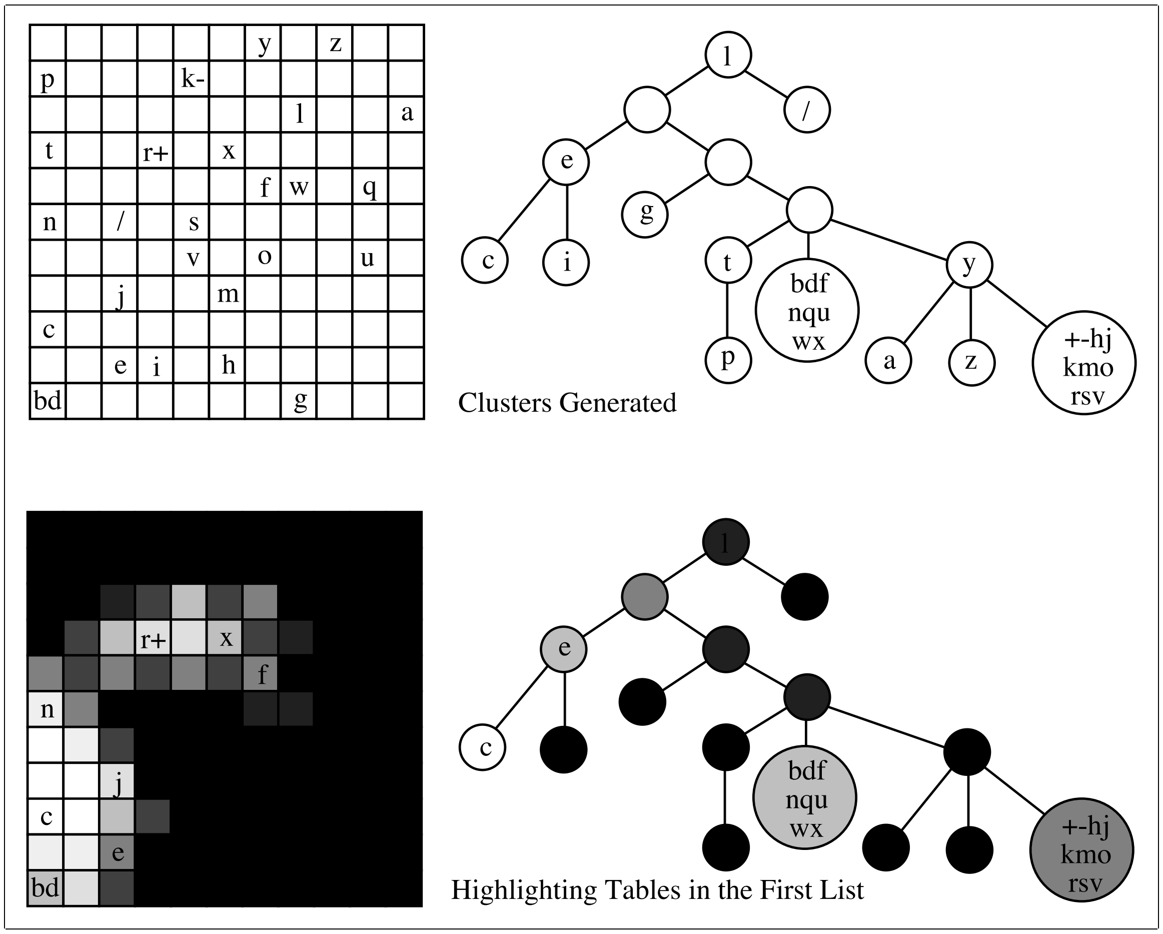


Figure 3.12: Results of the Larger Example

Figure 3.12 shows the results of the clustering and how the clusters can be highlighted. As in the smaller example, both the genetic algorithm and neural network generate similar sets of clusters. In the neural network tables with the same usage percentages are automatically found at the same node in the array, but the genetic algorithm has also been able to create these sets of tables (in the case of set g, by combining 32 tables, in the case of set p, by combining 15 tables.)

As can be seen from Figure 3.13 there is a large amount of overlap between the lists making it very difficult to generate a hierarchy. Each of the sets of tables unique to a list (c, g, i, p, q, z, /) were given their own leaf node by the genetic algorithm, and can be found near the edges of the neural network. Table sets c, e, i are near each other in both the genetic algorithm hierarchy and the neural network array. This is similarly true for table sets a, y, z. The tables which are found in multiple lists are generally found to the center of the array are are combined into two clusters by the genetic algorithm.

3.4.4 Comparison of Execution Times

The two algorithms have comparable execution times. Compared to a deterministic algorithm clustering the tables using an identical payoff function, the genetic algorithm, on average, found a clustering better than 99% of the possible clusterings with inputs of less than 14 tables. It was impractical to do comparisons with more than 13 input tables due to the running time of the deterministic program. Thus the heuristically based genetic algorithm does the same job as a deterministic algorithm in a small fraction of the time. For a comparison of running times of the deterministic, genetic algorithm, and neural network clustering algorithms see Figure 3.14. The time for running the neural network is directly related to the size of the array. As the array size is based on the $\sqrt{number\ of\ tables}$, the time complexity of the neural network is a step function.

3.4.5 Clustering as an Ongoing Process

Clustering will be an ongoing process in the database. New users will begin using the database, current users will return to search for new data. Perhaps even new tables will be added to the database. More and more relationships will connect the tables together, and more tables will be brought into the clustering. These new relationships may be very different from the older relationships. Over time the data in the database will be used for different purposes, so the clustering must adapt. The

#	ADA	SFS	ATM	SUR	CCD	SRB	SLM	IGS	Table Name
1.	0.0	0.0	0.0	8.6	0.0	0.0	1.4	0.0	AEROLOGICAL_STATE
2.	3.8	0.0	3.8	0.0	0.0	1.3	1.3	0.0	AIRCRAFT_DOC
3.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	AIRCRAFT_EXTRACT
4.	1.7	0.0	5.0	0.0	0.0	1.7	1.7	0.0	AIRCRAFT_FILE_INV
5.	2.5	0.0	7.5	0.0	0.0	0.0	0.0	0.0	AIRCRAFT_FLUX_DATA
6.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	AIRCRAFT_FLUX_INV
7.	3.3	0.0	0.0	0.0	0.0	3.3	3.3	0.0	AIRCRAFT_IMAGE_INV
8.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	AMS_DATA_87
9.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	AMS_DATA_89
10.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	AMS_INV
11.	0.0	4.0	6.0	0.0	0.0	0.0	0.0	0.0	ATMOS_COND_INV
12.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	BIOMASS DATA
13.	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	BRUT_SONDE_DATA
14.	2.5	2.5	0.0	0.0	5.0	0.0	0.0	0.0	C130_OPTICAL_DATA
15.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	CAL_MMR_DAT
16.	0.0	5.0	0.0	0.0	0.0	5.0	0.0	0.0	CAL_SE590_DATA
17.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	CLOUD_CAMERA_DATA
18.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	EXOTECH_TRANSECT_DATA
19.	0.3	2.9	0.9	1.8	0.9	1.8	1.5	0.0	FIFE_SITE_REF
20.	0.0	2.5	0.0	0.0	5.0	0.0	0.0	2.5	FRASER_OPT_DATA
21.	5.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	GAMMA_RAY_DATA
22.	5.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	GAMMA_RAY_GROUND_DATA
23.	5.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	GAMMA_RAY_SITE_REF
24.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	GIS_INV
25.	0.0	5.0	0.0	0.0	0.0	0.0	0.0	5.0	GRAVIMETRIC_DATA
26.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	GROUND_SE590_DATA
27.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	HEAT_MASS_FLUX
28.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	HELO_GEMMA_DATA
29.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	HELO_IRT_DATA
30.	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	HELO_SCATT_DATA
31.	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	HELO_SCATT_INV
32.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	HELO_SE590_DATA
33.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	KSU_EXOTECH_DATA
34.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	KSU_LIGHT_BAR_DATA
35.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	KSU_LIGHT_WAND_DATA
36.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	KSU_MOW_DATA
37.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	LEAF_ANGLE_DATA
38.	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	LIDAR_HEIGHT_DATA
39.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	LIGHT_BAR_DATA
40.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	MANHATTAN_AVG_DATA
41.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	MANHATTAN_DATA
42.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	MMR_GROUND_DATA
43.	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	MMR_HELO_DATA
44.	3.3	3.3	0.0	0.0	0.0	3.3	0.0	0.0	MMR_INV
45.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	MMR_LEAF_DATA
46.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	MOW_EXOTECH_DATA
47.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NEUTRON_PROBE_DATA
48.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NMCUPPERAIR_DATA
49.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NMCUPPERAIR_INV
50.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAAMET_STATION_REF
51.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAARADIO_DATA
52.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAARADIO_INV
53.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAASURFACE_DATA
54.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAASURFACE_DAY_INFO
55.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	NOAASURFACE_STATION_INFO
56.	0.0	3.3	0.0	0.0	6.7	0.0	0.0	0.0	OPTICAL_THICK_DATA
57.	0.0	3.3	0.0	0.0	6.7	0.0	0.0	0.0	OPTICAL_THICK_INV
58.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	PARABOLA_DATA
59.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	PARABOLA_INV
60.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	PHOTOSYN_BOX_DATA
61.	0.0	0.0	0.0	0.0	0.0	6.7	0.0	3.3	PHOTOSYN_INV
62.	0.0	0.0	0.0	0.0	0.0	5.0	0.0	5.0	PHOTOSYN_LEAF_DATA
63.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	PLANT_PRODUCTION_DATA
64.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	RADIATION_FLUX
65.	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	RAIN_CMILLY_DATA
66.	0.0	5.0	0.0	0.0	0.0	0.0	5.0	0.0	RAIN_DAILY_DATA
67.	0.0	5.0	0.0	0.0	0.0	0.0	5.0	0.0	RAIN_INV
68.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SATELLITE_EXTRACT
69.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SATELLITE_EXTRACT_COEFF
70.	0.0	5.0	0.0	0.0	5.0	0.0	0.0	0.0	SATELLITE_FILE_INV
71.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SATELLITE_INFO
72.	0.0	5.0	0.0	0.0	5.0	0.0	0.0	0.0	SATELLITE_INV
73.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	SIX_MIN_SURFACE_FLUX
74.	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	SODAR_DATA
75.	0.0	0.0	0.0	0.0	0.0	5.0	5.0	0.0	SOIL_GAS_FLUX_DATA
76.	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	SOIL_IMPEDANCE_DATA
77.	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	SOIL_MOISTURE_TRANSECT
78.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SOIL_PROPERTIES_DATA
79.	1.4	1.4	0.0	0.0	0.0	1.4	5.7	0.0	SOIL_PROPERTIES_INV
80.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SOIL_SITE_REF
81.	0.0	1.4	0.0	8.6	0.0	0.0	0.0	0.0	SOIL_STATE
82.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	SOIL_SURVEY_REF
83.	3.3	3.3	0.0	0.0	0.0	3.3	0.0	0.0	SPECTRO_INV
84.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	STORMFLOW_DATA
85.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	STREAMFLOW_DATA
86.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	STREAMFLOW_INV
87.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	SURFACE_FLUX_AVERAGES
88.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	SURFACE_FLUX_DAILY_TOTALS
89.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	SURFACE_FLUX_INV
90.	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	SURFACE_FLUX_VARIABLES
91.	2.5	2.5	0.0	0.0	0.0	5.0	0.0	0.0	SURF_RAD_INV
92.	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	TEMP_PROFILE_DATA
93.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	TOVS_DATA
94.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	TOVS_SUM
95.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	UNL_EXOTECH_DATA
96.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	UNL_LONGWAVE_DATA
97.	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	UNL_SURF_DATA
98.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	USGS_15MIN_DATA
99.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	VEG_BIOP_SUM
100.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	VEG_BIOP_WTS
101.	0.0	3.3	0.0	0.0	0.0	6.7	0.0	0.0	VEG_INV
102.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	VEG_SPECIES_DATA
103.	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	VEG_SPECIES_REF
104.	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	WATER_TEMP_DATA
105.	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0	WATER_TEMP_INV
106.	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	WIND_PROFILE_DATA

Figure 3.13: Percentages Used in the Larger Example

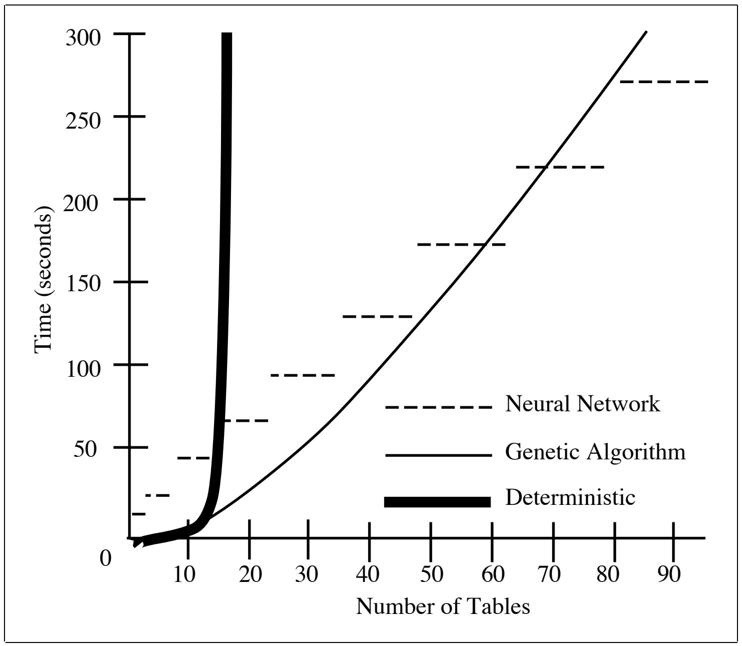


Figure 3.14: Comparison of Different Clustering Algorithms

clustering should dynamically support both the new and current users.

Totally re-clustering the tables will allow the index to show the most effective clustering based on all of the users' previous experiences. This will be very useful to new users, but may be confusing to current users when the clustering patterns change, possibly dramatically. Instead, the clustering can be performed incrementally. The current clusters can be used as an input to both the genetic algorithm and the neural network, instead of starting the clusters from scratch. This will allow the clusters to change slowly, giving the current users a common framework and adding in each new user's contributions.

More than one set of indices can be maintained in the system. New users can start with the most up-to-date indexing information, while current users can access the database using their familiar clusters. Since the clusters are derived from the input lists, different combinations of input lists will yield different orientations. Individuals, groups, and sites can maintain their own clusters as well as the global clustering for the entire database. The global clustering can be updated at regular intervals. Individuals, groups, and sites can choose how often to update their clusters. They can also decide to include all the available lists no matter how old, or only cluster based on recent usage patterns. However the clusters are maintained, the underlying database remains untouched.

3.5 Clustering Uses

The clusters that have been generated can be used for several purposes. They can aid both the user and the database administrator.

When users wish to start a new query, they can look through the existing set of topics that have been queried. One of these topics may match up with their needs giving them an excellent starting point for their queries. They may find that their area of interest straddles two existing queries. Using the existing lists the user can reduce their search time in the database as shown in Figures 3.15 and 3.16.

Users of the existing FIFE database interface must either directly find the necessary tables from among the 106 tables in the database, or use the existing menu system. This static menu system has 60 nodes in the hierarchy in addition to the 106 tables at the leaves. Users of the existing menu system need to know not only the topic of the table they are interested in, but also the name of the investigator who collected the original data, to traverse the hierarchy effectively.

If the user of the existing interface wishes to go back to that table in the future then the user must either remember the name of the table itself (was it SOIL-IMPEDANCE-DATA or SOIL-PROPERTIES-DATA?) or retrace their steps through the menus. The user is responsible for remembering these details. The only way a user can convey their experiences to another user is to write down an itinerary of

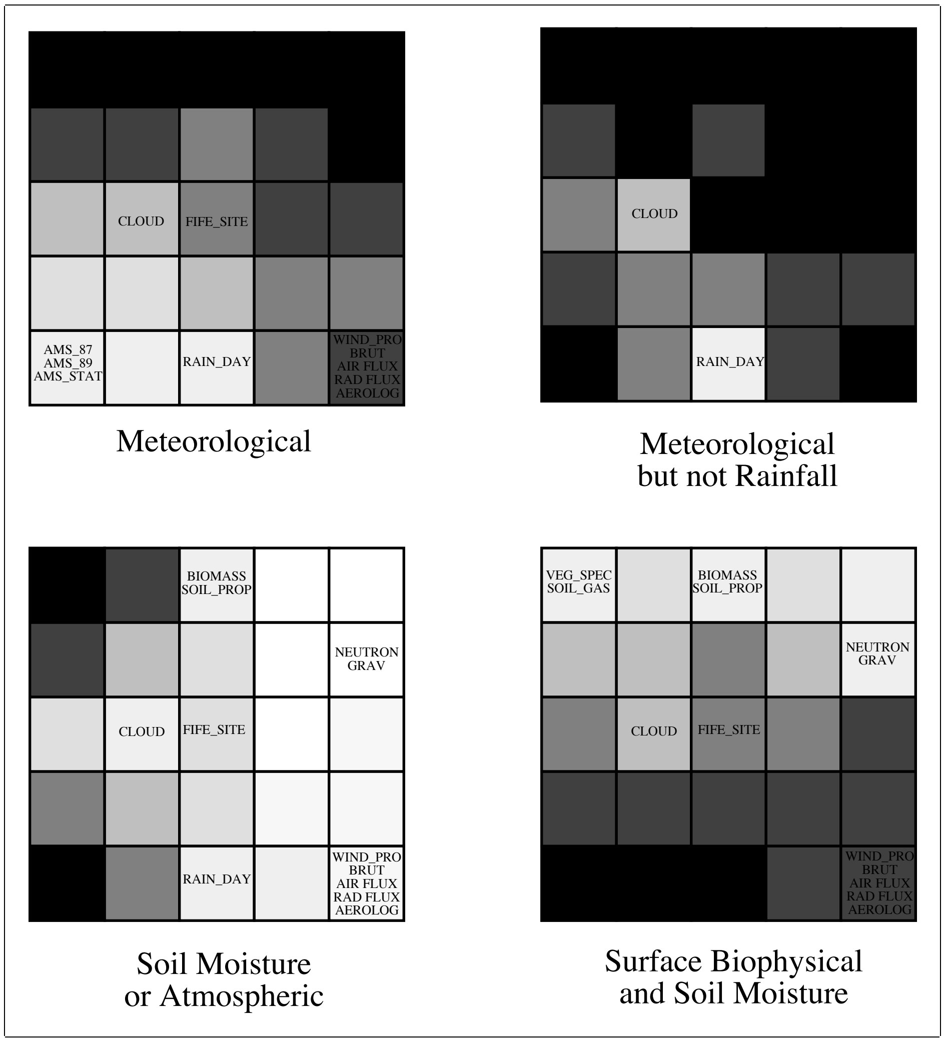


Figure 3.15: Sample Index Queries Using the Neural Network

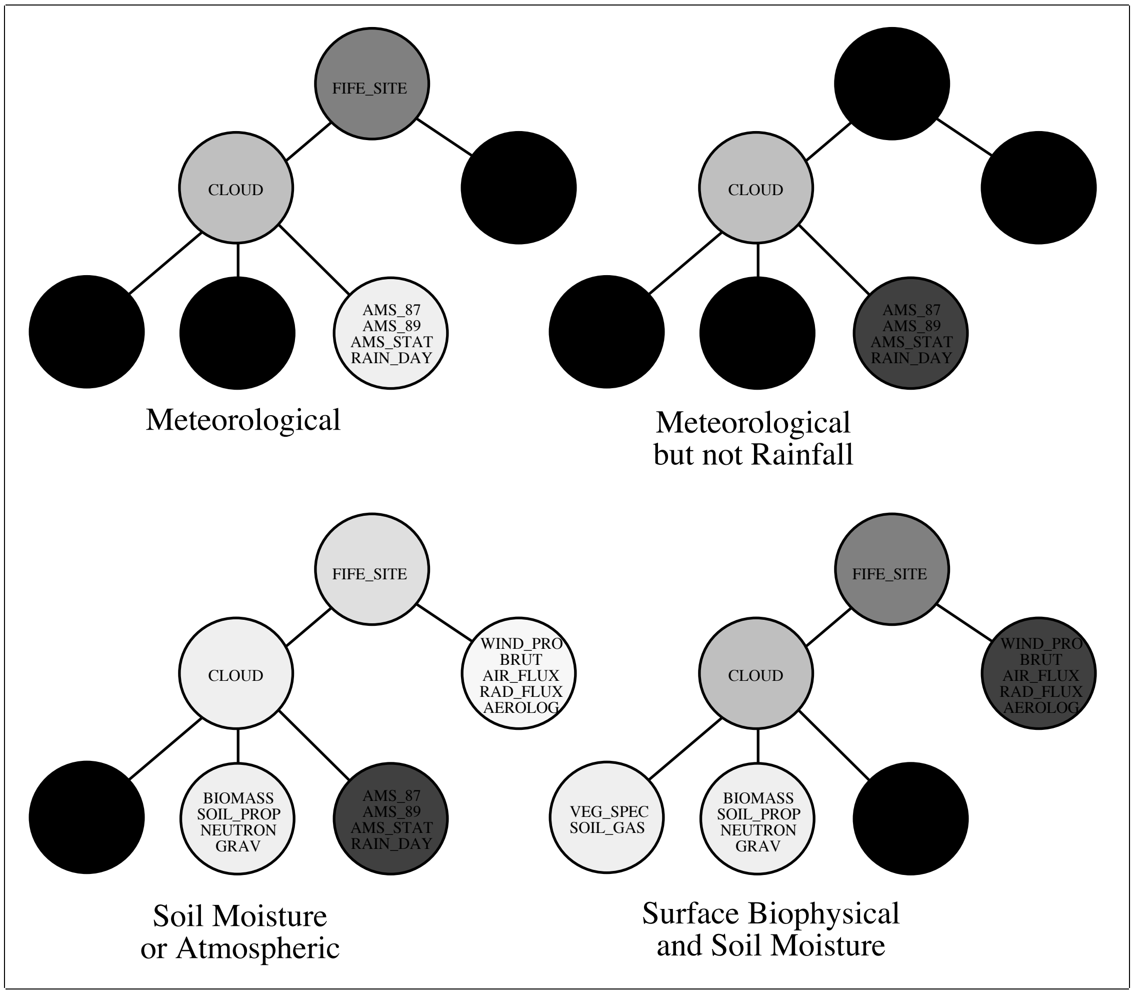


Figure 3.16: Sample Index Queries Using the Genetic Algorithm

their travels through the database for the other user to follow.

These clusters act like a filter, highlighting where the user has found valuable information, or where users in similar fields have found valuable information. This clustering can be used as an additional layer on top of the scientific database. The individual tables in the clusters could be linked to tables in the database itself allowing the user to browse through the clusters using a graphical interface, and drop into the database itself at the appropriate table.

3.5.1 Highlighting Clusters

Figure 3.15 shows how the clusters in Figure 3.8 can be highlighted. Each of the arrays in Figure 3.15 is in the same orientation as the larger (and more readable) Figure 3.8. The user can choose to see the tables important to someone interested in Meteorological information, or someone interested in Meteorological data but not Rainfall data. The user might want information related to Soil Moisture or Atmospheric conditions, or they may want information that combines Surface Biophysical data and Soil Moisture data.

By selecting the appropriate queries the clusters highlight the appropriate areas of the database. Important areas are shown in white; unimportant areas are shown in black. The brightness of the region shows how relevant the tables in that region are to the suggested query areas. Since the neural network forms a topological map of the input lists, these input lists allow us to find the literal peaks and valleys of interest.

Figure 3.16 shows how the clusters in Figure 3.7 can be highlighted. As with the neural network version, the user can choose combinations of the existing queries to see the tables that are of personal importance.

3.5.2 Nearby Clusters

As well as highlighting the important tables, the clustering diagram also shows relationships between the highlighted tables, and how general or specific the highlighted tables are by their location in the array. Thus it gives much more information to the user than simply highlighting table names based on the usage percentages directly, or using a general overview diagram of the database.

It is unlikely that any of the existing topics will exactly match the needs of a new user so the ability to find 'nearby' tables is very important. This can only be provided by integrating the experiences of many users. This information can be especially important if the user needs to find intermediate tables to link together the values from tables of interest. The ability to see 'nearby' tables is also important if the user has an inexact query. The existing topic lists can give the user a good starting point to begin browsing through the 'nearby' tables to see if they are of interest.

3.5.3 Zooming through the Clusters

Since there are complete weight vector values for every element in the neural network array, the computer can interpolate these values to change the size of the array without recomputing the weight vectors. This allows the user to 'zoom in' to the array to see an area in more detail, or 'zoom out' to see less detail. As the user 'zooms out' the individual clusters form into clusters of clusters; as the user 'zooms in' the clusters break apart. Figure 3.17 shows what the array looks like while zooming into and out of the meteorological index.

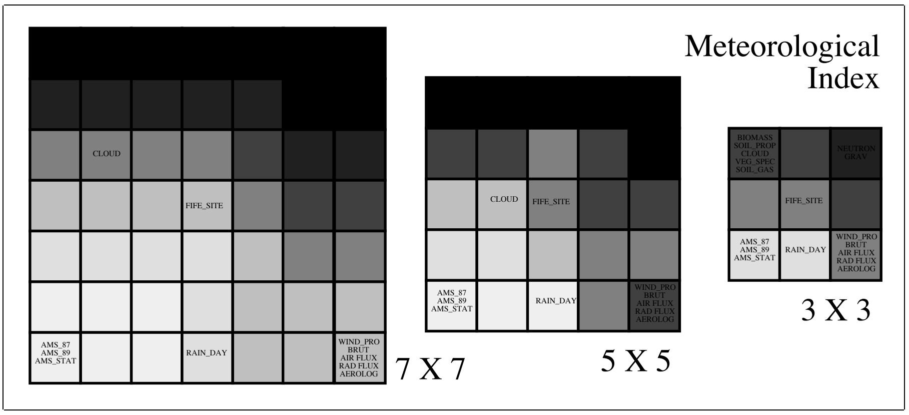


Figure 3.17: Zooming In and Out of the Array

3.5.4 Highlighting the Clusters in 3D

We can use 3D computer graphics to improve our visualization of the clusters. Figure 3.18 shows the meteorological index from Figures 3.15 displayed using colour 3D graphics instead of greyscale 2D graphics. The on-screen version uses colour as well as height to distinguish the appropriateness of the clusters. As the altitude increases the colour changes from blue to brown to green to yellow to white simulating a landscape from the blue of the ocean to the white of the mountain peaks. The higher the cluster the more appropriate to the topic, the lower the cluster, the less appropriate to the topic. The 3D environment also gives the user the ability to rotate the clusters. Looking down on the clusters from directly above would give the user the same view as the 2D representation of the clusters shown earlier.

3.5.5 Partitioning the Database

Using the clustering information provided by the users, the database administrator can find which tables are in common usage among which groups. For example, all the investigators would require access to the most commonly used tables but perhaps only the geologists are accessing certain tables, and the climatologists are the exclusive

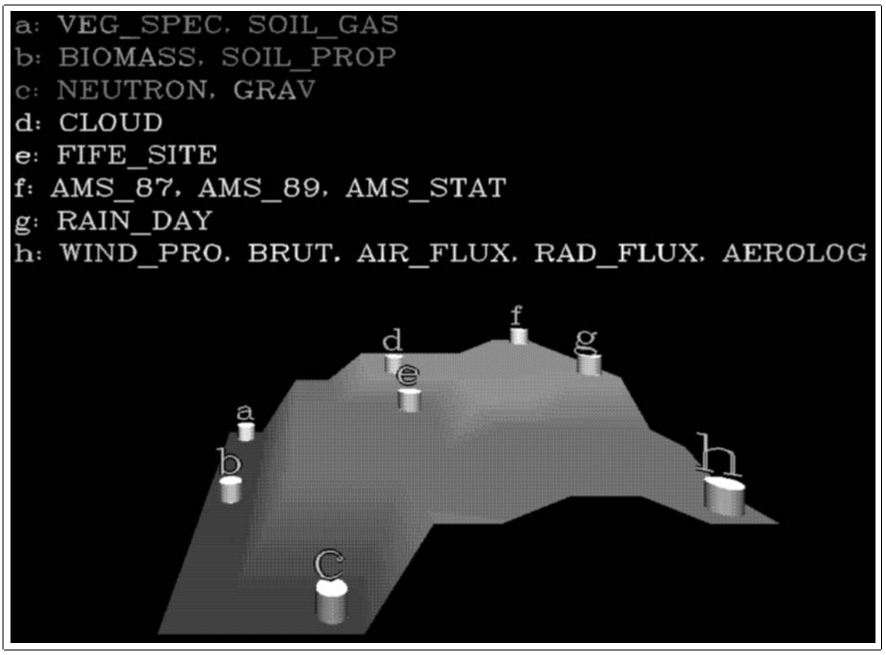


Figure 3.18: Visualizing the Clusters Using 3D Graphics

users of another set of tables. Using the clusters the administrator can determine which tables an investigator will need based on their areas of interest, because those are the tables that investigators with similar interests have used in the past.

3.6 Parallelism

I implemented two parallel versions of the neural network clustering algorithm, using PVM [BDG+91] and Concert/C [AKRY91, Gol93] to see whether parallel processing could significantly reduce the clustering time.

3.6.1 Parallel Algorithms

The algorithm for the uniprocessor version of an M by M self organizing feature map is shown in Figure 3.19 and its data structures shown in Figure 3.20.

The algorithm for the parallel version of an M by M self organizing feature map is shown in Figure 3.21 and its data structures shown in Figure 3.22. In the parallel

```
for row = 1 to M
    for col = 1 to M
        Net[row][col].weights = random values

Initialize $\alpha$, neighbourhood

while ($\alpha > 0$)
    Choose one input vector I at random

    BestMatch = null, null, null

    for row = 1 to M
        for col = 1 to M
        if Net[row][col].weights is a better match
            BestMatch = |I - Net[row][col].weights|, row, col

    for row' = (row - neighbourhood) to (row + neighbourhood)
        for col' = (col - neighbourhood) to (col + neighbourhood)
            Adjust Net[row'][col'].weights

    Adjust $\alpha$, neighbourhood

end while
```

Figure 3.19: Uniprocessor Algorithm

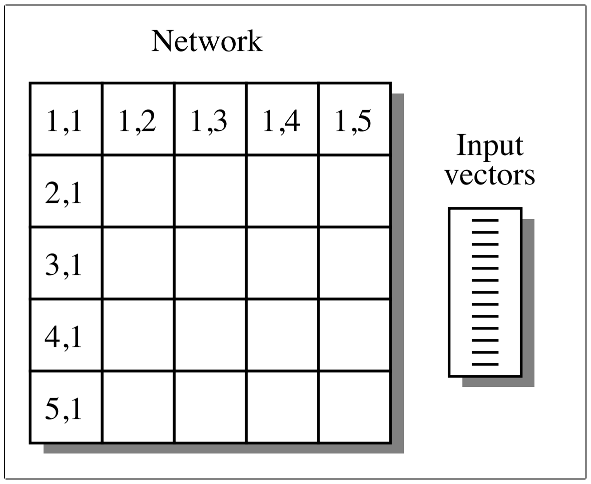


Figure 3.20: Data Structures for the Uniprocessor Version

version the M by M matrix is broken into M 1 by M matrices. Each 1 by M matrix and a complete set of the input vectors is given its own process.

In each iteration of the main loop in the parallel version, the controller sends the index of the chosen table's weight vector to each process. Each process sends back the 'goodness' of the best match in its row and the column of the best match. The controller then determines the overall best match and sends all appropriate processes (all those with an element in the affected neighbourhood) α, the neighbourhood, and the column of the best match. Each appropriate process then updates the appropriate values.

3.6.2 Parallel Results

I tested both parallel implementations using a network of ten sparcstation 2s connected by Ethernet. I used 53 lists accessing all 106 tables in the FIFE database. I created these lists by converting each submenu in the hierarchy of the existing FIFE menu system into a list. I made 7 runs for each value and averaged the middle 5. These runs were made late at night and in the early morning to decrease the likelihood of other users on the machines. I ran tests using 1, 2, 4 and 8 processors though the number of processors did not have to be a power of two. If there were more processes than processors, then the processes were distributed as evenly as possible over the available processors.

The results from running the PVM version are shown in Table 3.7 and graphically in Figure 3.23. The results from running the Concert/C version are shown in Table 3.8 and graphically in Figure 3.24.

The descriptions of the column headings for both the PVM and Concert/C results are given below:

```
**controller creates M processes**

**controller sends all input vectors to all M processes**

**in parallel in each process**
    for col = 1 to M
        $Net_p$[col].weights = random values

Initialize $\alpha$, neighbourhood

while ($\alpha > 0$)
    Choose one input vector I at random
    **Send the index of input vector I to each process**

    **in parallel in each process**
        BestMatch = null, null

        for col = 1 to M
            if $Net_p$[col].weights is a better match
                BestMatch = |I - $Net_p$[col].weights|, col

        **send BestMatch to the controller**

    **controller determines overallBestMatch at Net[row][column]**

    **controller sends $\alpha$, neighbourhood, column to processes**
    **(row - neighbourhood) to (row + neighbourhood)**

    **in parallel in processes (row-neighbourhood) to (row+neighbourhood)**
        for col = (column - neighbourhood) to (column + neighbourhood)
            Adjust $Net_p$[col].weights

    Adjust $\alpha$, neighbourhood

end while

**controller receives the row from each process**
```

Figure 3.21: Multi-Processor Algorithm

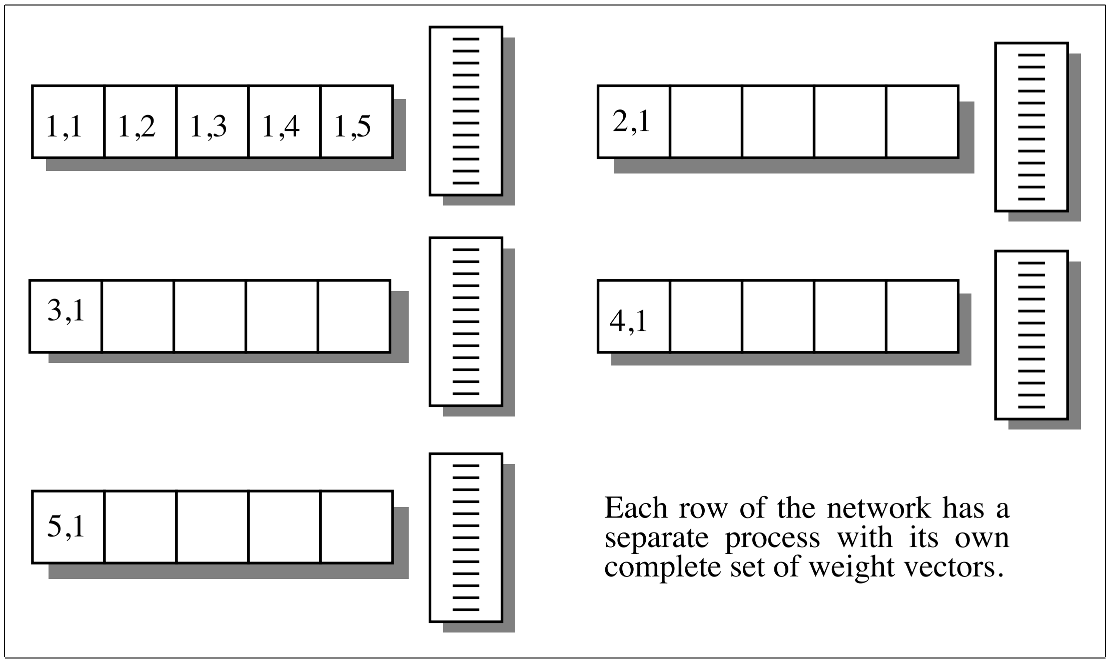

1,1
1,2
1,3
1,4
1,5
2,1
3,1
4,1
5,1
Each row of the network has a separate process with its own complete set of weight vectors.

Figure 3.22: Data Structures for the Multiprocessor Version

size - size of the network (m by m)

uni-1 - running times of the uniprocessor algorithm on 1 processor in secs.

par-1 - running times of the parallel algorithm on 1 processor in secs.

par-2 - running times of the parallel algorithm on 2 processors in secs.

par-4 - running times of the parallel algorithm on 4 processors in secs.

par-8 - running times of the parallel algorithm on 8 processors in secs.

Using both parallel programming environments there is continuous improvement as the number of processors is increased from 1 to 2 to 4 to 8. With PVM, even the parallel version run on one processor eventually outperforms the uniprocessor version (probably because memory is partitioned to make swapping easier in the parallel version.) The overhead due to communication costs between the various processors makes the uniprocessor version more appropriate for small problems and the parallel

size (m by m)	uni-1 (secs)	par-1 (secs)	par-2 (secs)	par-4 (secs)	par-8 (secs)
1	2	29	29	29	29
2	7	40	42	42	42
3	13	52	52	51	51
4	19	64	61	64	64
5	27	76	73	71	74
6	38	92	83	77	81
7	51	110	94	84	88
8	65	126	101	95	102
9	82	142	112	106	107
10	100	159	123	113	112
11	118	177	143	124	118
12	139	198	158	129	126
13	160	222	177	141	129
14	182	248	196	152	135
15	210	270	213	162	140
16	242	298	228	177	154
17	277	323	238	191	165
18	309	347	254	201	172
19	342	372	265	210	180
20	374	399			
21	419	430			
22	467	463			

Table 3.7: PVM Tabular Results

size (m by m)	uni-1 (secs)	par-1 (secs)	par-2 (secs)	par-4 (secs)	par-8 (secs)
1	2	28	25	25	25
2	7	57	40	32	32
3	13	89	56	39	39
4	19	124	77	55	49
5	27	160	101	68	60
6	38	214	127	80	71
7	51	269	150	95	83
8	65	314	186	110	95
9	82	361	219	130	113
10	100	406	248	145	130
11	118	461	275	162	148
12	139		312	180	162
13	160		339	197	181
14	182		376	219	198
15	210		423	244	220
16	242		474	272	245
17	277			302	269
18	309			331	290
19	342			357	
20	374			392	
21	419			415	
22	467				

Table 3.8: Concert/C Tabular Results

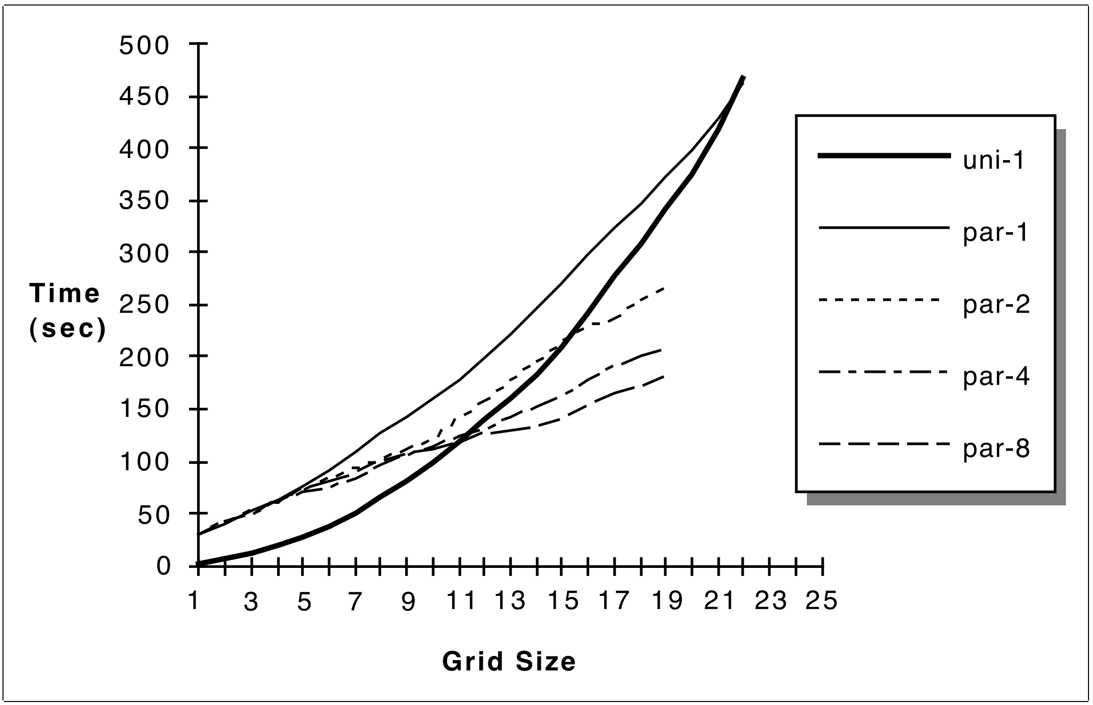


Figure 3.23: PVM Graphical Results

version more appropriate for large problems. The parallel code written in PVM consistently outperformed the Concert/C version.

3.7 Conclusions

In this chapter I have presented two clustering algorithms which cluster the tables of a scientific database based on the user's usage patterns. This clustering was performed using two types of adaptive algorithms (genetic algorithms and neural networks) to decrease the clustering time. Once these clusters have been generated they allow the user to browse through the hierarchy of clusters to see which tables should be important given their area(s) of interest. The original scientific database remains untouched allowing multiple different clusterings to exist simultaneously.

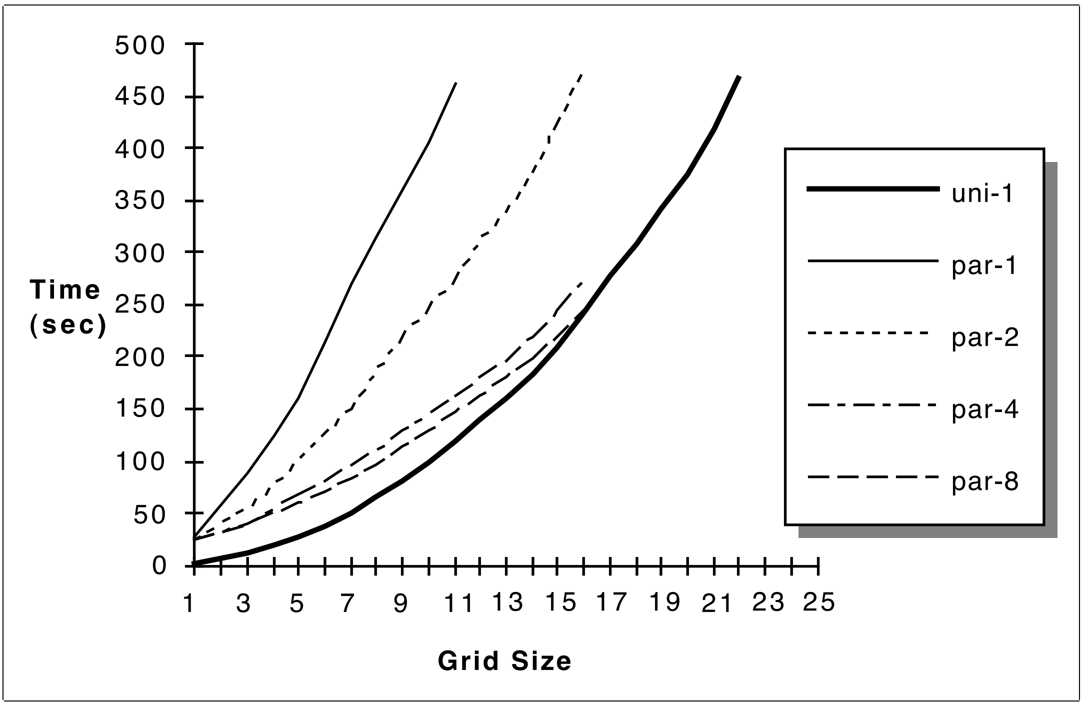


Figure 3.24: Concert/C Graphical Results

3.8 Enhancements

An enhancement that could be used on both clustering algorithms would be to 'average' multiple runs of each. This is difficult due to the variety of clusterings that are possible with each algorithm. A higher-level description of the clusters generated may alleviate this problem.

Another general enhancement would be to give a weight to the neighbours. Currently only the fact that two tables are neighbours is used. Either the tables are neighbours or they are not. Instead the count of the number of times tables are neighbours could be used to give a spectrum of neighbourlyness.

Currently the number of tables in a cluster is unbounded. Another possible factor in the clustering could be cluster size. This would encourage clusters to be a 'reasonable' size for easier viewing.

Currently tables that have never been used in any list are not involved in the clustering and do not appear in the final clustering. This makes it easy to see tables that were important and but it completely hides tables that have never been used. These tables could be given their own cluster separate from the rest after the clustering is completed, or they could be involved in the clustering if they were all given usage percentages of 0. This would increase the time taken to generate the clusters, but would make all the tables visible in the final output.

3.8.1 Genetic Algorithm

The payoff function remains constant throughout the run of the genetic algorithm. It might be better to vary the payoff function. Initially weight similarity could be emphasized to create the initial clusters. Then neighbourlyness could be emphasized to bring the clusters together. This could be done by leaving the individual payoff functions alone and changing the weights for their combination.

As well as using multiple chromosomes in a population, multiple independent populations of chromosomes could be used. Each of these populations may generate different sets of clusters. Allowing these populations to merge occasionally may allow better clusters to be generated in shorter time.

The genetic algorithm clustering code could also be converted to parallel form, though the genetic algorithm code is much larger than the neural network code. Since each chromosome has its payoff determined independent of the others this could be done in parallel. This could allow larger populations to be used, or speed up the existing population size.

3.8.2 Neural Network

The size of the neural network is currently set so every table could have its own cell in the array. A bigger array would allow the clusters to spread out further and more clearly show which clusters are near each other and which are further apart. This would increase the clustering time however.

Once the tables have been assigned positions in the array, these relative positions could be used to cluster the clusters. Using the physical distance between the tables in the array and the numerical differences between their weight vectors these tables could be clustered into a hierarchy.

Currently the parallel versions of the neural network clustering code deal with the input weight vectors one at a time. As the clustering proceeds and the neighbourhood shrinks it is very likely that multiple weight vectors could be processed simultaneously. This would further increase the degree of parallelism.

Chapter 4

SANDBOX

4.1 Introduction to the SANDBOX

Much of the data that is stored in scientific databases was collected through experimentation. Using virtual reality an investigator can 'recreate' the original experiment, collecting data from the scientific database in much the same way that the original data was collected. The investigator places virtual instruments into a virtual environment and collects data from the scientific database without ever typing in a query. Each investigator uses familiar measuring instruments and collects data in a familiar way. Using virtual reality, visualization, and hypertext we can hide the scientific database from the user behind a familiar facade. I call this interface the SANDBOX: Scientists Accessing Necessary Data Based On eXperimentation.

This gives the user a 'virtual laboratory' that can be configured for different experiments on different scientific databases by loading in different sets of instruments, and environments. The exact instruments, and the way space and time are modeled, will depend on the individual experiment. The laboratory can become as large as the universe or small as an atom, it can move through time or space, depending on the experiments being run inside it. Space can be measured in angstroms, miles, or light years. Time can be measured in nanoseconds, days, or millennia. Since the instruments are virtual, they can be calibrated to display their values in whatever manner or scale the user chooses.

Scientific databases often store information in multiple ways: files, relational databases, object-oriented databases. The database can contain numeric data, textual data, or graphical data. The database may also include meta-data (notes, drawings, diagrams, maps, photographs, sounds, etc.) This data can be seamlessly integrated into the virtual reality interface, giving the user additional information of any kind. As the user recreates the experiments the appropriate information is retrieved from the appropriate source. This allows the user to simultaneously see and relate different types of information from different sources. As this information is presented in a familiar way the user can analyze the data while retrieving it.

Data in the database is accessed through the instruments. As the user places virtual instruments into the virtual environment, they react as the real instruments would. These instruments allow the user to visualize the contents of the database before any actual data is retrieved, so the user can browse through the data. The instruments give the user feedback. The investigator can use this feedback to add additional instruments to the experiment, move the instruments to other locations, or remove unnecessary instruments. Once the user has placed the appropriate instruments into the environment and set the appropriate time interval, the information is retrieved from the database and stored in an external file for further use. The SANDBOX rewards the investigator's familiarity with the environment, not their familiarity with the database.

Section 4.2 discusses the definitions used in the SANDBOX. Section 4.3 discusses the overview of the SANDBOX. Section 4.4 discusses the instruments are defined and linked to the database in the SANDBOX. Section 4.5 discusses how actual space is mapped to virtual space in the SANDBOX. Section 4.6 discusses how actual time is mapped to virtual time in the SANDBOX. Section 4.7 discusses the two modes of operation in the SANDBOX. Section 4.8 discusses the components of the SANDBOX in detail. Section 4.9 discusses the steps to be taken to add this interface to an existing database. Section 4.10 discusses my conclusions.

Parts of this research were previously published as [JFLD94, JF94b, JF94a].

4.2 Definitions

I am using the following set of definitions:

- actual environment - the region of space and time in which all the actual experiments took place. This is bounded by the space and time boundaries of the actual experiments.

- virtual environment - the region of space and time in which all the virtual experiments take place. The total amount of virtual space may be larger than the space that the investigator has physical access to (e.g. the user may be able to 'see' objects within a 100 foot radius, but may only be able to interact with objects within a 3 foot radius.) Through zooming and panning the investigator has physical access to the entire virtual environment. Virtual time is the time that passes in the virtual environment. Within this time the user has access to the current moment.

- site - a location in space/time (x, y, z, t) where an instrument can be placed (either in the actual environment or the virtual environment.) I am assuming that at any point in time (t) there is at most one location (x, y, z) in the environment for each site, but the location of that site can change over time (e.g. a rover conducting experiments). While a site is a single point in space/time the instrument placed there may collect/display data from a region. Each site has a unique identifier. If the location of that site does not change over time with respect to the environment then the location can be the unique identifier.

- instrument - a device with the ability to convert site, time into values.
- experiment - an instrument at a site (instrument in operation) mapping instrument, site, time into values.

Actual instruments in the actual environment collected data from nature which was then stored in a scientific database. Virtual instruments placed in the virtual environment retrieve data from the scientific database.

I am assuming that actual space is mapped into virtual space, and actual time is mapped into virtual time. This implies that there can not be more than one copy of a site in the virtual environment simultaneously. Allowing actual time to map into virtual space would allow more than one copy of a given site in the virtual environment simultaneously. A space mapping function converts actual space (in the actual environment) to virtual space (in the virtual environment.) For example, 20Km of actual space can be mapped into 10 feet of virtual space. A time mapping function converts actual time (in the actual environment) to virtual time (in the virtual environment.) For example, 1 day of actual time can be mapped into 30 seconds of virtual time.

4.3 System Overview

An overview of the system is shown in Figure 4.1. The SANDBOX is composed of three main components: the Virtual Reality (VR) Interface, the Preprocessor, and Local Memory.

The VR interface is responsible for the maintaining the virtual environment, representing the virtual instruments visually and audibly, as well as monitoring the user's

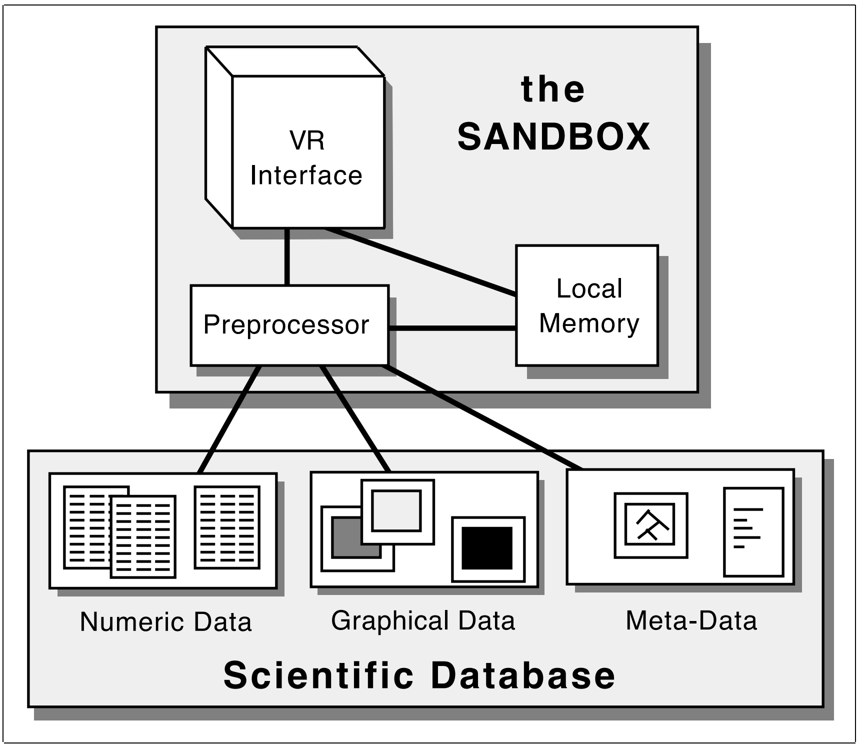


Figure 4.1: Overview of the SANDBOX

actions within the virtual environment. Based on the user's actions, the VR interface sends requests to the preprocessor to obtain the necessary data. Based on the current virtual time the VR interface displays the appropriate data from the local memory.

The preprocessor is responsible for interfacing with the various components of the scientific database to quickly retrieve data according the the needs of the VR interface and store it in local memory.

The local memory maintains all the information necessary to support the VR interface. This includes information on the user, the various tracking devices, and instruments that the user has placed in the virtual environment.

4.4 Instruments

As the user recreates the experiments with the virtual instruments, the SANDBOX retrieves the appropriate information from the appropriate source. Some instruments are linked to numeric and textual experimental data; some are linked to graphical, audio, and other experimental data; some are linked to meta-data. The instruments give the user a more familiar way of dealing with the data stored in a scientific database.

Before a virtual instrument can be used to display data values, the instrument must be linked to the scientific database. This linkage is a combination of an instrument class, an access function, and a set of filter functions. The instrument classes convert data into visual and audible form; the access functions retrieve data from the database; the filter functions perform transformations on the data.

4.4.1 Virtual Instruments

Each instrument is a member of some instrument class (e.g. each individual thermometer instrument is a member of the thermometer instrument class) as shown in Figure 4.2. The class maintains static information common to all instruments of that class.

Within the SANDBOX the values an instrument is measuring can be mapped into both a visual and audible form. This allows the user to see and/or hear the values. Each instrument class is composed of two functions, a G-function to map values into graphical form and an A-function to map values into audible form. The G-function and the A-function function independently, and can be stored independently, allowing the designer to choose appropriate combinations to create the needed instrument.

Each instrument class has a G-function to represent the instrument and its current value(s) in terms of graphics primitives such as cubes, cylinders, spheres, and pyramids. For example, Figure 4.3 shows a sample G-function for the thermometer instrument class. A thermometer can be created by graphically created by placing a

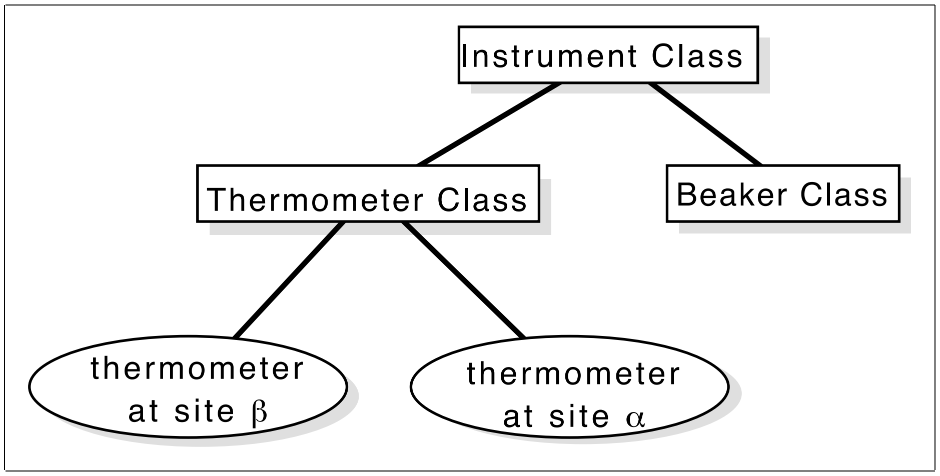


Figure 4.2: Instrument Classes

white cylinder atop a red cylinder atop a red sphere. The hotter the temperature, the taller the red cylinder and the shorter the white cylinder will be.

Each instrument class has an A-function to represent the instrument and its current value in terms of audio primitives. Each instrument can be linked to an external sound file, or to a function which synthesizes a sound. Instrument sounds can either be momentary or continuous. With momentary sounds, there is sound only when the value being monitored changes. With continuous sounds, there is a continuous sound representing the current value of the value being monitored. Momentary sounds are appropriate when values change rarely, but by large amounts (e.g. amount of rainfall.) The amount of change could be reflected by the amplitude, pitch, or length of the sound. Continuous sounds are appropriate when values change often, but gradually (e.g. temperature.) For example, the thermometer instrument class could map the temperature values onto the amplitude of a sound - the higher the temperature, the louder the sound.

Each instance of an instrument class, that is each virtual instrument placed at a site, maintains its own dynamic data. The dynamic data includes the data currently being visualized by this instrument (e.g. a thermometer instrument maintains tem-

```
void SANDBOX_drawThermometer(float * base, float percent)
            // draws a thermometer in the virtual environment
            // base is the XYZ coordinates of the base
            // percent is the percent that should be red
    {
    float vertX, vertY, vertZ, sphereDat[4];
    float radball, redheight, whiteheight;

    if (percent < 0.0) percent = 0;
    if (percent > 1.0) percent = 1.0;

    #define radther 0.02
    radball = 2 * radther;
    redheight = 0.2 * percent;
    whiteheight = 0.2 - redheight;

    vertX = base[0]; vertY = base[1]; vertZ = base[2];

    lmbind(MATERIAL, SANDBOX_gred);                 // draw red sphere
    sphereDat[0] = vertX; sphereDat[1] = vertY;
    sphereDat[2] = vertZ; sphereDat[3] = radball;
    sphdraw(sphereDat);

    pushmatrix();                                   // draw red cylinder
        translate( vertX - radther,
            vertY + radball + redheight,
            vertZ - radther);
        scale(radther, redheight,radther);
        SANDBOX_drawclosedcylinder();
    popmatrix();

    lmbind(MATERIAL, SANDBOX_white);                // draw white cylinder
    pushmatrix();
        translate( vertX - radther,
            vertY + radball + 2 * redheight + whiteheight,
            vertZ - radther);
        scale(radther, whiteheight, radther);
        SANDBOX_drawcylinder();
    popmatrix();
    }
```

Figure 4.3: Sample Thermometer G-Function

peratures for several days at a single site) and the current visualization settings on this instrument (e.g. the minimum and maximum temperatures to be displayed.)

4.4.2 Access Functions

An access function takes a site identifier, and a time range and retrieves information in the form {time, data values} from a specified part of the scientific database. This access function could retrieve data from a tabular file, a file containing a single large data value (e.g. a satellite photograph), a table in a relational database, or an object in an object-oriented database.

An access function has the following form:

	{source, {time attribute}*, {space attribute}*, {value attribute}$^+$, S-function, T-function}$^+$
source:	file, object, database relation
time attribute:	locates the data values in time
space attribute:	locates the data values in space
value attribute:	data values at that point in space and time
S-function:	maps a site into space attributes
T-function:	maps a time into time attributes

Given a site and a time, the S-function and T-function convert these values into appropriate space and time attributes for this part of the scientific database, respectively. An access function may not have any time attributes or space attributes if the time and/or site uniquely determined the source.

Each access function must have at least one value attribute though this attribute can be arbitrarily large (e.g. a single site photograph, or a single sound recording.)

Once the space and time attributes are generated by the S-function and T-function, the appropriate data values in the source for those space and time attributes can be retrieved. The inverse of the T-function is used to convert each of the time attributes to the internal time representation. This time index is added onto each {time, data values} data value so that all of the data values in local memory have a common time index.

Times in the scientific database can have multiple formats, and multiple values (e.g. minutes and seconds, or month and day and year.) As the actual experiment took place over a certain bounded range of time, we can convert these multiple time values to a single time index for the sake of quicker indexing. This way all of the times in memory have the same format. Time in the SANDBOX is taken as absolute time from the beginning of the experiment with the granularity of time dependent on the individual experiment. However, it is important that the actual time values be stored in memory as well so the user can be assured that the values being output by the interface are the same values that were recorded during the actual experiment.

For example, if a years worth of measurements were collected, some once per day (Jan. 15th), some once per hour (Jan. 15th at 2pm), and some once per minute (Jan. 15th at 2:04pm) we could convert all these measurements to a single time index. This index would start at Jan. 1st at 12:00am and end at Dec. 31st at 11:59pm with a granularity of one minute. Jan. 1st at 12:00am is set to 0, Dec. 31st at 11:59 is set to 525,599 (60 minutes times 24 hours times 365 days minus 1.)

4.4.3 Filter Functions

The SANDBOX uses two types of filter functions: access filters and display filters. Access filters filter the data values as they are retrieved from the scientific database before they are placed into the local memory. Display filters filter individual data values before they are displayed by a virtual instrument. See Figure 4.4.

The access filters are applied only once. Access filters are used to ensure all the

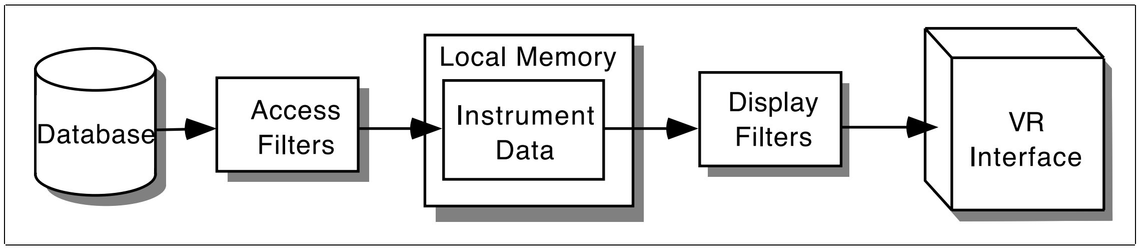


Figure 4.4: Filters

data stored in the local memory has the same format. For example at a given site some temperatures may have been recorded in degrees Celsius and others in degrees Fahrenheit. An access filter could insure that all the temperature measurements are in degrees Celsius. Access filters ensure that null values are consistent. Different experiments may have different null values (0, 'X', -99, etc.) An access filter could ensure that all the null values match. Access filters are used to perform time consuming operations on large data sets. For example rotating, cropping, and enhancing a satellite photograph would be very time consuming to perform for each frame. If the user does not require interactive control over these operations, it would be better to perform such operations once and store the altered version for display.

The display filters are applied each time a value is retrieved from the local memory, before it is displayed by one of the instruments. Display filters are used to give the users control over the virtual instruments. For example a user could set the minimum temperature to be displayed, or the maximum temperature to be displayed.

As with the instrument's G-functions and A-functions, the access filters and the display filters can range from very specific to very generic. Stored independently, they allow the interface designer to choose appropriate sequences of filters.

4.4.4 Linkage

A linkage is a combination of an instrument class and a set of access functions and associated filter functions:

$$\boxed{\{instrument\_class\ ,\ \{site\,range\ ,time\,range\ ,access\,function, filters\}^{+}\}}$$

Each instrument class has a set of access functions and filters based on the range of sites and times they apply to. A single instrument may be used to visualize data stored in different ways. For example, rainfall may be recorded as total accumulation, or as rainfall over the last hour. Wind direction and speed may be recorded as x, y, z components or as direction and speed. Each instance of an instrument in an instrument class may have a different access function and set of filters depending on when and where that instrument is placed.

When the user places an instrument at a site for a given range of times, a new instance of that instrument class is created. The appropriate access functions for that instrument class at that site over that range of times read the data values into that instrument's local memory, passing them through the access filters. The display filters are brought into that instrument's local memory as well, to be used when the data values are to be displayed.

4.4.5 Generic Instruments

Each scientific database requires its own set of instruments. The more familiar an instrument is to the user, the more obvious is its function. This may mean that different users of the same database will use different instruments to access the same data. Once these instruments' G-functions and A-functions have been defined, these instruments can be stored in a library and used again in other scientific databases in similar fields of study.

Some generic instruments include:

- fillers - fill/empty according to values
- expanders - expand/contract according to values
- pointers - point in a specific direction
- brighteners - brighten/darken according to values
- enumerators - more/fewer according to values
- colourers - change colour according to values
- spinners - spin fast/slow according to values
- densers - density varies according to values
- texters - display alphanumeric characters according to values
- markers - mark a location the user wants to remember

Examples of some of these instruments can be seen in Figures 4.5, 4.6, 4.7, 4.8, and 4.9.

Figure 4.5 shows meteorological information. Filler instruments in the form of a thermometer and a water beaker measure temperature and rainfall. A pointer instrument in the form of a wind-sock measures the wind direction.

Figures 4.6 and 4.7 show 1992 population data for the Great Lakes states. Figure 4.6 uses expander instruments to visualize the population of each state. The taller the column, the larger the population of that state. Figure 4.7 uses enumerator instruments to visualize this same information in a different way. Each person-enumerator represents one million people.

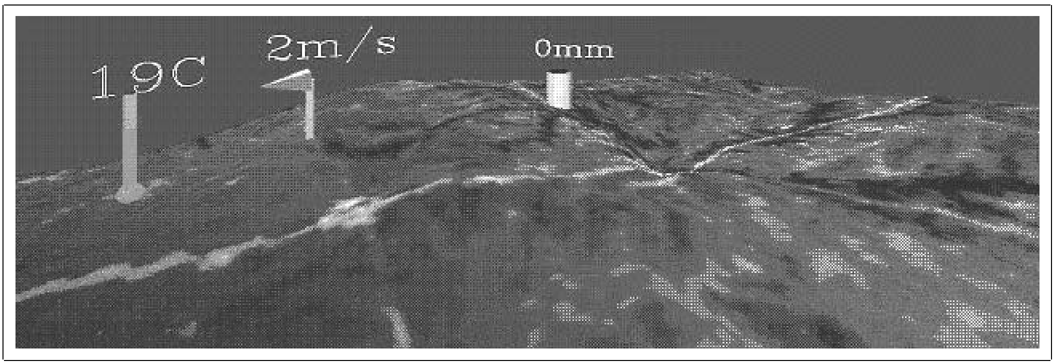


Figure 4.5: Meteorological Information

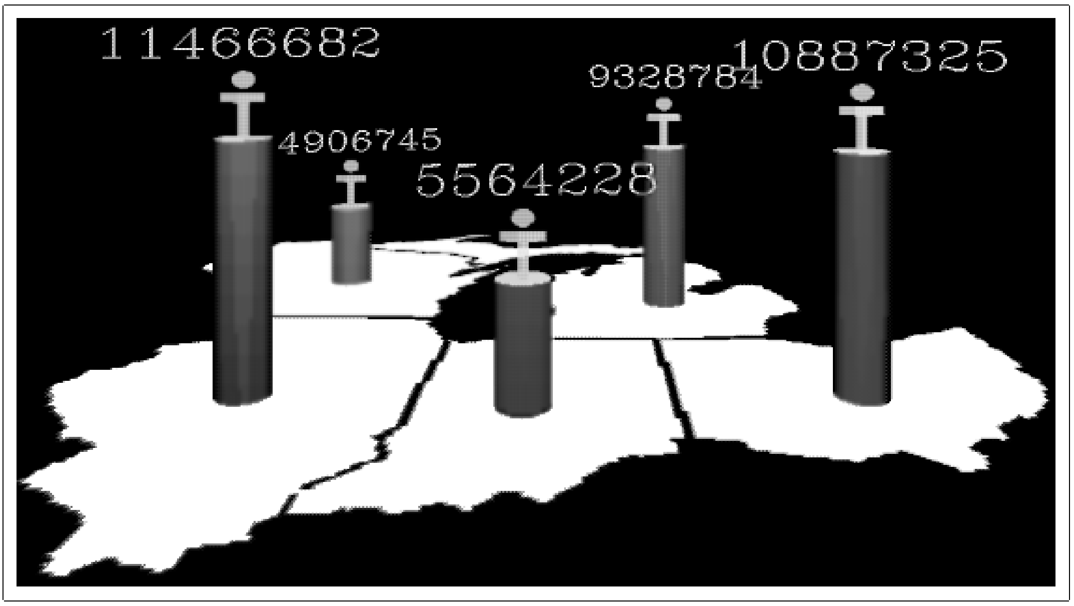


Figure 4.6: Great Lakes Census Data With Expanders

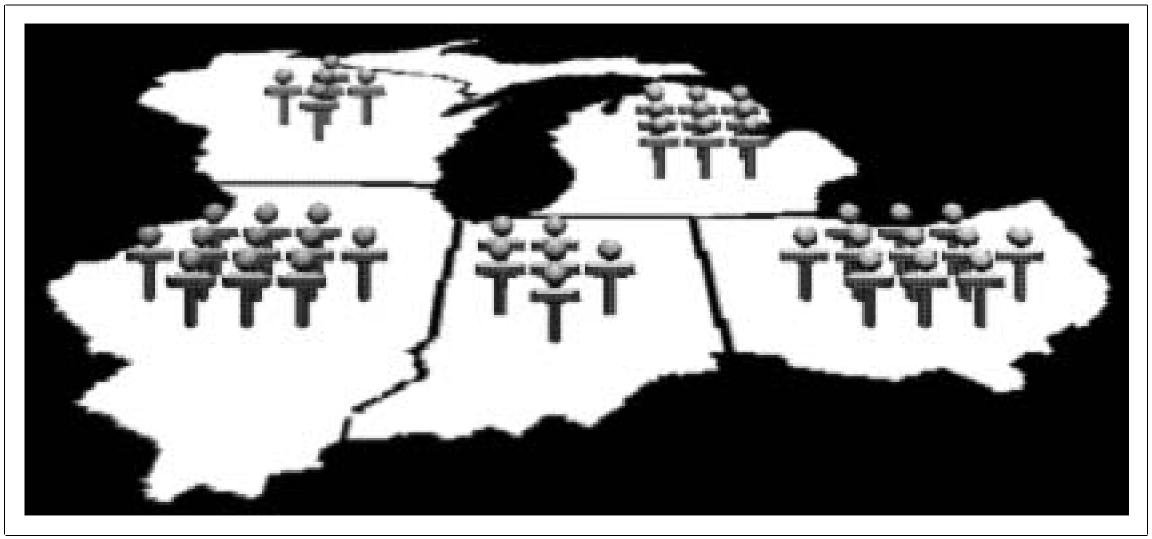


Figure 4.7: Great Lakes Census Data With Enumerators

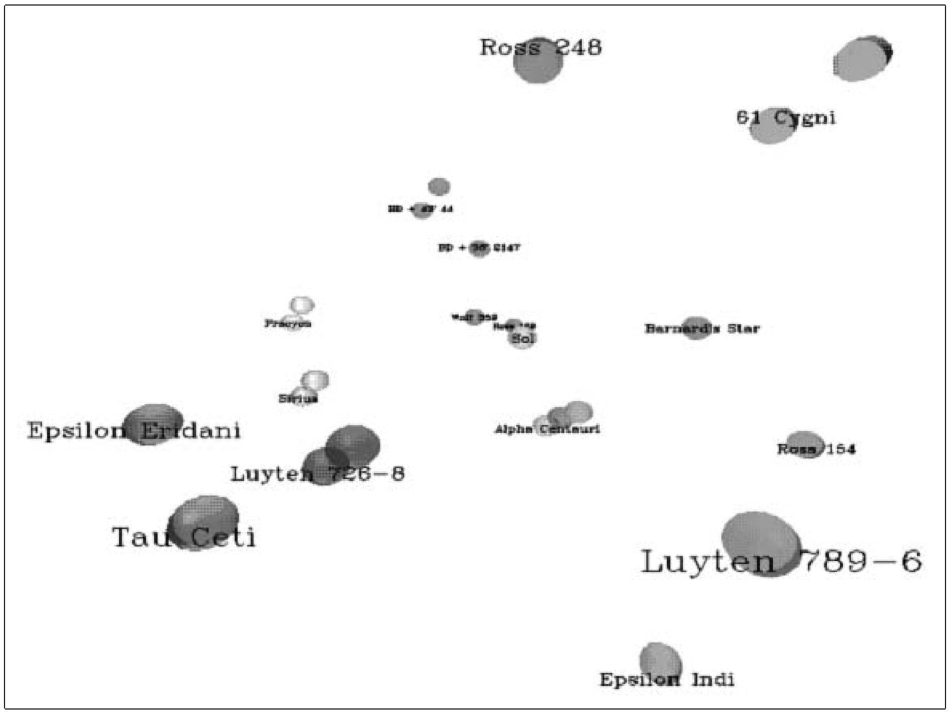


Figure 4.8: Star System Data Showing Number, Temperature

Figures 4.8 and 4.9 show a 3D representation of the sixteen nearest star systems to our sun. Both figures show the stars as seen by a person standing (floating) 25 light years away towards the constellation Pisces looking back towards our sun. As its common for star systems to have more than one star, both figures use enumerators, one sphere for each star. As the temperature of a star is indicated by its colour, Figure 4.8 uses colourer instruments to visualize the temperature of each star (from cool red to hot blue.) Figure 4.9 uses colourer and expander instruments to show not only the temperature of each star, but also its relative mass compared to our sun. The more massive the star, the bigger the sphere.

4.5 Space Mapping

The space mapping function maps actual space (x, y, z) into virtual space (x', y', z'.) This function depends on the environment and may be different for the different

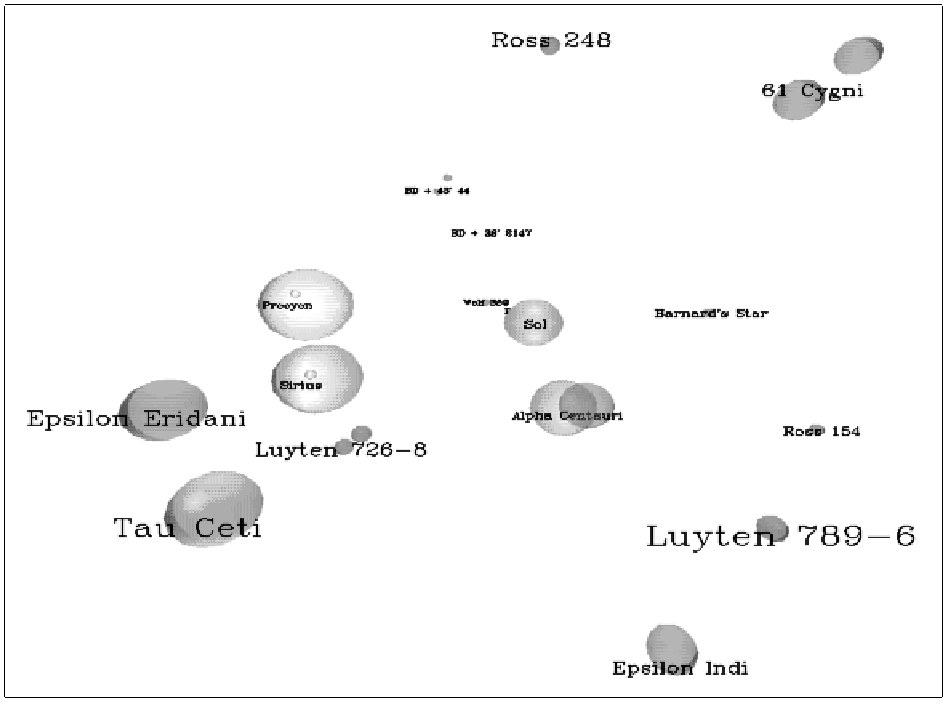


Figure 4.9: Star System Data Showing Number, Temperature, Mass

dimensions. The user is given control of this function to alter the size of the virtual space (enlarging it, shrinking it, moving it, rotating it.)

$$Virtual\ Space(x', y', z') = f_s(Actual\ Space(x, y, z)) \tag{4.1}$$

Mapping the individual actual sites into their virtual locations is straightforward and generic. However, simply showing the sites floating in space shows only how the sites relate to each other. This does not give the user a familiar environment. The more familiar the environment is, the easier it will be to work in. The closer the interface can come to mimicking the actual experiment, the more natural the interaction with the data will be. Graphical data from the database and graphical meta-data can be used to create a spacial context for the sites to reside in. Figure 4.10 shows a set of sites without an environment. Figure 4.11 shows these same sites with the environment added.

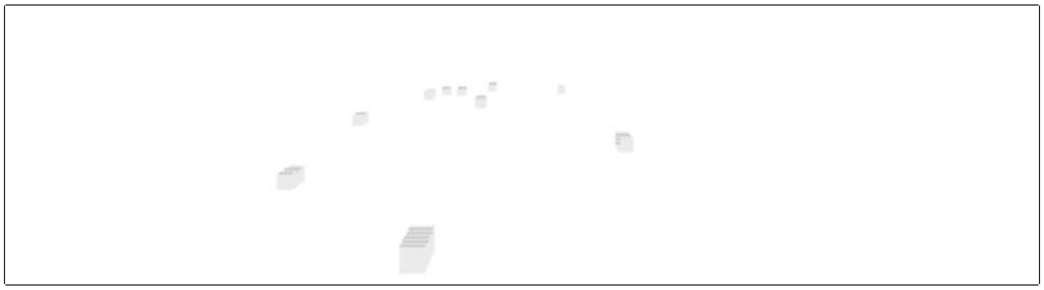


Figure 4.10: Sites Without Surrounding Environment

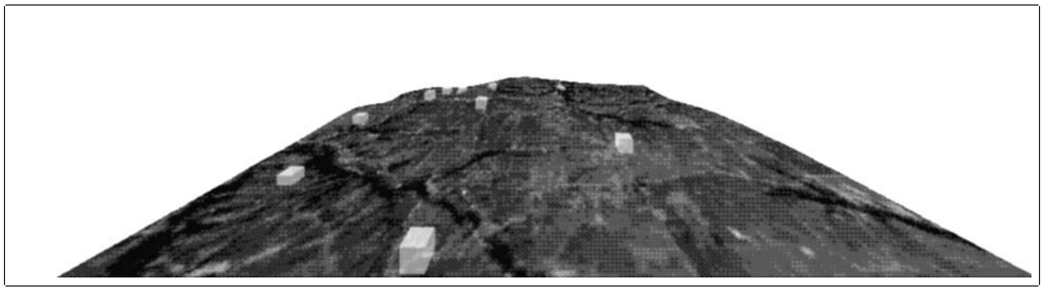


Figure 4.11: Sites With Surrounding Environment

4.6 Time Mapping

The time mapping function maps actual time (t) into virtual time (t'.) This function depends on the environment. The user is given control of this function to modify the behavior of time (speeding up, slowing down, or stopping virtual time.)

$$Virtual\ Time(t') = f_t(Actual\ Time(t)) \tag{4.2}$$

Virtual Time is then divided into individual frames. However, depending on the complexity of the scene, the actions of the user, and other executing processes, the time it takes to generate each frame may vary. In order to keep virtual time passing at a constant rate in the virtual environment there may need to be a variable number of frames for every unit of virtual time. The SANDBOX keeps track of both the virtual time and the actual time. Before each frame is generated the current virtual time is computed and used to display data from the corresponding actual time. This allows the frame rate to vary while the passage of virtual time remains constant.

Each scientific database requires its own ways of measuring and manipulating time. The more familiar these methods are, the easier they will be to use. Once these methods have been created graphically, they can be stored in a library and used again in other scientific databases in similar fields of study.

Some possible ways of measuring and manipulating time include:

- yearly calendar - choose decades, years
- monthly calendar - choose months, days, weeks
- clock - choose hours, minutes
- enumerators - choose experiment numbers

4.7 Modes of Operation

In the actual experiment the investigators first made decisions about when and where experiments would be placed, and then conducted the experiments. The SANDBOX also works in these two modes: Set up mode and Experimentation mode.

4.7.1 Set Up Mode

In set up mode the instruments, sites, and times are interlinked to show how they affect each other. Choosing an instrument shows which sites it can be placed at, and in what times it can be placed. Choosing a site shows which instruments can be placed there and which times instruments can be placed. Choosing a time shows which instruments can be placed and which sites they can be placed at.

Choosing an instrument and a site shows which times that instrument was placed at that site in the actual experiment. Choosing an instrument and a time shows at which sites that instrument was placed during the actual experiment. Choosing a site and a time shows which instruments were placed there at that time during the actual experiment.

Set up mode allows the user to browse through the data at a higher level, not looking at the individual experimental data values, but rather at when and where those values were collected. This way the user gets an idea of which instruments to place, when they should be placed, and where they should be placed before starting to actually place them. The user can see what sites and times satisfy their requirements.

Set up mode could also be useful to investigators planning the actual experiment itself. Before investigators go out to the field they can virtually visit the sites and see where and when other investigators are planning on setting up their experiments. This should make it easier to avoid conflicts before the experiments begin.

4.7.2 Experimentation Mode

In experimentation mode the user sets all three parameters: time, site, and instrument to retrieve data from the database. Experimentation mode has two methods: Independent and Dependent.

The independent method allows the user to choose times independently from instruments and sites. That is, a user chooses which instruments to place at which sites, and then independently chooses the time interval to be displayed. This makes it easier to place instruments and choose times than in the dependent method, but means that instruments may not be able to access data for the chosen time interval.

The dependent method allows the user to choose the times for which each instrument should be placed at a given site. This makes it easier for the user to request more complex combinations of information. For a given site the user may want tem-

```
repeat
    User input (movement and/or actions)
    Retrieve data for virtual time t
    Visualization and virtual environment update
until user exits
```

Figure 4.12: Sequence of Actions in the SANDBOX

perature information for the first three days followed by rainfall information for the next three days. In the dependent method the user would place the temperature instrument at the site for the first three days, and the rainfall instrument at the site for the next three days. In the independent method the user would need to place both the temperature instrument and the rainfall instrument for all six days to gather the same information.

4.8 System Components

The operation of the SANDBOX is centered around visualizing the appropriate data for the current frame to be displayed. For each frame the sequence is shown in Figure 4.12.

A more detailed version of the system overview shown in Figure 4.1 is shown in Figure 4.13. This system diagram shows the primary components of the SANDBOX. User input and visualization are handled by the VR interface. Data retrieval is handled by the preprocessor.

The primary component of the VR Interface is the Environment Manager. The Environment Manager manages the virtual environment. It keeps track of the user's position and actions. Based on these actions the Environment Manager sends requests for data to the Data Manager in the Preprocessor. It also uses Display Data from each

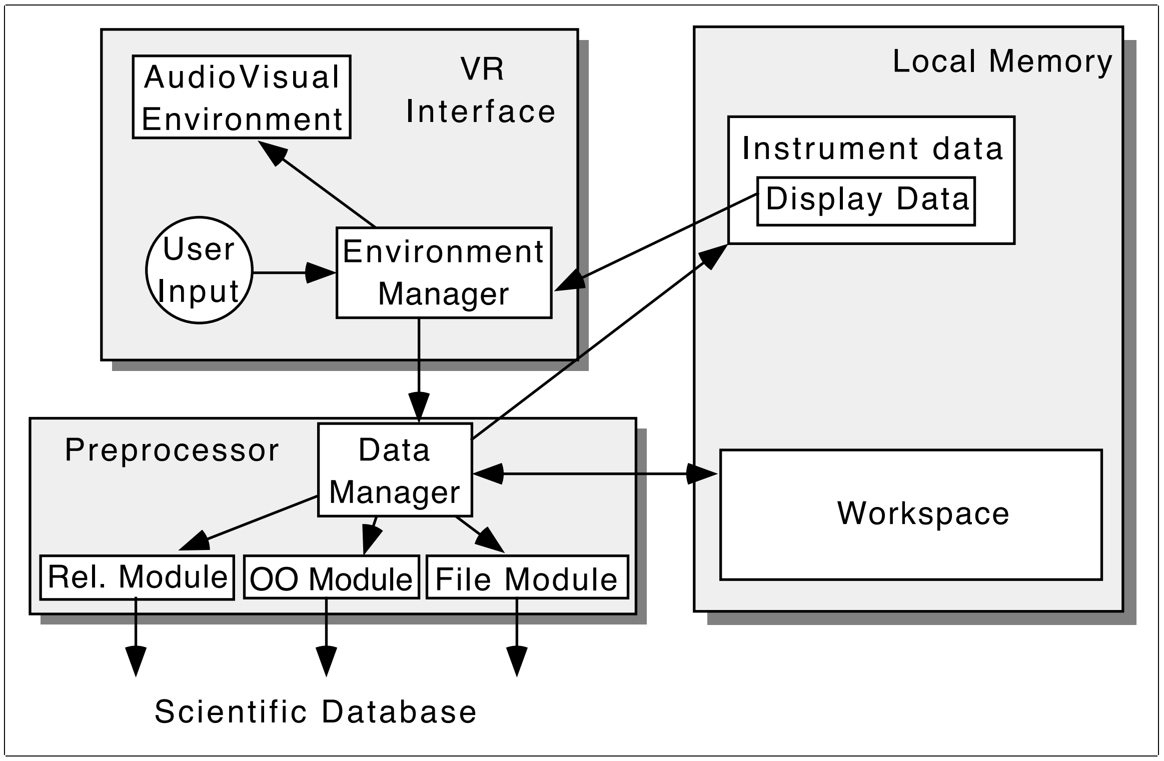


Figure 4.13: System Diagram of the SANDBOX

of the instruments in Local Memory to display images and sounds in the AudioVisual Environment.

The primary component of the Preprocessor is the Data Manager. The data manager receives requests from the Environment Manager and sends requests to the appropriate database access module (relational database module, object-oriented database module, file module) to retrieve that data. This data is then moved to the appropriate place in the Local Memory.

4.8.1 Data Storage

Current database access methods are not fast enough to support the needs of virtual reality [GQV$^+$93, RCM93]. Virtual reality requires very fast access times. For smooth movement of a three dimensional image, at least 15 frames per second must be generated for each eye [WS82]. Generating a frame involves accessing all the relevant information, converting it into graphical form, and drawing it once for each eye. Each frame must be generated within 67 milliseconds. Relevant portions of the database need to be brought into local (fast) memory before the visualization can begin, or as the visualization is proceeding in a form of progressive refinement[Bro88]. Typically, during scientific visualization all of the necessary information is loaded into RAM before the visualization begins [HS89]. This is clearly not possible here given the huge amount of data involved. Since visualization is used while retrieving information, not just afterwards, the entire database needs to be accessible.

Data storage is therefore hierarchically organized into four levels:

- Display data - The data currently ready to be converted to graphical and audible form in the current frame. Each instrument maintains its own display data.
- Instrument data - The data currently being indexed by an individual instrument. This includes all the data for this instrument at this site for the currently selected time interval. For each frame, a subset of this data becomes the Display Data. The instrument data also includes the current instrument settings.
- Workspace - The workspace acts as a cache for the Database data. It holds data that is not currently Instrument data, but might become Instrument data in the near future.
- Database data - The scientific database itself. The database may be local, or remote, and is probably a heterogeneous environment.

As an example, if the user is displaying temperatures at site X over a 3 day period then the display data holds the individual temperature being displayed in this frame. The instrument data holds all the temperature values for site X for those 3 days. The workspace may hold all the temperature values for all the sites where temperature can be measured over those 3 days. The database contains temperatures for all the sites where temperature can be measured for all the days in the database, as well as lots of other data.

Retrieving data from the database may involve combining information from multiple sources (multiple files, tables, or objects.) If additional space is available in secondary storage then these retrievals could be sped up my maintaining views of the database corresponding to the necessary combinations. This way the appropriate views could be accessed rather than generating them each time.

The display data, instrument data and workspace are part of the local memory. They are given priority in that order. The workspace can be sacrificed to make room for instrument data and instrument data can be sacrificed to make room for display data.

If all of the instrument data can not be stored in local memory there will be a moderate performance penalty. The preprocessor would need to divide the instrument data into blocks based on time. These blocks would be stored on disc as a cache for the database and brought into local memory as needed. The preprocessor would then ensure that the appropriate blocks of instrument data are available for the next several frames for all of the instruments currently placed at sites. Using the previous frames as a guide, the preprocessor can estimate which blocks of data will be needed in the future. Blocks from near in the future are given priority over blocks in the far future. Blocks in the past can be discarded to make room.

If all the visualization data can not be stored in local memory then there will be a severe performance penalty. Retrieving the the data to draw each frame would require accessing secondary storage. With each data retrieval taking approximately 10 milliseconds, this additional overhead would make the system unusable.

4.8.2 Data Retrieval

As the user chooses instruments, times, and other experimental parameters, the VR interface passes this information along to the preprocessor. In the SANDBOX 'where' is determined by the choice of site, 'when' is determined by the choice of time, and 'what' is determined by the choice of instrument. The preprocessor determines which parts of the database are likely to be accessed in the near future. Relational tables can be partitioned vertically and horizontally, objects can be isolated, and files can be marked. These blocks of information can then be moved into local memory before they are needed by the visualization system.

When the user selects an instrument, the data manager needs to quickly display all the possible sites where the selected instrument can be placed. In the independent method, the list of available sites can be maintained in the static data of the instrument class, and so this list is already available in local memory. In the dependent method the data manager creates an autonomous process to generate the list of available sites based on the currently selected time interval and place this information into the instrument data for the chosen instrument. Once this information is available the possible sites can be highlighted.

The data manager also creates an autonomous process to begin reading in all of the appropriate data (ordered ascending my time) into the workspace in local memory based on the currently selected instrument and time intervals:

$$\boxed{\Pi_{time,space,value}\left(\sigma_{time\,\epsilon\,current\_time\_intervals}\left(source\right)\right)}$$

When the instrument is placed at a specific site, the source can be further partitioned horizontally using the selected site. Information on the selected site that was retrieved by the autonomous process is moved from the workspace to memory indexed by the chosen instrument (becoming instrument data.) Information on other sites is kept in the workspace if there is room. Since the user placed one copy of this instrument, it is likely that they will place another, so this information is retained.

$$\boxed{\Pi_{time,value}\left(\sigma_{time\,\epsilon\,current\_time\_intervals,site=specific\_site}\left(source\right)\right)}$$

If the autonomous process was not able to move all of the appropriate data into the workspace before the instrument was placed at a specific site, then the first process is terminated and a second autonomous process is created to read in only that information for the selected instrument and the selected site and the selected time. This information is moved directly from the database to memory indexed by the chosen instrument (becoming instrument data.)

When the user increases the time interval being displayed, additional information must be loaded into instrument data for all the instruments currently placed at a site.

$$\boxed{\Pi_{time,value}\left(\sigma_{time\,\epsilon\,additional\_time\_interval,site=specific\_site}\left(source\right)\right)}$$

All of these attributes are loaded into instrument data supplementing the existing information for that instrument at that site. When the user reduces the time interval being displayed, similar information can be discarded from the instrument data. If there is room, this information can be kept in the workspace.

The actual queries generated will depend on the specific way the data is stored. This means that traditional query optimization practices can be applied to the individual queries to reduce the amount of access to the database and speed the data retrieval.

4.8.3 Instrument Data

The SANDBOX requires rapid access to the data values in memory based on time. Given a time index, the data structure must be able to quickly find the nearest stored time and its associated data values. In order to achieve fast access to the instrument data values we are using a modified version of the T-tree data structure proposed by Lehman and Carey [LC86a, LC86b]. This was originally proposed to aid in indexing in main memory databases [DKO$^+$84, Sin88, AP92, Ulu92].

In the SANDBOX, each instrument placed at a site keeps its own T-tree of instrument data as shown in Figure 4.14. Each node in that tree is shown in Figure 4.15. A T-tree is a modified AVL tree that holds multiple data items in each node. Each data item contains the time values and the data values for a particular instrument at a particular site at a particular time. As well as the data items, each node stores the minimum and maximum time values, and a count of the number of data items in the node. These T-trees are ordered by time index.

By storing multiple data values in a single node the T-tree reduces the amount of extra information that needs to be maintained in memory compared to a standard AVL tree. This frees up more space in memory to hold data values. Each node of the T-tree has twice as much supplementary information as a node in the AVL tree. An AVL tree has two child pointers and a balance factor. A T-tree has all of that plus a pointer to the first data item, a pointer to the last data item, and a count of the data items. If there are more than two data items in the T-tree then the T-tree will take up less space in memory.

Data values are retrieved from the database for a set of time intervals (e.g. a day's worth of data, or an hour's worth of data.) Each of these time intervals is given its own node, and inserted into the T-tree. In an AVL tree each tuple of data would need

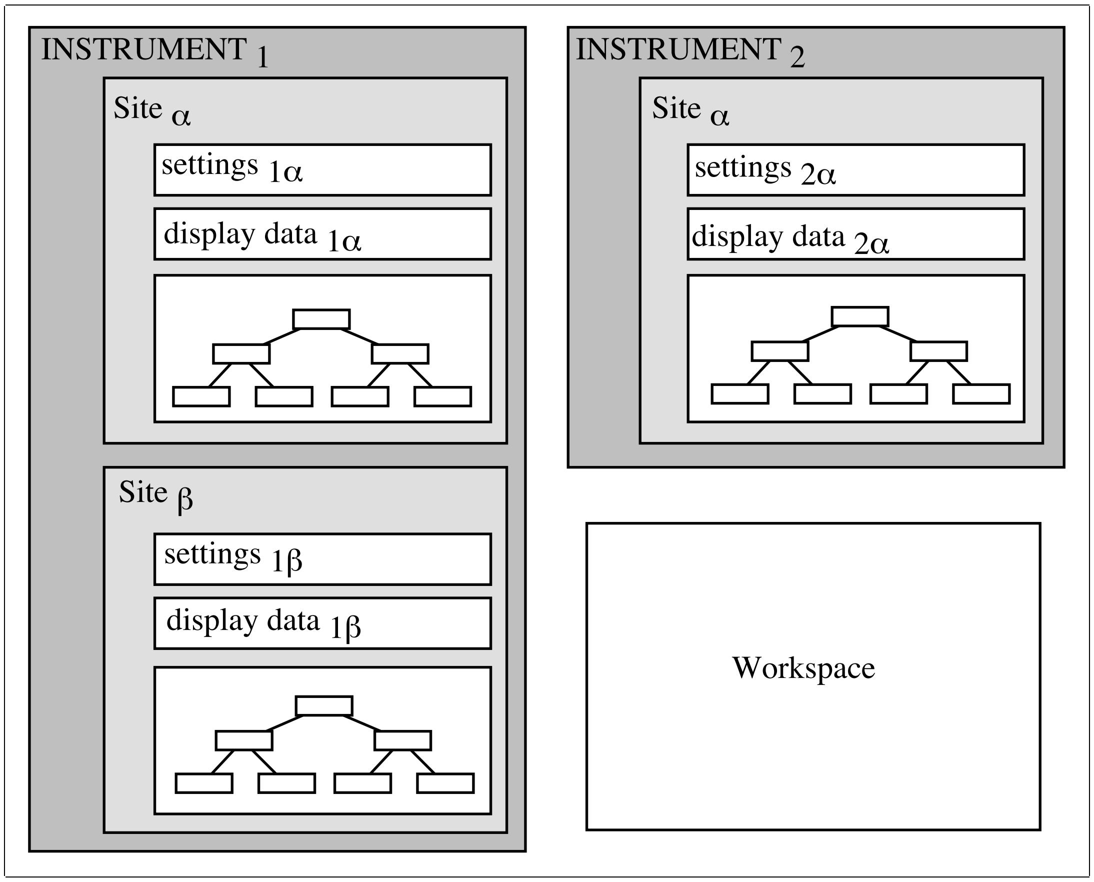


Figure 4.14: Enhanced View of Local Data

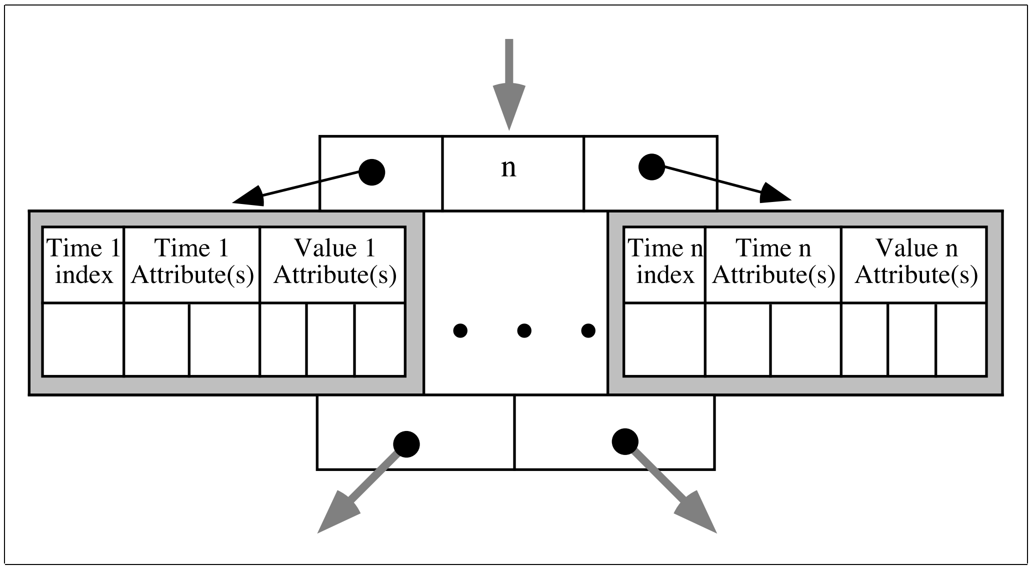


Figure 4.15: Each Node of Instrument Data

to be given its own node and inserted independently. As there are fewer nodes in the T-tree, keeping it balanced requires fewer operations, speeding up the data retrieval process. Removing a time interval worth of data values from memory is easier in a T-tree as an entire node can be removed at one time, rather than removing all of the data values independently in an AVL tree. As data is commonly retrieved from a database in chunks, rather than as individual tuples the T-tree has a more natural storage structure. The T-tree does require that all the data values for a given time interval be retrieved from the database before that interval can be inserted into the tree.

Other internal data structures such as arrays and hash tables are not practical in this environment. As the data must be kept in sorted order for eventual output the time to insert data into an array is unacceptable. As data values could have been collected at varying time rates, and the VR system needs to display the value taken at the nearest time, searching for the appropriate data using a hash table would be difficult.

Unlike indexing in full main memory databases here there are very few possible transactions. Each time a new interval of data for a particular instrument at a particular site is brought into memory, it is given its own node and inserted into the T-tree for that instrument at that site. If the new interval of data overlaps an existing interval then the data is merged. Each time an interval is removed from memory that node is deleted from the T-tree. If necessary, after each insertion or deletion the T-tree is re-balanced. If an instrument is removed from its site then the entire set of instrument data for that instrument at that site is removed from the local data.

4.8.4 Visualization

The environment manager tracks the user's current position in the virtual environment to adjust the users viewpoint. It monitors the user's interaction with the virtual environment and sends appropriate requests to the data manager for display data.

Each instrument placed at a site maintains its own instrument data indexed by time. Measurements in the various original experiments could have been taken at various time rates (e.g. temperature readings taken once per hour, wind direction taken once per minute.) With the instrument data indexed by time, measurements taken at different time rates can be quickly retrieved and related by the visualization system.

Every time a new frame is to be generated, the virtual time is converted into actual time. For each instrument currently placed at a site, its instrument value taken at the time nearest to the current actual time becomes the display value or values for that instrument. These display values are then sent to the environment manager to be converted into visible and audible form. If this nearest value is a null value then that null value is displayed.

If the appropriate instrument data is not available in local memory to become display data (because it has not been loaded into memory yet) then the visualization component will not display any readings for that instrument. Instead it will display a special value to indicate that the data is unavailable. When data becomes available (in a future frame) the appropriate data for that frame will be displayed. All of the other instruments with their display data available in local memory would still remain active. It should be noted that unavailable data is not the same as null data. If the database contains null values then those null values will be displayed by the visualization component.

The only other alternative would be to freeze the virtual environment until the requested data is available. The previous frame would remain visible, but the user would not be able to interact with the environment. This can be very disorientating as the user's actions no longer have an immediate effect on the virtual environment.

Searching for the data for a particular instrument at a particular site at a particular time is performed by starting at the root of the T-tree for that instrument at that site and making comparisons with the minimum and maximum time values to determine which node contains the requested values. Binary search is then used to search through the individual data items within the chosen node to find the data item with the nearest time value. If no node contains an appropriate range, then the data value in the tree with the nearest time is used. If the T-tree is a balanced tree with n nodes and v values per node then the maximum number of memory accesses is log(n) + log(v.) The data retrieval algorithm is shown in Figure 4.16.

```
Done = FALSE
BestMatch = NULL
currentNode = root

while (not Done)

    // Match not found within a node, return best found so far

    if (CurrentNode == NULL)
        Done = TRUE
        return(BestMatch)

    // Match not found within node with unavailable data

    if (CurrentNode == UNAVAILABLE)
        BestMatch = UNAVAILABLE
        Done = TRUE
        return(BestMatch)

    // Match found within a node

    if (CurrentNode.min.TimeIndex < t < CurrentNode.max.TimeIndex)
        BestMatch = binsearch(CurrentNode)
        Done = TRUE
        return(BestMatch)

    // Match can only be found to the left of this node

    if (t < CurrentNode.min.time)
        if (|t-CurrentNode.min.TimeIndex| < |t - BestMatch.TimeIndex|)
            BestMatch = CurrentNode.min
        CurrentNode = CurrentNode.leftChild

    // Match can only be found to the right of this node

    if (CurrentNode.max.time < t)
        if (|t-CurrentNode.max.TimeIndex| < |t - BestMatch.TimeIndex|)
            BestMatch = CurrentNode.max
        CurrentNode = CurrentNode.rightChild
end while
```

Figure 4.16: Retrieving a Value from the Instrument Data

4.8.5 User Input

The user can interact with the environment in several ways. The user can change the behavior of objects within that environment, or change the environment itself.

The SANDBOX gives the user a generic way of choosing instruments, while the instruments themselves will be database specific, or from a set of general instruments. The SANDBOX provides a generic way of choosing sites while the environment those sites reside in will be database specific. The SANDBOX provides a generic way of choosing times from a set of general options.

The user primarily interacts with the environment by placing and moving instruments. Where the user can place the instruments is determined by the space mapping function and the location of the sites. A menu, or pallet, of virtual instruments in the virtual environment allows the user to see the instruments, identify them, and physically grab them off the pallet, and place them at a site in a very natural way.

The user also interacts with the instruments by changing the settings on the virtual instruments. Depending on the instrument the user could set the range of values to be displayed, or the set of values to be displayed. The user manipulates graphical option menus for each of the instruments which in turn modify the parameters of the display filters.

The user also interacts with the instruments by choosing the times that the instruments are in operation. As well as the pallet of instruments in the virtual environment there needs to be a way for the user to choose appropriate time ranges (e.g. through a clock, or a calendar.)

As well as being able to choose which sites to place instruments at, the user can change the behavior of virtual space (how big it is, or where it is.) The user manipulates graphical space menus which in turn modify the parameters of the space mapping function.

As well as being able to choose the times that the instruments are active, the user can change the behavior of virtual time itself (speeding it up, slowing it down, or stopping it.) The user manipulates graphical time menus which in turn modify the parameters of the time mapping function.

4.8.6 Data Output

Once the user has placed all of the appropriate instruments into the virtual environment, set the appropriate times, and adjusted the appropriate settings on the instruments the user can output the data.

All of the instrument data that is displayable (i.e. instrument data that is within the displayable range for a given instrument) is output. Each instrument class is taken in turn and given its own external file. Each instrument at a site is taken in turn. As the instrument data for each instrument is stored in a binary search tree ordered by time, with each node internally sorted into ascending order by time, a depth first inorder search of the tree will produce the data in ascending order by time.

4.9 Creating the Interface

In order to use the SANDBOX to interact with a scientific database, the designer must go through the following sequence of actions:

1. create the space and time mapping functions.

2. use the existing meta-data in the database to create a familiar environment.

3. choose the instrument classes, access functions, and filler functions from an existing library, or create them as needed.

4. create the linkages as needed.

The first two steps are necessary to set up the virtual environment. After that, the instruments can be created and linked to the database as they are needed.

4.10 Conclusions

In this chapter I have described the general design of the SANDBOX. I have shown how the instruments are created and then linked to a scientific database; how the data is stored and accessed; and how the user interacts with the interface. The next chapter will show an implementation of this paradigm.

Chapter 5

SANDBOX Implementation

5.1 Introduction to the SANDBOX Implementation

I have implemented a prototype of the SANDBOX in C++ [KP90, Poh94] and GL [McL91] under Unix [Ste92] using the CAVE [CSD93, Ele94] virtual reality theatre at the Electronic Visualization Lab at the University of Illinois at Chicago as shown in Figure 5.1.

I implemented this prototype on a subset of the FIFE scientific database. In this reenactment of the FIFE field experiments, the user is surrounded by an elevated 3D plane showing the 20km by 20km square site, a pallet of instruments to choose from on the right wall, and a calendar to choose dates from on the left wall. The 3D plane initially shows an enhanced satellite view of the experiment site showing roads, and rivers. See Figure 5.2.

Figure 5.3 shows an experiment in progress in the SANDBOX. The user has placed a water beaker, a thermometer, and a wind-sock into the virtual environment. The water beaker shows 6 millimeters of rainfall has collected at its site. The thermometer shows a temperature of 22 degrees Celsius at its site. The wind-sock shows a northerly wind blowing at 2 meters per second at its site.

Section 5.2 discusses how the instruments are chosen, placed, and modified. Section 5.3 discusses how the wand is used. Section 5.4 discusses the graph wall. Section 5.5 discusses the 3D plane. Section 5.6 discusses the calendar. Section 5.7 discusses how meta-information is viewed. Section 5.8 discusses how external information is

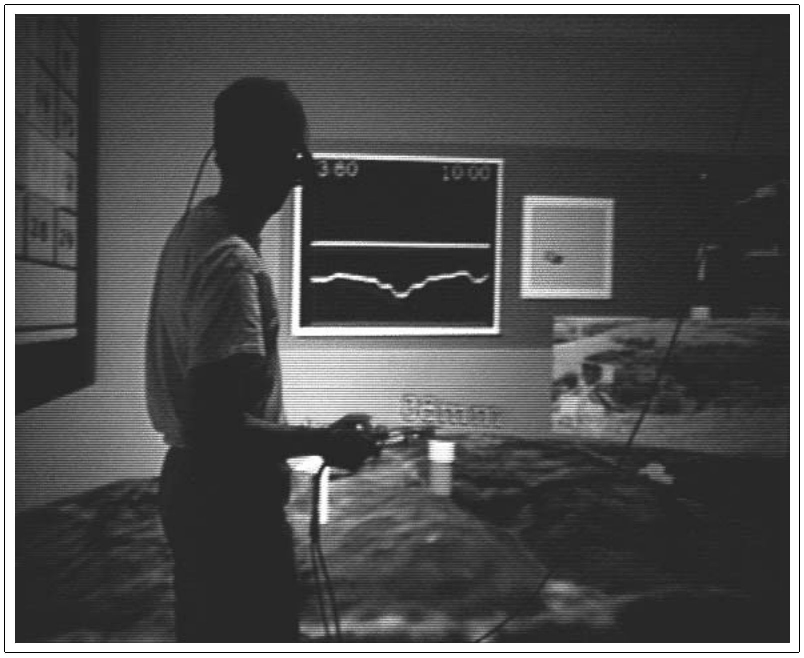


Figure 5.1: Photograph of SANDBOX in the CAVE

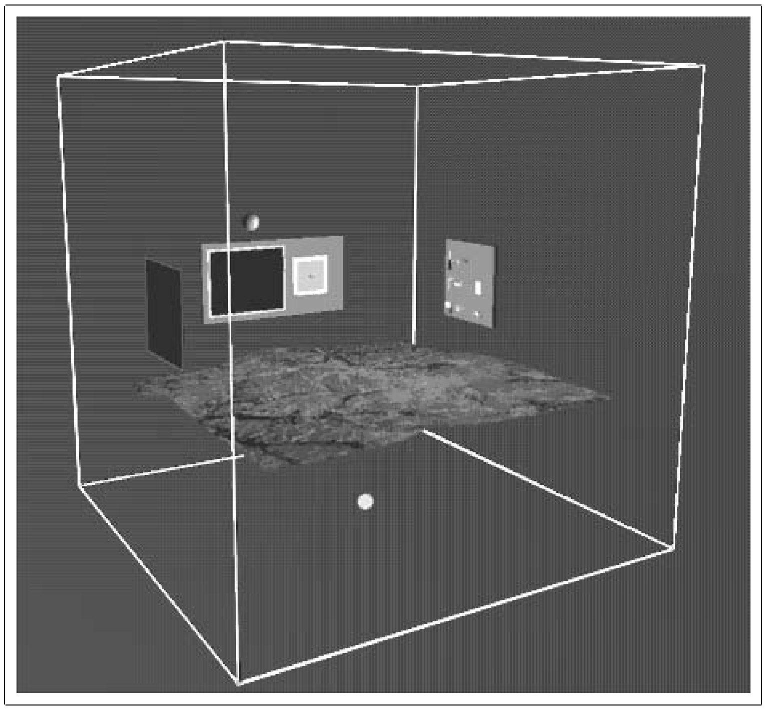


Figure 5.2: The SANDBOX in the CAVE

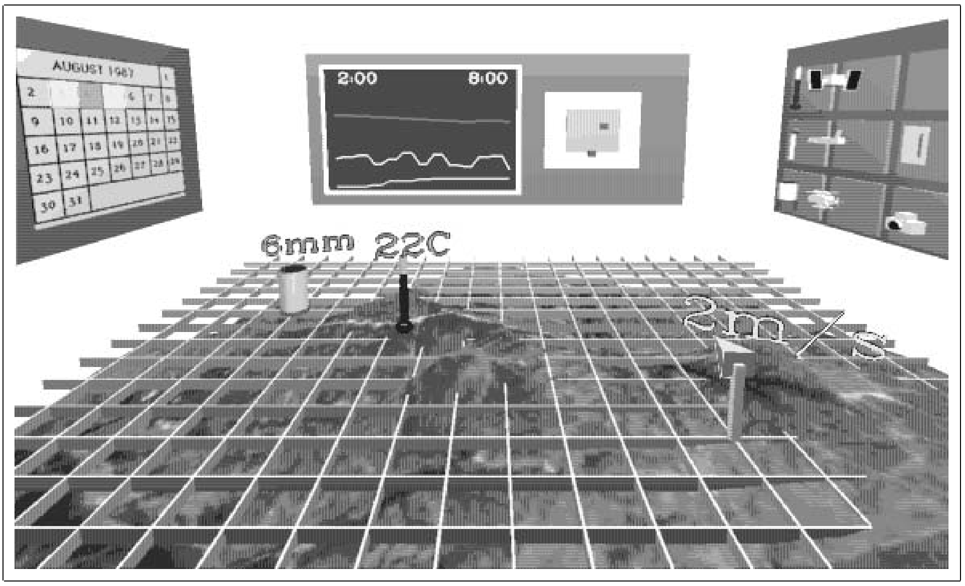


Figure 5.3: Overview of the Interface

viewed. Section 5.9 discusses a desktop version of the interface. Section 5.10 compares usage times in the SANDBOX to the real world. Section 5.11 compares the SANDBOX to more traditional access methods. Section 5.12 discusses reaction to the SANDBOX implementation. Section 5.13 gives some conclusions about this implementation. Section 5.14 discusses enhancements to the SANDBOX.

5.2 The Instruments

The user interacts with the data in the scientific database through the instruments.

5.2.1 The Instrument Pallet

Figure 5.4 shows a close-up view of the instrument pallet. Some of the instruments (thermometer, wind-sock, and water beaker in the left hand column) are linked to attributes in the relational database. Some (LANDSAT satellite, airplane, helicopter in the center column) are linked to graphics files [SLN+92c]. Some (notepad, camera, in the right hand column) are linked to meta-data [SNL+93]. The instruments on

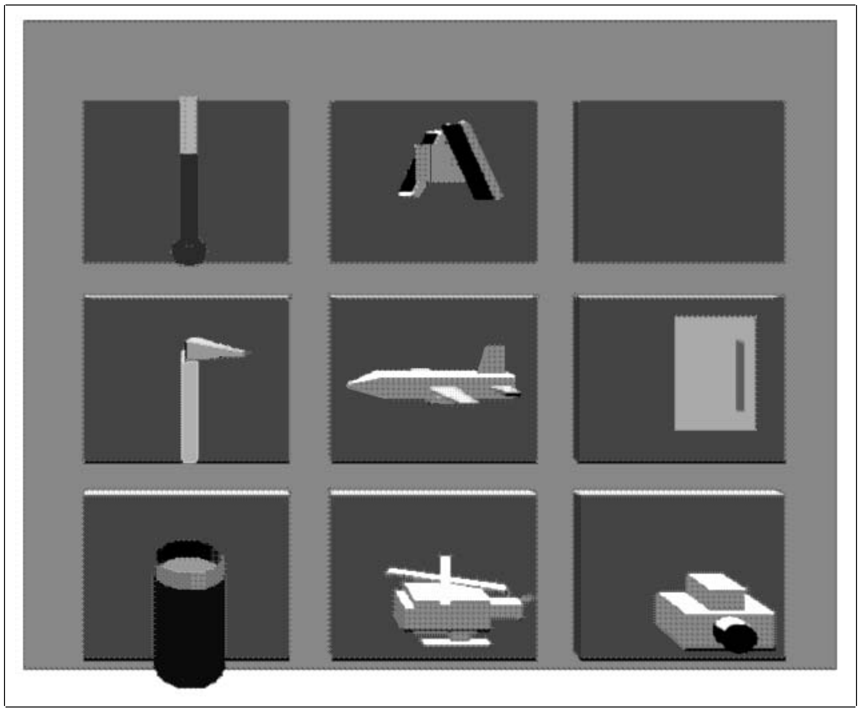


Figure 5.4: The Instrument Pallet

the pallet are 3D and animated (the beaker fills with water, the solar panels on the satellite spin, etc.) to improve their recognizability. The instruments have obvious affordances [Nor88]. All temperatures are measured with the thermometer, all rainfall amounts are measured with the beaker, no matter how they are stored in the database.

5.2.2 Placing Instruments

As the user moves the wand over an instrument, the area around it highlights to yellow showing the user that the instrument can be selected. The user chooses an instrument by moving the wand to the instrument on the instrument pallet, pressing a button on the wand, and carrying a three dimensional copy of the instrument off the pallet. All of the sites where the user can place that instrument are then highlighted in yellow on the 3D plane as shown in Figure 5.5. When the user moves the wand (and the virtual instrument) over one of the available sites it highlights in red as shown in Figure 5.6.

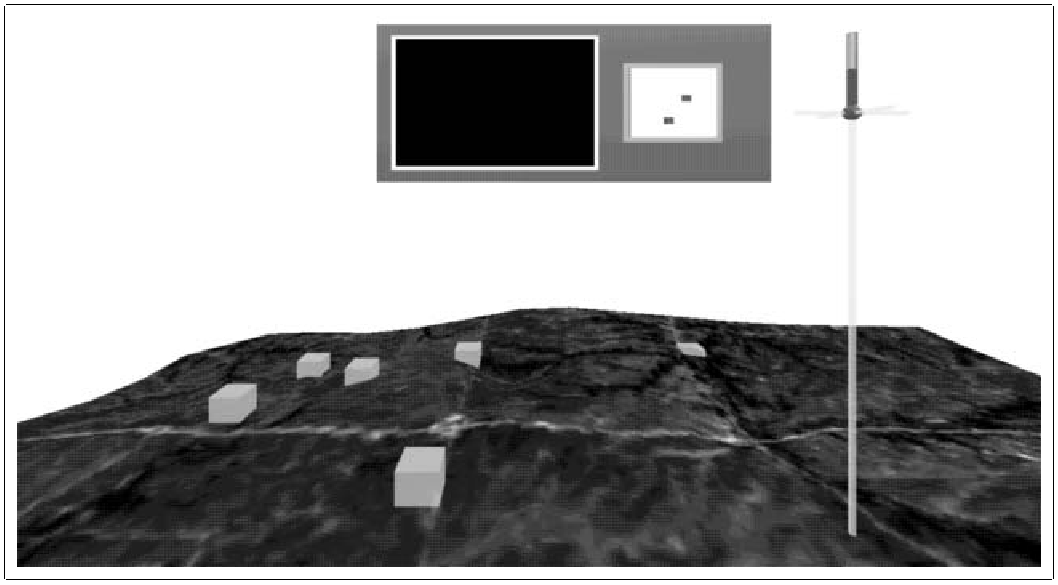


Figure 5.5: Available Sites Highlight in Yellow Based on Held Instrument

The user places an instrument at this site by pressing a button on the wand as shown in Figure 5.7. The instrument immediately begins taking measurements.

This gives the user a very natural method of placing instruments as the user can literally pick up an instrument, carry it over to a site, and place it there. The user directly manipulates the objects in the virtual environment [Sch83]. There is a continuous visual representation of objects and actions; physical actions replace complex syntax; the impact of an action is immediately visible. This allows a novice user to quickly begin using the system.

The user can move instruments from one site to another using a similar method. When the user moves an empty wand over a site with an instrument, the site highlights in yellow. The user picks up the instrument by pressing a button on the wand. As before, all of the sites where the user can place that instrument are then highlighted in yellow on the 3D plane.

As it would require a lot of bending down if the user had to physically place the virtual instrument onto the 3D plane, I have implemented a limited form of

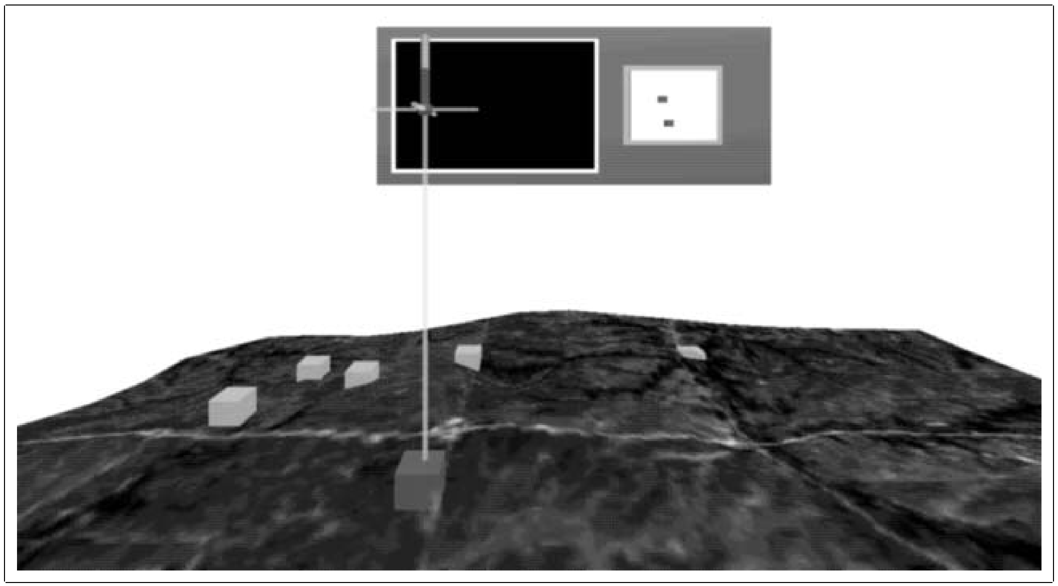


Figure 5.6: Selected Site Highlights in Red When Wand is Overhead

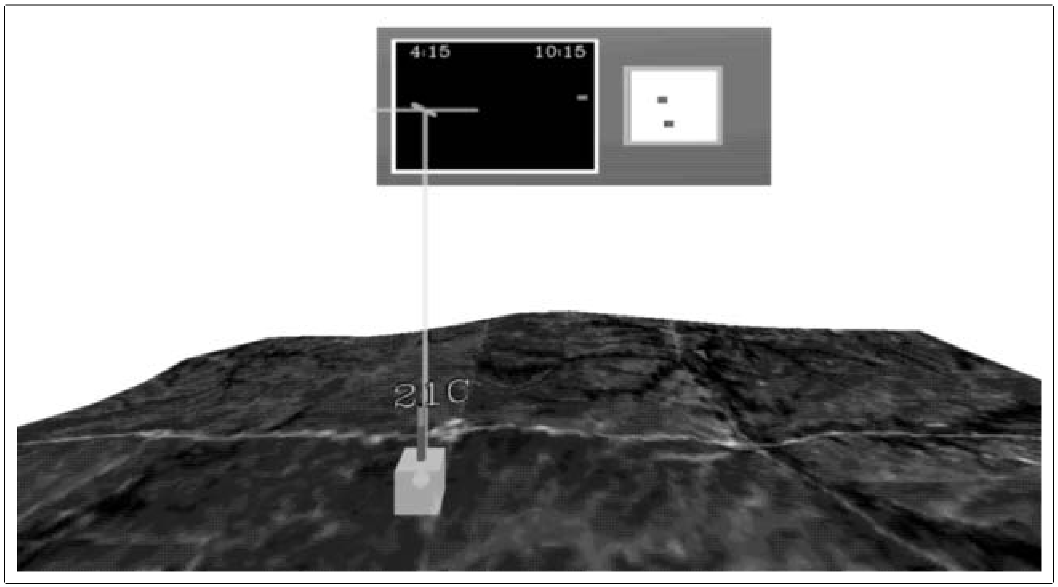


Figure 5.7: Instrument Placed at Site Begins Operating

raycasting. With raycasting, a virtual beam emanates from the wand and the user can select anything in the path of the beam. In my implementation a vertical ray extends downward from the wand. The user can select a site by moving the wand over the site at any altitude, allowing the user to 'drop' an instrument onto the 3D plane or 'grab' an instrument off the 3D plane from high above it. This allows the user to make selections at a distance, and has been shown to reduce arm fatigue [JE92, SGLS93].

While the amount of data in this database is very large, the number of sites for each experiment is typically small (usually around 10 to 15 sites), giving sparse coverage of the total experiment area. Because of this, conventional visualization techniques can not be employed (i.e. there is not enough information to draw meaningful colour mapped surfaces over the 3D plane.)

5.2.3 Instruments Under Operation

Once the instrument is placed at a site, it begins to operate. The mercury level in the thermometer rises and falls with the temperature. The water level in the beaker rises and falls with the rainfall. The orientation of the wind-sock changes with the direction, and speed of the wind. This allows the user to see how the measurements inter-relate (e.g. the mercury level dropping in a thermometer as a beaker begins to fill.)

The user can also hear the instruments. A beaker makes a 'drip' sound when its water level rises. The faster the water level is rising, the louder the 'drip.' A wind-sock makes a 'whoosh' sound. The stronger the wind, the louder the 'whoosh'. A thermometer makes a 'cicada' sound. The higher the temperature, the louder the 'cicada.' (The cicadas do double duty as crickets during the night.) This way, a

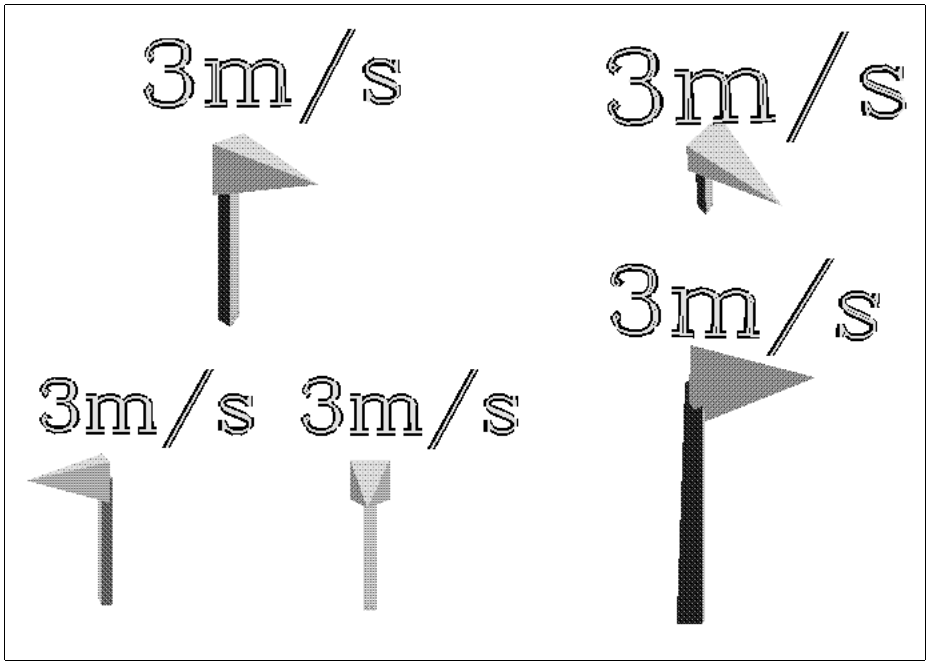


Figure 5.8: Quantitative Values Rotate to Follow the User

user doesn't have to watch all the instruments all the time. An instrument draws attention to itself when there is a change in the value it is monitoring. A user busy placing a thermometer in one corner of the experiment, will hear the sound of the rain beginning to fall. Since the sound is 3D the user can easily turn around and locate which instrument is calling attention to itself.

If the user needs to see a record of how the measured values change over time, the values can be displayed in a graph on the graph wall. If the user requires quantitative numeric, as well as qualitative graphical values, they can be displayed above each instrument. These quantitative values rotate horizontally and vertically to follow the user. They are always readable no matter where the user is standing in the CAVE as shown in Figure 5.8.

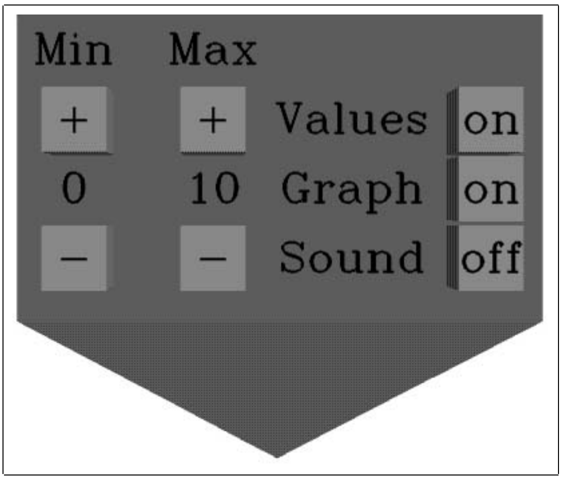


Figure 5.9: Menu for an Individual Instrument

5.2.4 Changing Settings on Instruments

The user can change the settings on the virtual instruments using the menu shown in Figure 5.9. The user moves the wand over to the instrument, and then brings up the menu using the wand button. The user can set the minimum and maximum values displayed by the instrument (and at the same time change the minimum and maximum values displayed on the graph for this instrument.) The user can turn on or off whether the instrument displays quantitative values overhead, whether its values are shown on the graph, and whether the instrument makes sound. Each instrument maintains its own individual settings.

The interface employs an object-oriented paradigm. Existing virtual reality systems [SLGS92, SGLS93] allow for the creation of sliders, menus, and button panels. These are simply transferences of 2D systems into the 3D virtual environment where operations on objects are performed using a separate detached interface control panel. In our approach each object in the virtual environment maintains its own state, and has its own interface control panel. This reduces user errors during interaction by giving users access only to those functions which are appropriate to the object [Nor88].

5.3 The Wand

The current position of the wand in the cave is shown by a translucent 3D cross as shown in Figures 5.5, 5.6, and 5.7. In the CAVE the position of this 3D cross should correspond to the actual position of the physical wand, but due to errors in the tracker this position may be slightly off. The 3D cross allows the user to see where the VR system believes the wand actually is. This is also useful in the desktop version of the SANDBOX where there is no physical wand within the virtual environment.

The SANDBOX only uses two of the three buttons on the wand. One button is used for making selections, and the other is used to display menus. The selection button allows the user to pick up or drop instruments. It is used to select and deselect dates from the calendar. It is used to choose options from menus. The menu button is used to display or hide the various menus in the system.

The user receives audio feedback (a 'thunk' sound) when making selections with the wand. Whenever the user grabs an instrument, drops an instrument, chooses a date, activates a menu, or deactivates a menu the user receives this audio feedback as well as visual feedback.

5.4 The Graph Wall

Figure 5.10 shows the graph wall. The graph wall of the CAVE shows the user's position on the right, and the graphs on the left. The user's position (marked by a small blue square) is shown relative to the 3D plane (a large green square) and the physical CAVE boundaries (a large white square.) The position of the wand (marked by a small red square) is similarly shown.

The user can tell at a glance whether the 3D plane is entirely inside the CAVE walls, and entirely accessible, or whether parts of it lie outside the walls and are

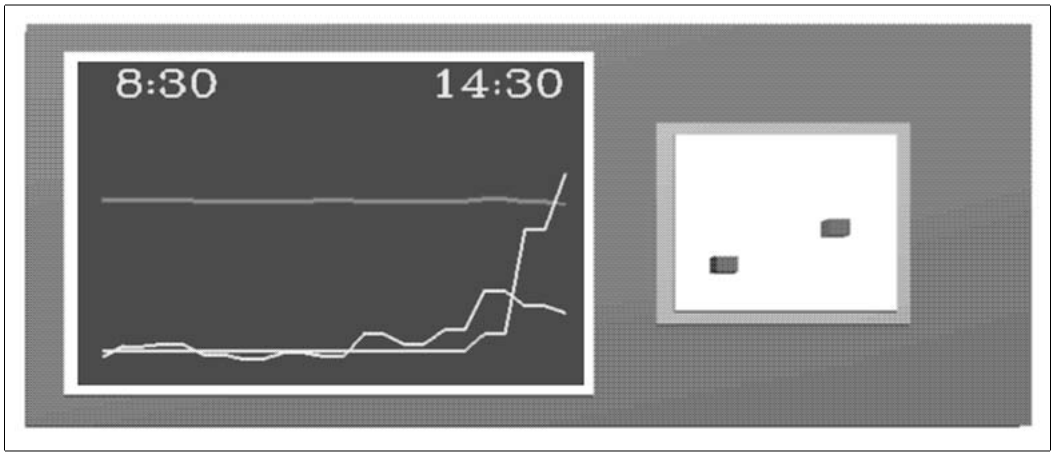


Figure 5.10: The Graph Wall

currently unaccessible. If the user loses track of the wand, this will make it much easier to find. For example, Figure 5.10 shows the 3D plane extending beyond the edges of the CAVE. The user is standing in the left, back corner and the wand is right of center in the CAVE.

The graph shows values over the last six hours. The right edge of the graph shows the current time of day, and the values at the current time of day. The left edge of the graph shows the time six hours previous, and the values six hours previous. All of the valid instrument readings within the last six hours are connected by lines. The colour of the lines matches the colour of the instruments (red for the thermometer, blue for the beaker, yellow for the wind-sock) for ease of identification.

The minimum and maximum values on the graph are set to match the minimum and maximum values each instrument is set to display. For example, if a thermometer is set to display values from 10 degrees to 30 degrees then the graph values range from 10 to 30. Watching the values change over time the user can see trends that are not apparent from the data itself.

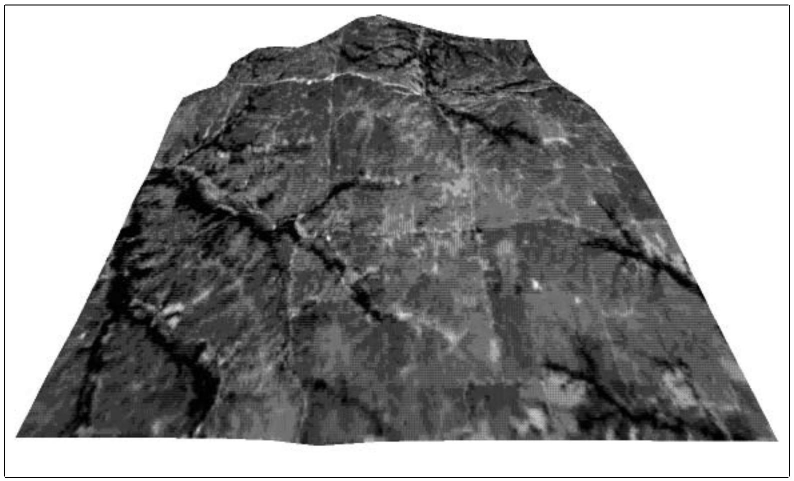


Figure 5.11: The 3D Plane

5.5 The 3D Plane

Figure 5.11 shows the 3D plane. The user can alter how the 3D plane is displayed. The user can enlarge or shrink the 3D plane. The entire 3D plane can be displayed within the CAVE, or parts of it can lie outside the walls of the CAVE, depending on whether the user wants an overview of the entire experiment, or a close-up view of a certain area. The user can turn the grid lines on to break the 3D plane up into kilometer square blocks, or turn them off to get a better view of the landscape. The user can raise or lower the 3D plane. The 3D plane could be placed on the floor to look down at it from high above, or at waist height to get a better sense of the topography, or above the user's head to look at the terrain from below. The menu for the 3D plane is shown in Figure 5.12.

The boundaries of the 3D plane match the boundaries of the actual environment. Information on the latitude, longitude, and elevation of each site in the actual environment was recorded. Interpolating the elevation information from the various actual sites, I constructed the 3D plane in the virtual environment. Kansas is rather flat, so I enhanced the elevation information to allow the user to see the variation in

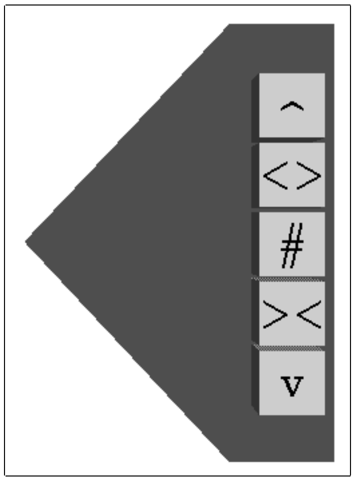


Figure 5.12: The Plane Menu

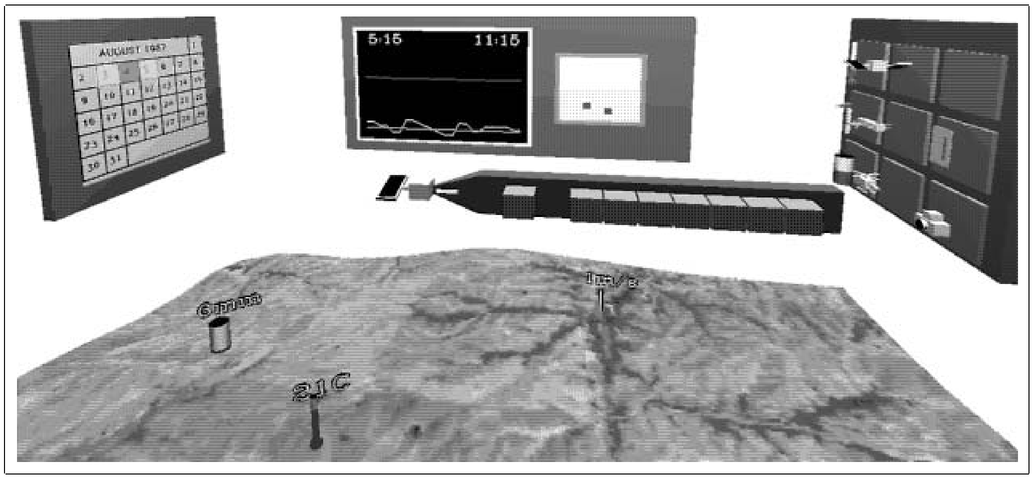


Figure 5.13: Viewing the Experiment in a Different Light

elevation more clearly.

When the user grabs the satellite instrument and places it in the sky above the 3D plane, the user can choose which band to view the 20km by 20km landscape in. The user can choose to see the landscape in visible light, infra-red, or the 5 other LANDSAT bands. LANDSAT photographs from the database are mapped onto the 3D plane as shown in Figure 5.13 where the user is viewing the landscape in the infra-red. The menu for the satellite is shown in Figure 5.14.

In the actual scientific database the user must refer to a site using its site ID number. In our system the user can see where the sites are located. If the user wishes

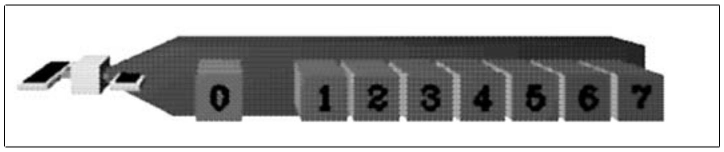


Figure 5.14: The Satellite Menu

to measure the temperature near a river, or at high altitude, or where the satellite shows lots of activity in the infra-red, the user can see exactly where to place the instrument by looking at the 3D plane. The graphical information is integrated with the numeric information. In the actual scientific database the user would have to integrate this information manually.

The greyscale satellite photograph must be processed before it can be texture-mapped onto the 3D plane. As the CAVE is rather dim, I enhance the contrast of the satellite photos before storing them in local memory. Only the center of the photograph matches the area of the 3D plane, and that part of the photograph must be rotated 10 degrees clockwise to match the orientation of the 3D plane. A sample satellite photograph is shown in Figure 5.15 with the area corresponding to the experiment 3D plane highlighted. As all of the satellite photographs were taken from the same location in space, this processing is the same for all of the photos. In the actual scientific database the user would have to do this processing for each satellite photo to match the photo to the experiment sites.

By grabbing the helicopter instrument and placing it in the sky above the 3D plane, the user can see information on soil moisture obtained from the air during the experiment. The satellite photographs are mapped onto the 3D plane itself. The soil moisture data collected from the air is coloured transparent blue and then mapped on top of the greyscale satellite photograph. This allows the landscape to show through the soil moisture information. The graphical information from space can be combined

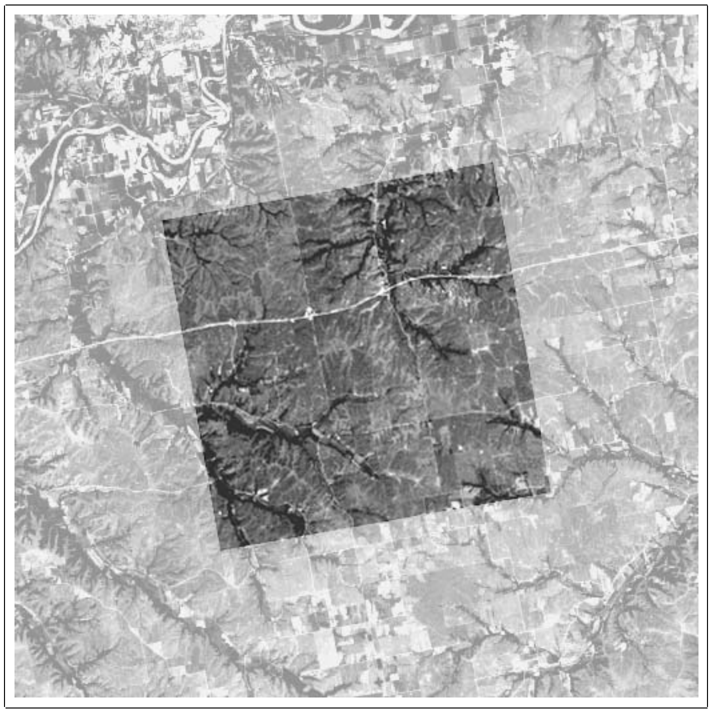


Figure 5.15: An Original Satellite Photograph

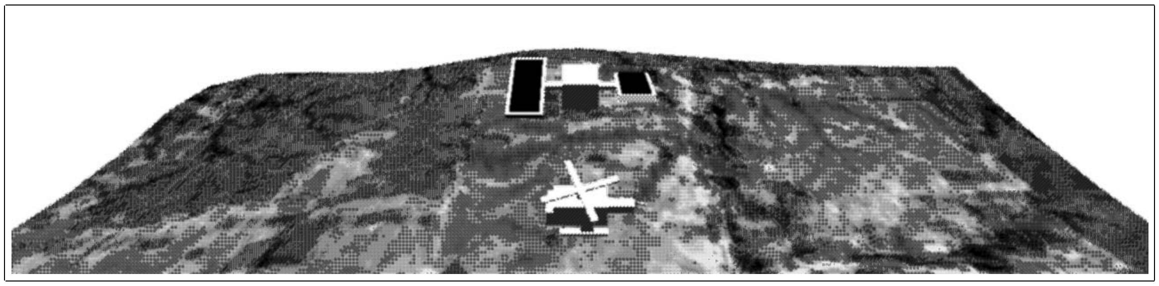


Figure 5.16: Integrating Multiple Views of the Landscape

with the graphical information from air as shown in Figure 5.16.

5.6 The Calendar

The user selects days from the calendar by clicking on them with the wand. Virtual time only passes when at least one day is selected. The current virtual day cycles through the selected days. Selected days are shown highlighted in blue; the current day is shown highlighted in red. As the user moves the wand over a day it highlights in yellow (including current day's blue being yellow highlighted to green, and the

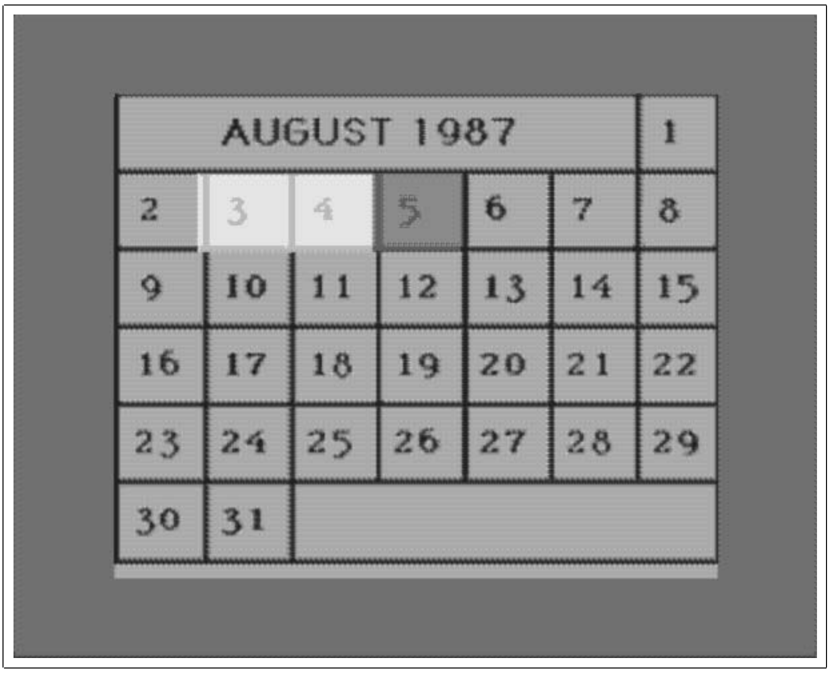


Figure 5.17: The Calendar

selected day's red being yellow highlighted to orange) to show that the day can be selected or deselected. Figure 5.17 shows a close-up view of the calendar where the 3rd, 4th, and 5th if August 1987 are selected for display and the current day is the 5th.

A virtual sun orbits the 3D plane rising out of the east each morning at 6am, and setting in the west at 6pm. As the sun dips below the western horizon, the moon rises in the east and continues to orbit the 3D plane opposite the sun. While the Earth's moon does not exhibit this behavior, it gives the user better control over their virtual environment. By grabbing the sun or moon with the wand, the user can adjust the behavior of time, speeding it up, slowing it down, or stopping it. The menu for the Sun/Moon is shown in Figure 5.18.

The sky changes color as the virtual day progresses, changing from black at midnight, to purple at dawn, to bright blue at noon, to purple again at dusk, and back to black at midnight. This gives the user additional temporal feedback. The user can

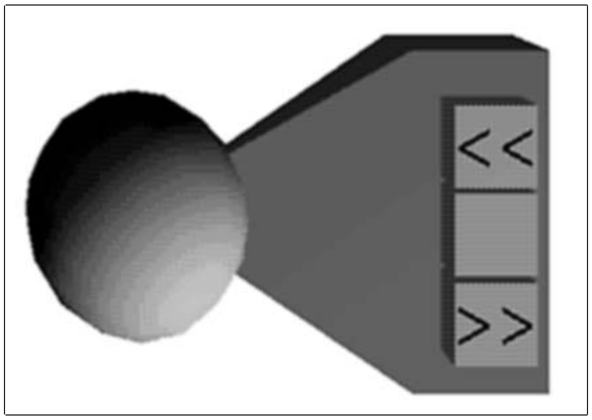


Figure 5.18: The Sun/Moon Menu

clearly see and hear how the instruments behave as the hours pass by.

5.7 Viewing Meta-Information

The user views textual meta-data (e.g. site information, notes) with the notepad, and graphical meta-data (e.g. photographs taken a site) with the camera. If the user wishes to see information about a site (its latitude, longitude, and elevation) the user grabs the notepad instrument and places it at a site. A page with the text then appears above it. If the user wishes to see a photograph taken at a site, the user grabs the camera instrument and places it at a site. The picture of that site then appears above it. The meta-data is integrated with the numeric information and the graphic information. In the actual scientific database the user would have to integrate this information manually.

Other possible meta-data instruments include a microphone instrument to listen to recorded audio and a movie projector instrument to play back video. Like the quantitative values in Figure 5.8, the notepad, camera and movie projector instruments rotate so the user can see their values from anywhere in the virtual environment. Figure 5.19 shows an assortment of meta-data instruments: Notepad, Camera, Mi-

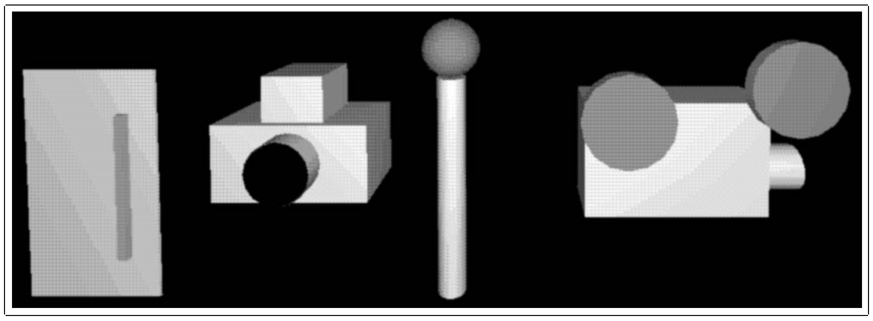


Figure 5.19: Meta-data Instruments

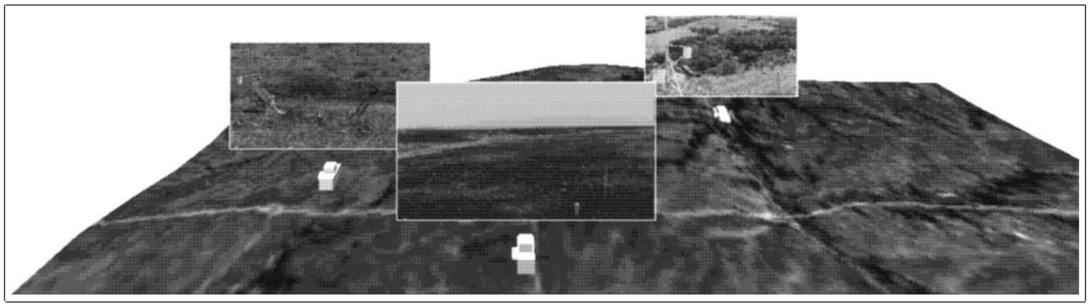


Figure 5.20: Viewing Meta Information in the SANDBOX

crophone, and Movie projector. Figure 5.20 shows three cameras displaying their respective site photographs.

Figure 5.21 shows four instruments placed in the virtual environment: a beaker measuring rainfall, a wind-sock measuring wind speed and direction, a thermometer measuring temperature, and a camera displaying a photograph taken of a site. The beaker, wind-sock, and thermometer have their current values displayed graphically, audibly, and numerically. Their values over the last six hours are shown in the graph. The user is simultaneously seeing numeric data from the database visualized by the instruments, graphical data showing the features of the 3D plane, and meta-data showing an actual photograph of a site.

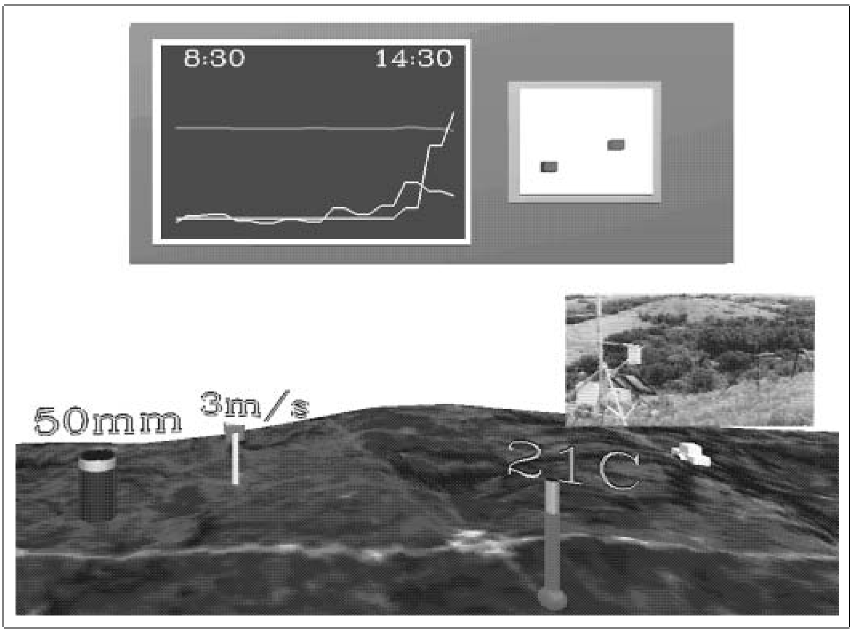


Figure 5.21: The Instruments Giving Feedback

5.8 Viewing External Information

All of the information that an investigator needs to run their experiments may not be in the scientific database. Since the user is not isolated from the real world while interacting with the virtual world inside the CAVE, the user can bring items into the virtual environment. The user may want to bring in a list of experiments to run, or a map of sites to examine.

Unfortunately the CAVE must be kept dark so that the projected images appear bright. This makes it difficult to see anything brought into the CAVE. To alleviate this problem, the user can turn on a virtual spotlight in the SANDBOX as shown in Figure 5.22. This spotlight illuminates the area surrounding the user. The spotlight follows the user's movements around the CAVE, allowing the user to see and read whatever they are holding while interacting with the virtual environment.

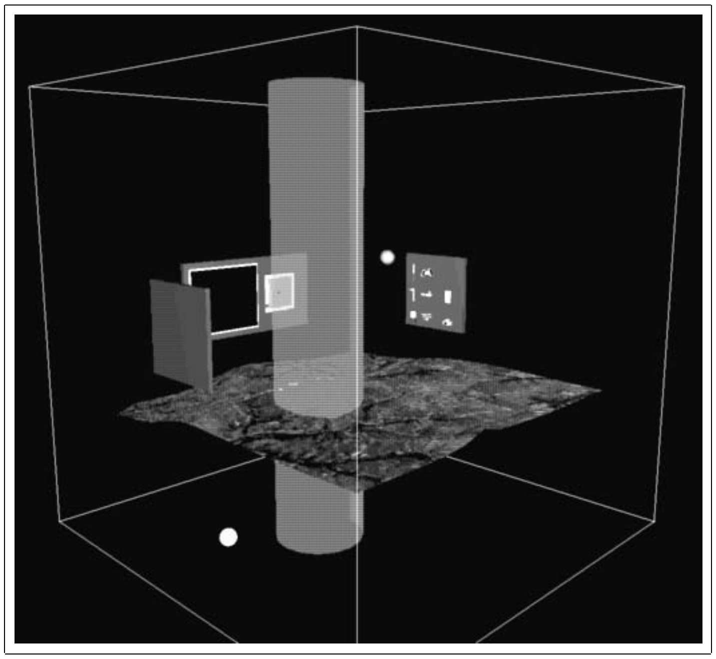


Figure 5.22: Spotlight in the CAVE

5.9 Desktop Version

This method of extracting data through recreation of experiments, and the integration of the meta-information does not rely on virtual reality. While virtual reality provides more of a direct manipulation immersive environment, these same techniques could be applied to more common, and less costly, hardware.

5.9.1 2D Desktop Version

The original proof of concept demo of the SANDBOX is shown in Figure 5.23. This was a hypercard stack running on a Macintosh Powerbook. The user could place up to four thermometers and water beakers into the environment and choose any dates in August 1987 to display. The instruments showed their values graphically as well as numerically.

This interface relied on a standard two dimensional desktop interface, making three dimensional features such as the elevation of the plane hard to visualize. The

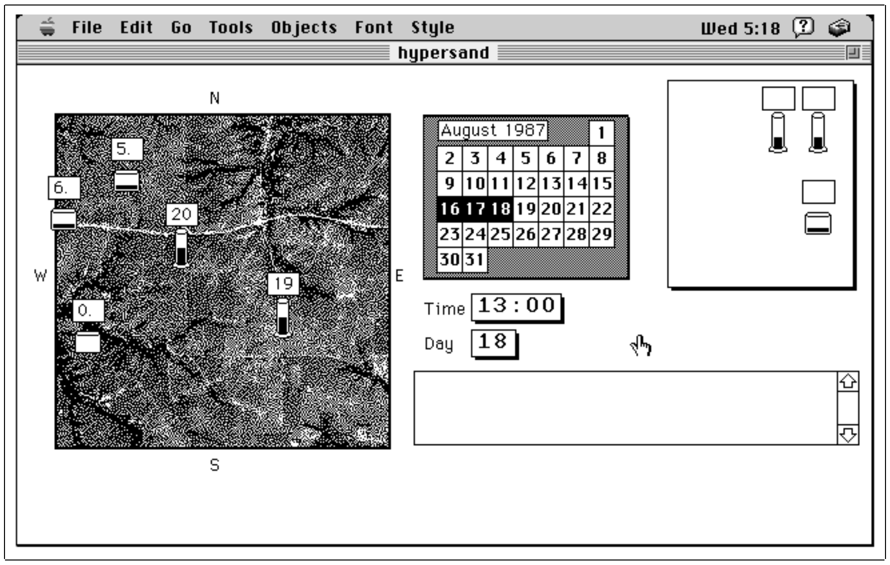


Figure 5.23: Original Desktop Hypercard Prototype

user has a static view of the experiment sites. In this case the plane is viewed from directly above hiding the elevation information. Head tracking and 3D displays have been experimentally shown to improve the user's interactions with virtual worlds [Akk93, AB93, MHR91, Wic91].

5.9.2 3D Desktop Version

The code that comprises the SANDBOX prototype is not limited to the CAVE. This same program can be run in a desktop environment that retains the three dimensional interaction of the CAVE. Unfortunately this desktop interface is not immersive, and the user has less direct manipulation than in the CAVE.

The SANDBOX code was written so that it could adapt to the number of available walls in the CAVE. This way it could be run in CAVEs with fewer than three walls and a floor. For example, Figure 5.24 shows the SANDBOX running in the CAVE with only two walls and a floor. For the desktop version the calendar and the instrument pallet are moved from their respective walls to join the graph wall, and the 3D plane is moved up to appear on the wall instead of the floor. See Figure 5.25.

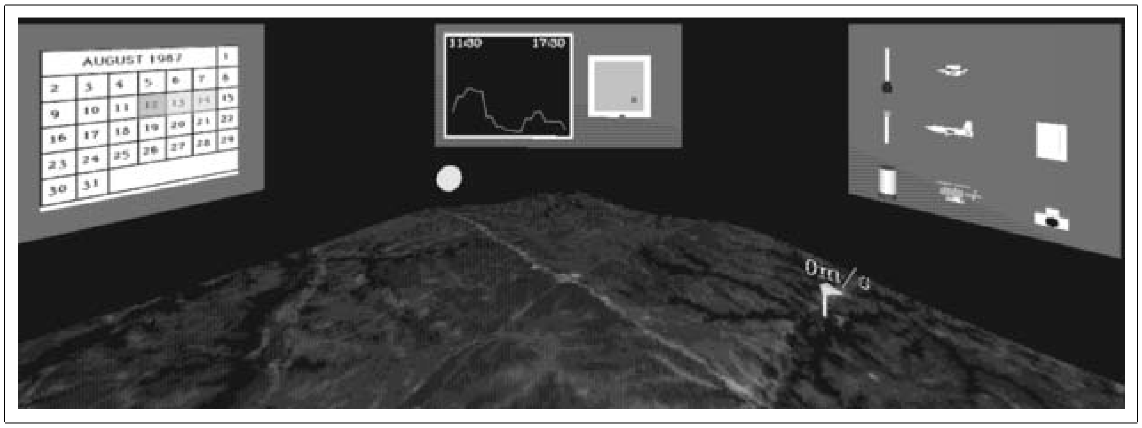


Figure 5.24: Two-Wall SANDBOX

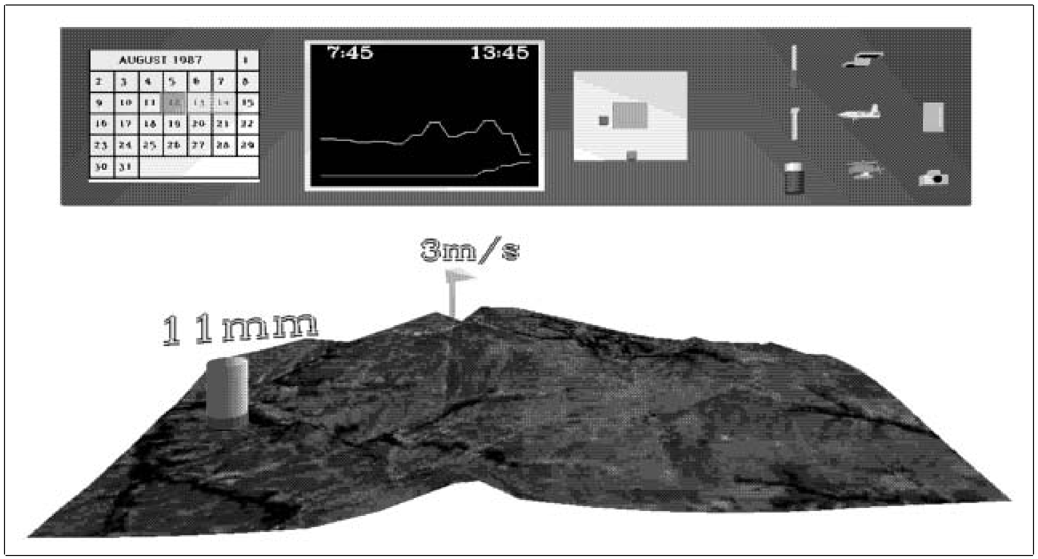


Figure 5.25: Desktop Version of the SANDBOX

The SANDBOX code could also be adapted to a Head Mounted Display or a BOOM, however the affordable Head Mounted Displays do not have enough resolution to handle the detail involved in the SANDBOX, and a colour BOOM is excessively expensive. Both of these hardware platforms also isolate the user from the real world making it impossible to see external information while interacting with the virtual environment. A 'Fish Tank' desktop system would probably be the best alternative for those wanting a currently affordable hardware platform.

5.9.3 Desktop Hardware Requirements

Table 5.1 shows the average frames per second generated by several different hardware platforms running the desktop version of the SANDBOX. Faster platforms are obviously better than slower ones, but special graphics hardware is also necessary for the interface to maintain an acceptable frame rate. This special hardware could include a hardware z-buffer to speed up the drawing of overlapping polygons, or hardware to speed up the drawing of textures onto polygonal surfaces.

The frame rate is also important as it relates to the lag between a user performing an action, and the user seeing that action reflected in the virtual environment. This lag time includes the time it takes the trackers to notice a change, and the time to draw the next frame to reflect this change. If this lag time is too long the user will feel disoriented as actions no longer have immediate consequences.

The Onyxs were able to generate frames faster than their display hardware could display them. The VTX was able to maintain acceptable frame rates with texture mapping on or off. The Indigo^2 Extreme and the Indigo Elan were able to maintain acceptable frame rates with texture mapping off, but not with texture mapping on. The Personal Iris and the two Indys were not able to provide acceptable frame rates.

Machine	Hardware Z-buffer	Hardware texture mapping	Frames/sec w/o texture mapping	Frames/sec w texture mapping
Onyx/4 RE^2	yes	yes	>60	>60
Onyx/2 VTX	yes	yes	>30	>30
VTX	yes	yes	10	9
Indigo^2 Extreme	yes	no	20	2
Indigo Elan	yes	no	10	2
Personal Iris 4D/35	yes	no	5	-
Indy SC 24	no	no	4	1
Indy PC 8	no	no	2	1

Table 5.1: Comparison of Desktop Platforms

In the current implementation of the SANDBOX interface to the FIFE database, the time taken to texture map the satellite photograph onto the 3D plane is a very large amount of the time it takes to draw a single frame. Texture mapping is also used to display the site photographs. Texture mapping is a computationally intensive operation, but a necessary one, if we want to integrate the graphical data, and meta-data into the environment.

Other scientific databases with large amounts of graphical information, or those needing a graphically enhanced environment would have similar needs. Therefore the hardware chosen to run this desktop version should be able to perform real-time texture mapping to keep the frame rate above 15 frames per second.

One option is to give the user control over the texture mapping. Users of hardware without real-time texture mapping could turn texture mapping off when it wasn't necessary, and improve their interaction with the system. When they need to look at the landscape, or a site photograph they can turn texture mapping on. This increases the burden on the user, but makes the system available to a wider audience.

Another option is to use a form of progressive refinement. Whenever the user

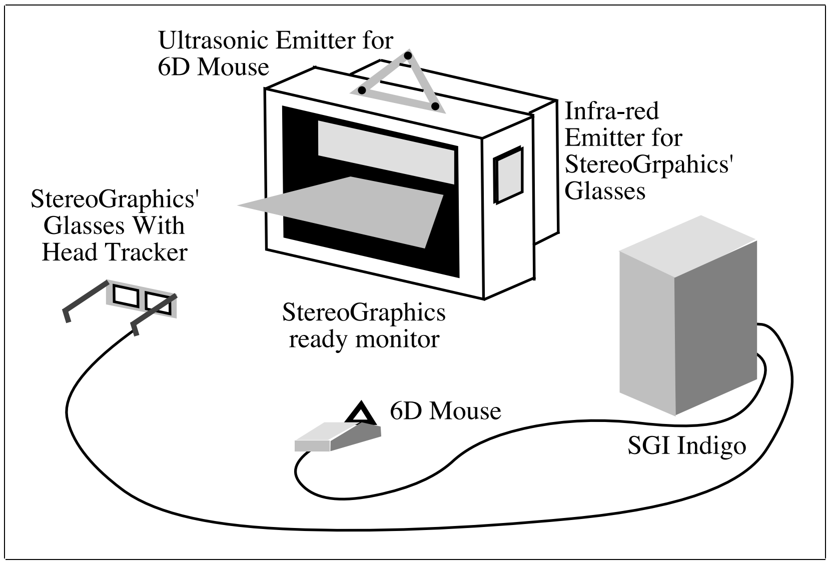

Ultrasonic Emitter for 6D Mouse
Infra-red Emitter for StereoGrpahics' Glasses
StereoGraphics' Glasses With Head Tracker
StereoGraphics ready monitor
6D Mouse
SGI Indigo

Figure 5.26: Desktop Version Hardware

remains stationary, the environment is drawn in full detail using texture mapping. Whenever the user moves, or turns their head, the texture mapping is turned off to improve interaction speed.

The components of the typical desktop version are shown in Figure 5.26. They include the following hardware: an SGI Indigo^2 Extreme and StereoGraphics-ready monitor, a pair of StereoGraphics' CrystalEyes glasses with infra-red emitter, a head tracker, and a Logitech 6D mouse with ultrasonic emitter. The monitor and glasses allow the user to see the SANDBOX in 3D. The 3D plane appears to extend into and out of the monitor screen. The head tracker allows the scene to shift appropriately as the user moves their head, increasing the illusion of 3D. The 6D mouse replaces the wand in the CAVE allowing the user to interact with objects in the 3D environment. Holding the 6D mouse in the air the user can move it in all three dimensions (up/down, left/right, and in/out) as well as using roll, pitch, and yaw. In effect, this hardware set up brings one of the CAVE walls to your desktop as shown in Figure 5.27.

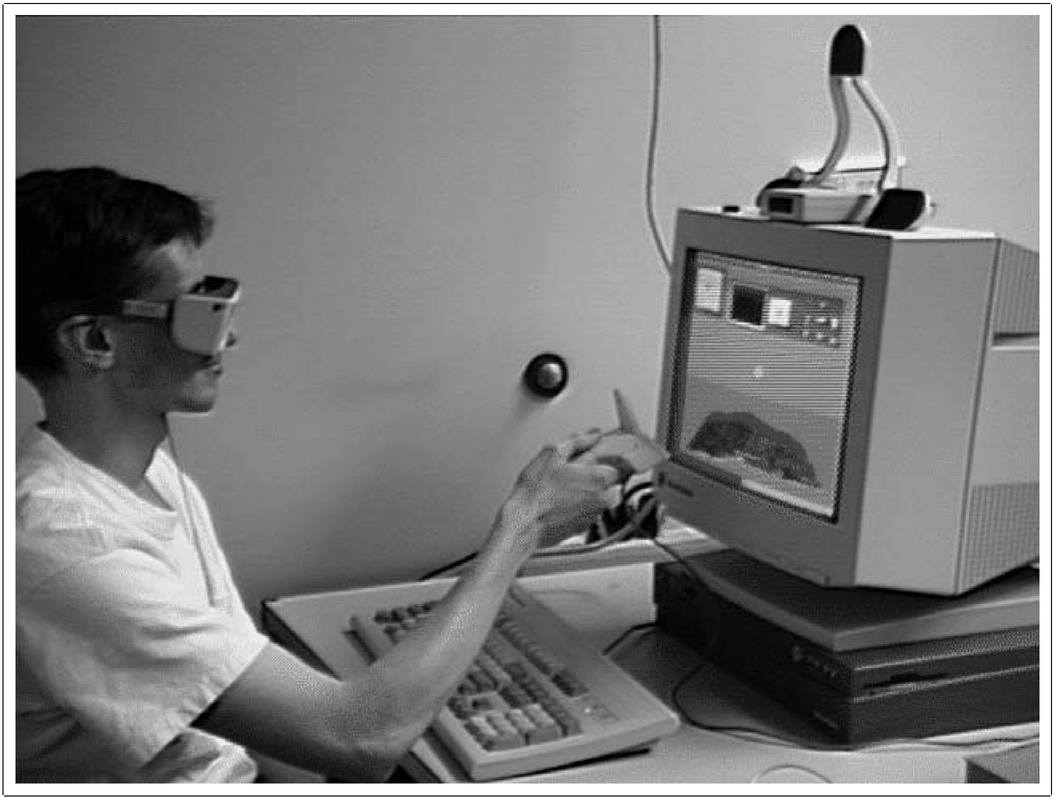


Figure 5.27: Photograph of Desktop SANDBOX

5.9.4 Comparison of CAVE and Desktop Versions

The CAVE and the desktop version each have their advantages and disadvantages. These are described in detail below and summarized in Table 5.2.

Immersion

The CAVE is immersive, the desktop version is not. A user sitting a comfortable distance away from the desktop version will have the monitor display filling 45 degrees of their field of view. A user standing in the CAVE will have the CAVE walls filling approximately 120 degrees of their field of view (limited by the glasses being worn, not the CAVE walls.) The CAVE user is also given the ability to turn their head and look around the environment; the desktop user must look at the screen. The CAVE is superior in this regard.

Visual Acuity

Visual acuity is most popularly measured using the Snellen fraction (e.g. 20/20 is average vision, 20/200 is legally blind.) When looking at the desktop version the user has a visual acuity of 20/45. When inside the CAVE the user has a visual acuity of 20/110 [CSD$^+$92]. While a user standing at the center of the CAVE can see approximately twice as many pixels as the desktop user, these pixels are spread over a much larger field of view causing the drop in visual acuity. The desktop version is superior in this regard.

Sound

Both the CAVE and the desktop version can generate 3D sound, that is sounds that appear to emanate from a particular point in space. The CAVE however has a much more natural mapping between the visual space and the audio space. If the sound is coming from an instrument on your right, you can turn to the right and see the object generating that sound. With the desktop system all of the visual cues are in front of you on the screen. Sounds can still come from behind you, but you must turn the virtual environment to see the object generating it. The CAVE is superior in this regard.

Cost

The desktop version is overwhelmingly cheaper than the CAVE. A desktop version of the system capable of doing real-time texture mapping would cost about $50,000 while the CAVE costs $1,000,000. There is a also range of possible hardware for the desktop version allowing the user to pay for only the performance which they need. The desktop version is more mobile, less prone to hardware failure, and built from more common hardware components. The desktop version is clearly superior in this regard.

Ease of Control

Even with a stereo display and devices such as a spaceball or a 6D mouse, the user of the desktop system can not have the same ease of control or degree of direct manipulation that the CAVE user enjoys. There is a more natural mapping between the users actions and their effects in the CAVE than in the desktop version. The CAVE is superior in this regard.

Ergonomics

The stereo glasses and head tracking equipment are virtually the same in the CAVE and desktop versions. The CAVE's wand is slightly less comfortable than a 6D mouse since the user must hold onto the wand, rather than being able to set it down on the desk. Desktop users also have the ability to sit down while interacting with the virtual environment. If the user is going to be conducting a large experiment, this extra comfort can be very important. The desktop version is superior in this regard.

Multiple Users

Both the CAVE and the desktop version allow any number of users. Each user wears a pair of stereo glasses, but only one user is tracked and carries the wand/6D mouse. It is slightly easier to accommodate multiple users in the CAVE since all the desktop user must sit or stand directly in front of the screen to see the stereo effect.

Criteria	Desktop	CAVE
Immersion		+
Visual Acuity	+	
Sound		+
Hardware Cost	+	
Ease of Control		+
Ergonomics	+	
Multiple Users		+
Ability to Integrate External Material	+	

Table 5.2: Comparison of Desktop and CAVE Implementations

Ability to Integrate External Material

As discussed previously, the CAVE gives the user a rich virtual environment to work in, but it is difficult to interact with external material into that virtual environment. The user can bring items into the CAVE but it may be difficult to see them. The user of the desktop system has easy access to external information through the books on their desk, their notes, their telephone, other computer programs, etc. If most of the information that the investigator needs to work with is in the database (in either numerical, textual, or graphical form) then the CAVE gives a much richer environment to interact with that information. On the other hand if the investigator needs access to a lot of external supplementary material, then the desktop version gives the user more restricted access to the virtual environment without limiting their access to the supplementary material.

5.10 Comparison of Usage Times

I employed several user interface evaluation techniques (heuristic evaluation, cognitive walkthroughs, and usability testing) [Mon84, JMWU91, Joh92] to determine the effectiveness of this interface. The amount of testing I could do in the CAVE itself was limited due to its geographic distance away and its popularity as a research tool. I was able to perform some testing in the CAVE itself, some testing of the desktop version of the CAVE, and some testing of the paradigm itself independent of its implementation.

If the SANDBOX is truly a 'natural' way of interacting with a scientific database then users should be able to interact with the SANDBOX as quickly and easily as they interact with objects in the real world.

To test this I ran several experiments using objects in the real world and virtual objects in the CAVE and in the desktop version of the interface. The CAVE environment has already been described. The desktop environment tested here made use of a regular 2D monitor and standard mouse. That is, no head tracking, no StereoGraphics glasses, and no 6D mouse. Thus this test was performed using 'common' workstation technology. A user working with the complete 3D desktop version should perform better than with the version tested here.

For the 'real world' part of this experiment a portion of the database lab in the WSU Department of Computer Science the size of the CAVE was cleared. A real calendar was used as well as real objects to represent the various instruments. Plastic cups were used for the beakers. The thermometers and wind-socks were constructed from paper, wood and plastic. A desk covered with a blow-up of one of the satellite photographs took the place of the 3D plane. See Figure 5.28.

The first experiment dealt with the time needed to perform the three basic actions in the SANDBOX for the FIFE database, which are listed below:

- Placing an instrument at a site
- Choosing a date from the calendar

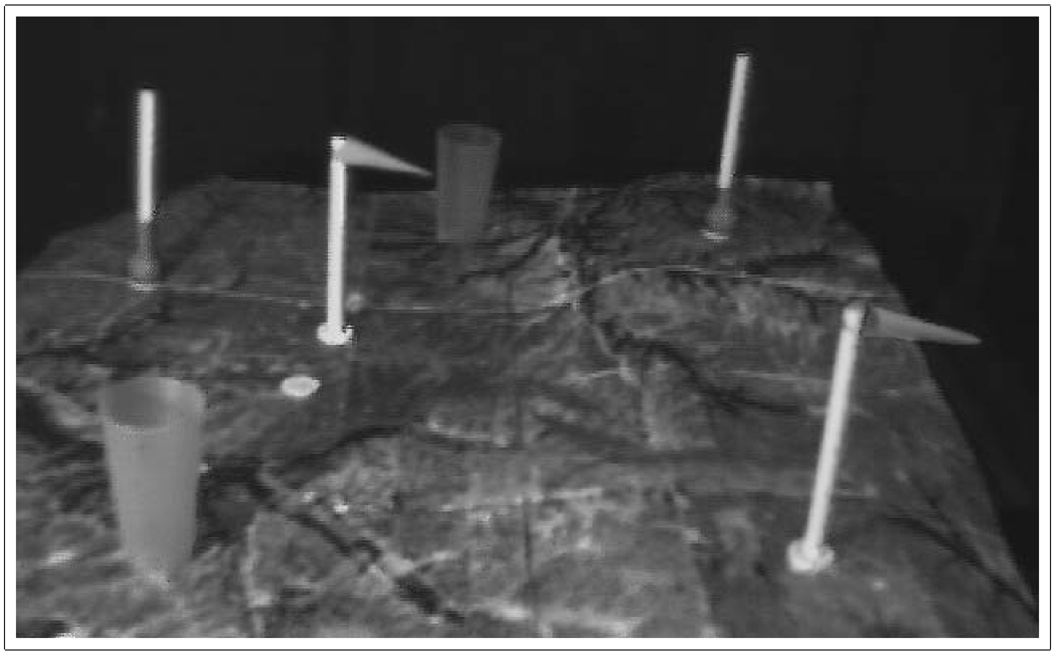


Figure 5.28: 'Real World' Instruments

- Adjusting the settings on an instrument

To test the time needed to place an instrument at a site, the user casually walked 5 feet to the instrument pallet, chose the beaker tool, walked back the same distance and placed the beaker at a certain site. As there is no physical distance in the desktop version, the desktop pointer was initially set to the center of the 3D plane.

To test the time needed to choose a date from the calendar, the user casually walked 5 feet to the calendar, and selected the 15th of August. In the 'real world' test the user touched the date to select it. As there is no physical distance in the desktop version, the desktop pointer was initially set to the center of the 3D plane.

To test the time needed to adjust the settings on an instrument, the user stood next to a thermometer on the 3D plane. The user activated the menu for the thermometer and increased its maximum temperature displayed by 10 degrees. In the 'real world' this test was not performed due to the nature of the virtual menus.

Five feet was used as the test distance as the CAVE is a ten foot cube. This is the

	'Real World'			CAVE			Desktop*		
(times in seconds)	min	ave	max	min	ave	max	min	ave	max
Placing an Instrument	3	4	6	6	7	7	13	15	22
Choosing a Date	2	2	3	3	3	4	6	9	13
Adjusting Settings	-	-	-	5	8	11	11	16	26

Table 5.3: Comparison of 'Real World', Desktop, and CAVE

same distance the user would walk from the center of the CAVE to one of the walls.

Several users performed each of the tests. The results of the first experiment are shown in Table 5.3. As expected the users of the 'real world' version of the interface performed the fastest. People have been performing these kinds of actions all their lives. Users in the CAVE were slower, but much faster than the users of the desktop version. Users perform very similar actions in the CAVE but have the extra encumbrance of the LCD glasses and the wand. Users of the desktop version have a much less direct and natural interface to the data leading to significantly longer access times.

Users of all three versions of the interface improved their performance through repeated use. The least improvement over time was seen with the 'real world.' There was moderate improvement over time with the CAVE and a large amount of improvement in the desktop version. As people have been taking and placing objects all their lives, there is little room for improvement. There is some room for improvement in the CAVE as the users become accustomed to the glasses and the wand. The desktop environment used here is very different from the 'reality' of grabbing and placing objects so there is great room for improvement as the user becomes more familiar with the controls.

5.11 Comparison of SANDBOX and Traditional Access Methods

In this section I will use several examples to illustrate how the SANDBOX can be a more effective data retrieval tool than traditional access methods. It is difficult to do direct comparisons because traditional access methods are designed to retrieving data for specific queries, where the SANDBOX is designed to provide an environment were the user can browse through related data.

While traditional access methods (i.e. query languages) favor those who are familiar with the structure and content of the database, the SANDBOX favors those who are familiar with the original experiments. Traditional access methods are appropriate for those who know what data they want, where it is in the database and how to retrieve it. They are appropriate for those who need to retrieve data based on complicated combinations of information. The SANDBOX is appropriate for those who are not sure what data they are interested in and need to browse through the data to find appropriate data to retrieve. The SANDBOX is appropriate for those who need to integrate graphical and meta-data with numeric and textual data.

The query language used in this section was SQL running on an Oracle database on a Macintosh Quadra 700. This was compared against the 'real world' environment described previously. Users taking part in the SQL tests were familiar with databases in general and with SQL, thus avoiding the training time that investigators less familiar with databases would need to incur.

Users of both SQL and the 'real world' interface were given a general introduction to the experiment in general and an overview of the interface. They were told which tables were important in the SQL interface and what their attributes meant, and how instruments were placed and dates selected in the 'real world.'

The SQL queries were performed twice. The first time, the user was unfamiliar with the database, being given only the information discussed above. The second time took place several days later, to see how fast the user could repeat the query.

In the 'real world' version of the SANDBOX test it was necessary for a person to 'play computer.' That is, a person performed all those activities that the environment manager would perform in the actual system.

5.11.1 Example 1: Visualizing Numeric Data

Let us assume an investigator is interested in knowing what directions the wind was blowing on August 28th, 1987. With the existing interface the user must go through the following sequence of actions:

1. Finds that the AMS_DATA_87 table contains the appropriate wind data
2. Finds the appropriate columns in the AMS_DATA_87 table

   ```
   DESCRIBE AMS_DATA_87;
   ```

3. Queries database for a list of all sites in the AMS_DATA_87 table on August 28th, 1987

   ```
   SELECT UNIQUE STATION_ID
   FROM AMS_DATA_87
   WHERE OBS_DATE = '28-AUG-87';
   ```

   returning a list of 10 sites

4. Chooses site 3
5. Queries database to extract wind information from site 3 on August 28th, 1987 from the AMS_DATA_87 table

```
SELECT OBS_TIME, U_WIND, V_WIND, W_WIND
FROM AMS_DATA_87
WHERE OBS_DATE = '28-AUG-87'
AND STATION_ID = 3
ORDER BY OBS_TIME ASC;
```

With the SANDBOX the user goes through the following sequence of actions:

1. Clicks on August 28th from the calendar
2. Grabs wind-sock instrument. All of the sites where wind measurements were taken are displayed
3. Places wind-sock at one of the sites

Looking at the resulting file, the user of the query language will find information in the form of x, y, and z components of the wind speed (e.g. 1.23, 4.27, 0.85.) The user must now compute the actual wind direction from these values.

Placing a wind-sock at site 3, the user of the SANDBOX would immediately see the direction that the wind was blowing and how that direction changes throughout the day.

User testing showed users familiar with SQL took an average of 5 minutes to get their results the first time, and 2 minutes the second time. Novice 'real world' users took only 26 seconds on average.

5.11.2 Example 2: Integrating Numeric Data and the Environment

Let us assume an investigator is interested in measuring rainfall at the highest point on August 4th, 1987. With the existing interface the user must go through the following sequence of actions:

1. Finds that the AMS_DATA_87 table contains the appropriate rainfall data

2. Finds the appropriate columns in the AMS_DATA_87 table

   ```
   DESCRIBE AMS_DATA_87;
   ```

3. Queries database for a list of all sites in the AMS_DATA_87 table on August 4th, 1987

   ```
   SELECT UNIQUE STATION_ID
   FROM AMS_DATA_87
   WHERE OBS_DATE = '4-AUG-87';
   ```

   returning a list of 10 sites

4. Finds that the FIFE_SITE_REF table relates site numbers to elevations

5. Finds the appropriate columns in the FIFE_SITE_REF table

   ```
   DESCRIBE FIFE_SITE_REF;
   ```

6. Queries database to find elevations of all the sites in the AMS_DATA_87 table

   ```
   SELECT STATION_ID, ELEVATION
   FROM FIFE_SITE_REF
   WHERE STATION_ID IN
      (SELECT UNIQUE STATION_ID
      FROM AMS_DATA_87
      WHERE OBS_DATE = '4-AUG-87')
      ORDER BY ELEVATION DESC;
   ```

7. Chooses site 11

8. Queries database to extract rainfall information from site 11 on August 16th, 1987 from the AMS_DATA_87 table

```
SELECT OBS_TIME, ACCUM_RAINFALL
FROM AMS_DATA_87
WHERE OBS_DATE = '4-AUG-87'
AND STATION_ID = 11
ORDER BY OBS_TIME ASC;
```

With the SANDBOX the user goes through the following sequence of actions:

1. Clicks on August 4th from the calendar

2. Grabs beaker instrument. All of the sites where rainfall measurements were taken are displayed

3. Looks at the 3D plane to see where the highest site is

4. Places beaker at the highest site

The SANDBOX allows the user to deal with familiar instruments such as beakers and familiar concepts such as elevation as opposed to artificial concepts such as tables and site IDs.

User testing showed users familiar with SQL took an average of 16 minutes to get their results the first time, and 7 minutes the second time. Novice 'real world' users took only 18 seconds on average.

5.11.3 Example 3: Integrating Numeric and Graphical Data

Let us assume an investigator is interested in comparing ground temperatures to the infra-red LANDSAT photograph on August 16th, 1987 to see how they correlate. With the existing interface the user must go through the following sequence of actions:

1. Finds that the AMS_DATA_87 table contains the appropriate temperature data

2. Finds the appropriate columns in the AMS_DATA_87 table

   ```
   DESCRIBE AMS_DATA_87;
   ```

3. Queries database for a list of all sites in the AMS_DATA_87 table on August 16th, 1987

   ```
   SELECT UNIQUE STATION_ID
   FROM AMS_DATA_87
   WHERE OBS_DATE = '16-AUG-87';
   ```

   returning a list of 10 sites

4. Finds that the FIFE_SITE_REF table relates site numbers to site locations

5. Finds the appropriate columns in the FIFE_SITE_REF table

   ```
   DESCRIBE FIFE_SITE_REF;
   ```

6. Queries database to find site locations of all the sites in the AMS_DATA_87 table

   ```
   SELECT STATION_ID, LATITUDE, LONGITUDE
   FROM FIFE_SITE_REF
   ```

```
WHERE STATION_ID IN
   (SELECT UNIQUE STATION_ID FROM AMS_DATA_87
   WHERE OBS_DATE = '16-AUG-87');
```

7. Finds the IR LANDSAT photo for August 1987
8. Finds the site locations on the IR LANDSAT photo
9. Chooses sites 3, 7, 25, and 29
10. Queries database to extract temperature information from the chosen sites on August 16th, 1987 from the AMS_DATA_87 table

```
SELECT STATION_ID, OBS_TIME, DRY_BULB_TEMP
FROM AMS_DATA_87
WHERE OBS_DATE = '16-AUG-87'
AND STATION_ID IN (3, 7, 25, 29)
ORDER BY OBS_TIME ASC;
```

With the SANDBOX the user goes through the following sequence of actions:

1. Clicks on August 16th from the calendar
2. Grabs satellite instrument and places it in the sky
3. Clicks on satellite to activate its menu and selects the IR band. The 3D plane now displays the landscape as seen in infra-red
4. Grabs thermometer instrument. All of the sites where temperature measurements were taken are displayed
5. Places thermometer at appropriate sites

Looking at the resulting file, the user of the query language will find that site 7 was out of action for most of the day and sites 25 and 29 never gave any results at all on the 16th of August. Only the measurements from site 3 are valid. The user must now go through the same sequence of actions again to re-query the database for information from other sites.

Placing thermometers at sites 7, 25, and 29, the user of the SANDBOX would immediately see that those instruments were not giving any feedback. The user could then move those thermometers to other nearby sites where the instruments give feedback. The SANDBOX allows the user to easily browse through the data itself, not just the structure that has been imposed on the data.

5.11.4 Example 4: Integrating Meta-Data and the Environment

Let us assume an investigator is interested in seeing which sites were on grassy planes. With the existing interface the user must go through the following sequence of actions:

1. Finds the site photographs
2. Converts photographs to viewable form
3. Chooses sites whose photographs show grassy planes
4. Queries database to find elevation of those sites and the surrounding sites (as photographs do not show full 360 degrees)

```
SELECT STATION_ID, LONGITUDE, LATITUDE, ELEVATION
FROM FIFE_SITE_REF
ORDER BY STATION_ID DESC;
```

5. Plots the locations and elevations of the chosen sites and surrounding sites to see which are indeed on flat grassy planes.

With the SANDBOX the user goes through the following sequence of actions:

1. Clicks on the camera instrument. All sites where site photographs were taken are displayed

2. Looks at the 3D plane to see which sites are on flat planes

3. Places camera at flat plane sites

4. Looks at displayed photographs to see which of those sites were grassy

5.11.5 User Comments

Users of the SQL interface were frequently frustrated by that interface. They needed scrap paper to remember table names and attribute names, and even then mistyped names and commands were common. They were constantly scrolling back through their previous queries to see what they had done already, and trying to get an overview of the data they were dealing with. Mistakes often meant screenfulls of useless data scrolling by or terse error massages. The interface was described as ‘annoying.’

The users felt their primary problem was that they did not know the overall structure of the database. The felt the secondary problem was that they were familiar with SQL but not everyday users. Finally, there was the problem of typing (and frequently mistyping) in the queries themselves.

It should be noted that these tests were performed with only 2 of the 106 tables in the FIFE database, and only with the appropriate columns, and only with the data for a single month. In these tests the users were given a very good idea of where the data was, their job was only to retrieve it.

On the other hand, the 'real world' version of the SANDBOX paradigm seemed very natural to the users and they were able to retrieve the necessary data much quicker, and easier.

The earlier experiments indicated that users performed actions in the 'real world' faster than they performed them in the CAVE and in turn faster than they performed them in the non-3D desktop version. While I was unable to attempt these experiments in the CAVE itself, a couple users did try them in the non-3D desktop version of the previous experiments. The first query took users 26 seconds on average in the 'real world' and 53 seconds in the non-3D desktop version. The second query took users 18 seconds in the 'real world' and 29 seconds in the non-3D desktop version. The CAVE and 3D desktop version should fall between the 'real world' and non-3D desktop values.

5.11.6 Cognitive Load

Data retrieval is only one aspect of the scientific analysis process [Mar84, SBM92]. The SANDBOX integrates data retrieval with some visualization ability and represents them using familiar metaphors, reducing cognitive load and enhancing the scientific analysis process.

Investigators using the traditional interface must first take their understanding of the experiments and convert it into an understanding of the database schema. As the schema is designed to facilitate efficient storage of information it may be very unrelated to the way users naturally think of the data. Then the investigators need to learn the query language to retrieve the data they require. Then, once they have the required data, they need to visualize that data in a way that is meaningful to them.

Investigators using the SANDBOX have several advantages. The database schema is hidden from the user, while still giving the user access to all the data. They deal with familiar concepts and instruments in a more natural environment. They can perform visualization while retrieving data.

5.12 Reaction

While testing the SANDBOX in the CAVE I received a lot of feedback from the users.

5.12.1 Reaction to the SANDBOX in the CAVE

My initial testing in the CAVE suggests that users find this paradigm to be very natural. Picking up and placing instruments appears to be very easy and intuitive. I've observed users bending down to place their instruments onto the 3D plane, even when they can just stand over the site and push the button on the wand. The environment and the instruments are realistic enough that the users are treating them as real objects.

Unfortunately this 'realism' can be physically tiring. Users must walk over to the instrument pallet to choose instruments, must walk over to the calendar to choose dates. This interface currently uses a limited form of raycasting, allowing the user to select a site by positioning the wand over the site, not forcing the user to actually place the instrument on the virtual 3D plane. Implementing raycasting to a greater extent, allowing the user to choose or instruments or dates from anywhere in the CAVE should further reduce this fatigue.

The wand may also not be the appropriate tool for the SANDBOX in the CAVE. Since the user is pointing at dates on the calendar, or carrying instruments off of the instrument pallet, the user's hand itself may be more appropriate. The CAVE also has a modified driving glove that can be used instead of the wand. This seems to give the user an even greater feeling of direct manipulation as now there is nothing between the user's actual hand and the virtual instrument they are carrying, or the menu they are activating.

Currently when the user is carrying a virtual instrument, the instrument is placed at the location of the physical wand. This can cause a couple of problems. First the wand itself visually occludes the walls so the entire virtual instrument can not be seen. Second, the focal point for all objects in the CAVE is at the walls. If the user is focusing on the virtual instrument then the wand is out of focus; if the user is focusing on the wand then the virtual instrument is no longer 3D. Future versions of the SANDBOX should move the location of the virtual instrument slightly away from the physical wand to avoid both the problem of occlusion and focus. This would be slightly less 'realistic', but more effective.

Some users have found the calendar and instrument pallet to be uncomfortably high. These currently extend from four to six feet off the ground. As discussed in Section 5.5, I allow the user to change the height of the 3D plane - initially positioned three feet above the floor. In future versions I will allow the user to change the height of the entire environment to make it more accessible and comfortable for users of different heights. In general, this suggests that the virtual environment should be more flexible.

Originally I positioned the calendar, graph, and instrument pallet just inside their respective physical walls. This gave the users a visual boundary to the environment so they would not accidentally walk into the CAVE walls. Unfortunately, the tracker's accuracy drops off near the walls forcing us to move all three of them one foot into the CAVE. This reduces the user's effective physical work area to 8 by 8 by 8 feet.

As the accuracy of the tracker improves the user's work space will increase.

Sound appears to be very useful in small doses (especially the water droplet when the rain is falling), but becomes overwhelming when overused. This is especially true when multiple cicadas are making noise. Since the temperature is usually quite constant across the 3D plane, one alternative may be to average the readings of all active thermometers. One cicada would then announce this average value. In general, this may be the most appropriate way to deal with continuous sounds.

5.12.2 Reaction to the FIFE Implementation

Several users, upon first entering the CAVE asked "Is this the moon?" The greyscale satellite photographs do not give the user the impression that they are in Kansas. This suggests that other images should be available for the 3D plane such as a colour picture from an atlas, or a local roadmap. These would give the user's a more familiar landscape to deal with. Even the existing satellite photographs might be enhanced using false colour. In general, this suggests that there is a greater need for supplementary environmental information in the SANDBOX.

There is also a need for some reference material beyond the information mapped onto the plane. There is currently no legend to tell the user what black or white represents on the satellite photograph. The user should be able to bring up more supplementary information on the graphics.

Even with additional graphical environmental information it would also be informative to place that environment in a larger context. In this case the user is standing in the middle of a 20km square block of Kansas, but where in Kansas? This contextual information could be supplied in several ways. When the SANDBOX for the FIFE data is started it could position the user in orbit above the site with the

Unites States outlined. It could then zoom the user into Kansas and finally into the experiment site.

A couple users have objected to the moon orbiting the 3D plane opposite the sun, as the Earth's moon does not do this. They did like the idea of the sun orbiting the 3D plane, and the ability to alter the behavior of time using the sun, but they felt that the behavior of the moon was not realistic. Removing the moon would only allow the user to change the behavior of time when the sun is above the 3D plane, which seems to be an unreasonable limitation. An alternative may be to use the same direct manipulation technique on the hands of a virtual analogue clock. The rotating hands of the clock would then serve the same function as the rotating sun and moon.

The sun rising in the east each morning is enough information to tell a user where north, south, east, and west are. However several users did have to think before being able to say which way was north. This suggests that a compass should be part of the virtual environment. This could simply be the letters 'N', 'S', 'E', 'W' floating at the appropriate sides of the 3D plane, or a much more elaborate overlay of the plane showing not only the compass directions but latitude and longitude values (or northing and easting values if the user prefers.)

The coloured lines on the graph make it easy to distinguish between the lines for the various instruments (thermometers in red, water beakers in blue, wind-socks in yellow.) However when two or more instruments of the same type are placed on the 3D plane there are multiple lines of the same colour, making it very difficult to see which line belongs to which instrument. Currently the user has the ability to turn the graph lines on or off for each instrument individually, but the user may need to see the values of several instruments of the same type over time. This suggests that the each of the lines and their associated instruments should be labeled to avoid confusion. Each line could be given a number or letter that could also then be placed, floating, alongside the proper instrument.

5.13 Conclusions

In this chapter I have described an implementation of the general design of the SANDBOX using the CAVE virtual reality theatre. I have shown how the user interacts with the virtual reality interface: placing instruments, choosing times, viewing data (numeric, graphical, meta-data), and extracting data. I have described a desktop version of this interface and discussed how it compares to the CAVE implementation. I have shown some initial reaction and the results of tests on the usefulness of this interface

5.14 Enhancements

While the SANDBOX makes interacting with a database more natural, there are still problems of information overhead to deal with. In the current implementation it would be very tiresome and repetitive to place many instruments at many sites, or to choose many days from the calendar. Clustering can be an effective means of reducing this burden.

5.14.1 Clustering Instruments

In the actual experiment scientists could place clusters of instruments simultaneously (e.g. a thermometer and a water beaker and a wind-sock.) Future versions of the SANDBOX should have this ability as well. The user should have the ability to combing existing instruments into instrument packages. These packages would then be available on the instrument pallet like any other instrument. Each of the individual instruments would function independently but they could be placed and removed as a single package.

In a large scientific database there would be many possible instruments, far too many to display conveniently on a single instrument pallet. An investigator doesn't take all their instruments with them when they go into the field, they only take those instruments that are necessary for the task at hand and leave the others behind. A second, much larger instrument pallet would be needed to hold all of the possible instruments. The user would be able to activate this pallet, scroll through it to the appropriate instruments, and then carry instruments from it over to the smaller pallet. This pallet could also contain additional meta-information about the instruments themselves.

Currently the user must adjust the settings on each instrument independently (their minimum and maximum values, whether they make sound, etc.) Future versions of the SANDBOX should allow to make changes to the instrument on the instrument pallet which would affect all future copies of that instrument pulled off of the instrument pallet.

5.14.2 Clustering Sites and Times

Being able to select multiple sites and times simultaneously would also be useful. How this would be done would depend heavily on the specific implementation of space and time in the virtual environment.

Giving the user a specific marking instrument, such as a small flag, would help the user remember sites. As in the field, the user might want to survey the landscape before placing any actual instruments. The user could place the marker instrument at likely looking sites and then come back later to place the instruments. These flags could have different colours for different topics. This way the SANDBOX remembers the sites for the user.

5.14.3 Initial Setup

Clearly the users would want the ability to save off their current virtual environment to come back to it later. We can also use this information to help first time users of the SANDBOX. As with the clustering that was described in Chapter 3, we can use the usage patterns of previous users to help new users.

The clustering algorithms of Chapter 3 clustered based on the tables accessed. When instruments are placed into the environment their definitions are converted into queries based on tables. By keeping track of these tables, they can be converted back into likely instruments.

When a user begins to use the SANDBOX they could choose from an pre-existing setup. This set up could include which instruments are automatically placed on the instrument pallet and their initial settings, what times are selected, or what instruments are already placed into the environment.

Chapter 6

Conclusions and Future Directions

In this chapter I will discuss my conclusions for the work overall and my plans for future work in these areas.

6.1 Conclusions

In this dissertation I have investigated two ways of improving access to scientific databases: clustering the tables of the database based on individual usage patterns, and using virtual reality to allow the user to access the scientific database by recreating the original experiments. Both methods allow the user to bypass the current structuring of the data. The adaptive clustering allows the user to bypass the structuring of the tables and deal with the concepts they are involved in. The SANDBOX allows the user to bypass the structure of the data completely and interact with the data directly.

6.2 Future Directions for Clustering

The clustering can be enhanced in several ways:

1. Clustering based on on attributes as well as tables,
2. Adapting the information collection to other database models (e.g. object-oriented),
3. Naming the clusters.

6.2.1 Clustering Based on Attributes

Currently the clustering treats a table as the smallest unit of information. This could be extended to the attribute level. Information is being collected not only on the tables accessed but on the attributes that being displayed, and the attributes used to join the tables together. This information could be used to cluster the attributes of the tables as well as the tables themselves. The user could then see which tables are appropriate to there area of interest and be able to look below the table level to the attribute level to see which attributes of those tables may be appropriate.

6.2.2 Adapting the Information Collection

The clustering is performed based on the lists created by the information collection process. Currently this process parses SQL queries. This can be extended to deal with query languages in other types of databases without affecting the clustering algorithms.

6.2.3 Naming the Clusters

Currently the clusters are identified by the tables they contain. It would be useful to be able to name these clusters to describe the concept that each cluster represents. This would require more data than the list names that are currently available. This could involve creating a list of keywords for each table in the database, or the user supplying keywords before beginning their queries.

6.3 Future Directions for the SANDBOX

The SANDBOX can be enhanced in several ways:

1. Increasing the visualization capabilities,

2. Allowing multiple investigators using CAVEs at geographically distant sites to cooperate on setting up a virtual experiment,

3. Decreasing the data retrieval time from the database using various access methods

4. Allowing the user to make annotations.

5. Creating a graphical toolkit to interface the SANDBOX to existing databases.

6.3.1 Visualization

Virtual reality is already being used to visualize information once it has been removed from scientific databases. The SANDBOX allows an investigator to employ virtual reality to retrieve data from a scientific database. Combining these, and giving the SANDBOX more features of a visualization system would give the user a total environment in which to do their work. These visualization features range from enhancing the graphing capabilities of the system to integrating existing visualization packages.

Currently the graph wall is limited to colour coded line charts for the various instruments placed in the virtual environment. Different types of charts (bar charts, pie charts, etc.) and multiple charts could be allowed. A better system of labeling the charts should also be used so that multiple instances of the same instrument can be differentiated on the chart wall.

One way to increase the visualization capabilities of the SANDBOX is to allow the interpolation of sites and times. In the current implementation the value for each instrument is the data value with the nearest time. The SANDBOX could be set to interpolate the data values from the set of nearest times. In our current implementation the user can only place a virtual instrument at any or all of those sites where a real instrument was placed during the actual experiment. The SANDBOX could allow the user to place the instrument anywhere and interpolate the values at that site from the values at the actual sites stored in the database. This added visualization capability would need to be carefully controlled so the user knows what data is real and what data is interpolated.

6.3.2 Cooperation

Investigators from many geographically distant sites came together to cooperate on setting up the actual experiments. The SANDBOX could similarly allow investigators to come together to set up large virtual experiments. This could involve some users with desktop systems and others with CAVEs. Users at different sites could place instruments in their own virtual environments and have them appear in all of the other virtual environments.

This would also require giving the users a way to communicate with each other. This could be done through a common 'blackboard' within the virtual environment, or through external communication channels (e.g. a cellular phone that the user brings into the CAVE or sits on their desk.)

6.3.3 Decreasing Data Access Times

Current databases are not well suited to supporting virtual reality applications. VR systems require very fast data access but little or no transaction management or updates to the database. Access time could be decreased through traditional query optimization techniques, through precompiled views for common groups of data, or through more exotic large data storage structures (i.e. persistent object stores) which stress retrieval speed. Decreasing the data access time would be especially important if each user did not have all of the data stored locally, but had to access it across a network from a remote site.

6.3.4 Annotations

The SANDBOX currently is designed to integrate the existing meta-information. This could be expanded to allow a user to add meta-information to the database, lists of sites, graphical visualizations, recorded messages, etc. This would give each user a more flexible environment to work in, and a way to pass along their findings to other users.

6.3.5 Graphical Toolkit

The SANDBOX gives the user an intuitive environment to retrieve data from a scientific database. In order to use this interface a designer must first create the virtual environment and link it to an existing database. Creating a graphical toolkit to help the designer do this will make it easier to adapt the SANDBOX to various databases. This toolkit would allow the user to choose the appropriate functions for each instrument and link them to the database in a direct graphical manner.

Appendix A

3D Illustrations

This appendix contains three stereo pictures from the SANDBOX. It is very difficult to get a true feeling for 3D environments using only 2D paper. These illustrations give you an opportunity to see how the environment looks in 3D. The first set of pictures shows the instrument pallet. The second set show the 3D plane. The third set shows the great lakes states population information.

The following stereoscopic pairs are arranged to be viewed by crossing your eyes. Your right eye should look at the picture on the left and your left eye should look at the picture on the right.

The easiest way to do this is to place the pictures on a table in front of you. Place your index finger between the two pictures and slowly move your finger towards your face. Focus your eyes on the finger. The two images will break into four and the middle two will merge together. When the middle two images have merged into one blurry image you are almost there. The trick then is to refocus your eyes on the middle 3D image.

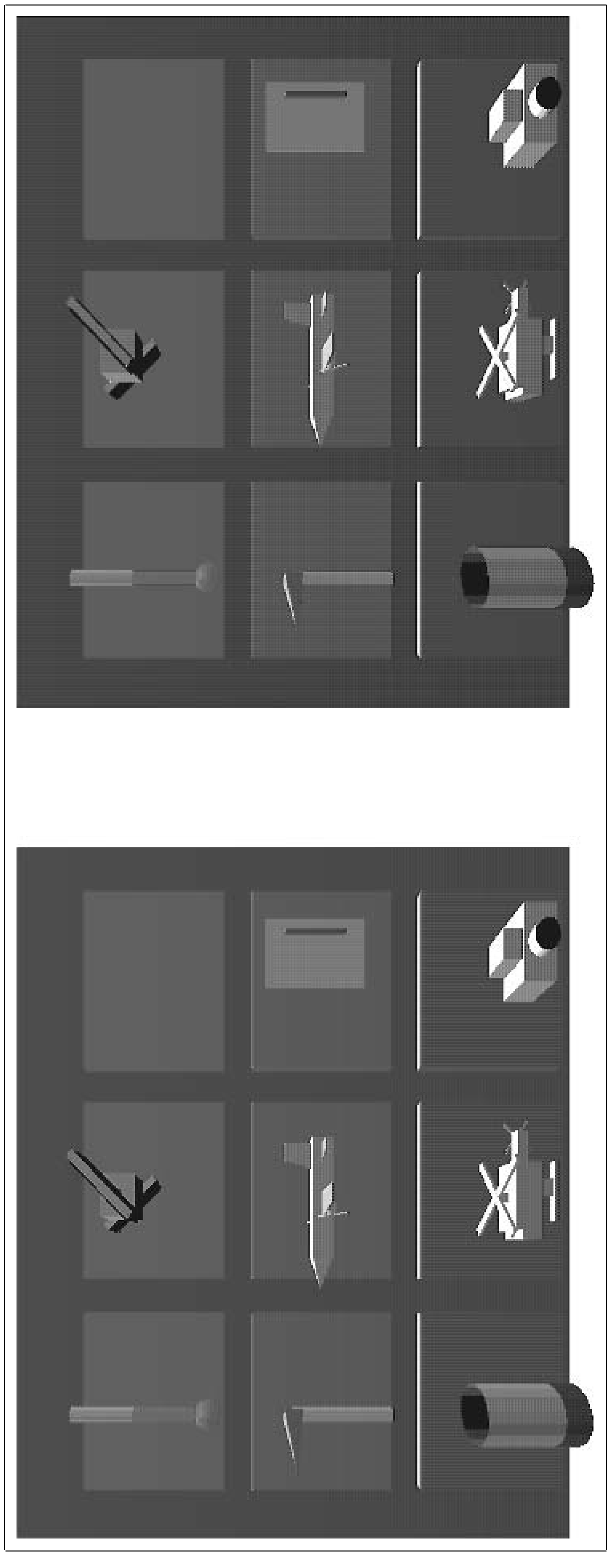


Figure A.1: 3D Image of the Instrument Pallet

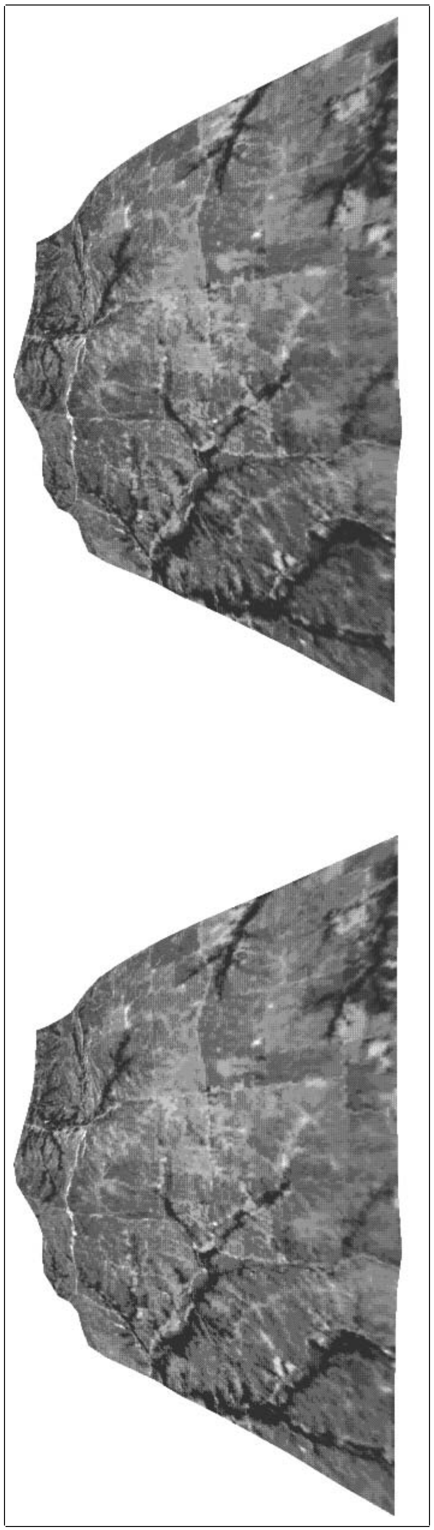


Figure A.2: 3D Image of the 3D Plane

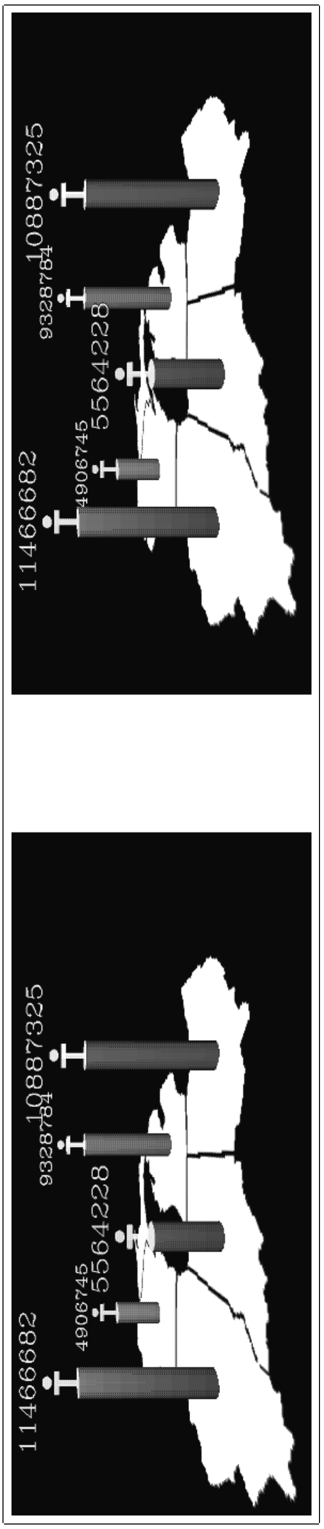


Figure A.3: 3D Image of the Great Lake State Populations

Bibliography

[AB92] S. Aukstakalnis and D. Blatner. *Silicon Mirage: the Art and Science of Virtual Reality.* Peachpit Press, Berkeley, 1992.

[AB93] K. Arthur and K. Booth. Evaluating 3D task performance for fish tank virtual worlds. *ACM Transactions on Information Systems*, 11(3):239–265, July 1993.

[AC93] G. Acciani and E. Chiarantoni. A neuron model for centroid estimation in clustering problems. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms 1993*, pages 131–136, Innsbruck, Austria, April 1993.

[ACF$^+$93] M. Arya, W. Cody, C. Faloutsos, J. Richardson, and A. Toga. QBISM: A prototype 3-D medical image database system. *Data Engineering*, 16(1):38–42, March 1993.

[Akk93] R. Akka. Utilizing 6D head-tracking data for stereoscopic computer graphic perspective transformations. In *Proceedings of SPIE '93 - Stereoscopic Displays and Applications IV vol. 1915*, pages 147–154, February 1993.

[AKRY91] J. Auerbach, M. Kennedy, J. Russell, and S. Yemini. Interprocess communication in concert/C. Technical Report RC17341, IBM T. J. Watson Research Center, October 1991.

[AMY88] R. Akscyn, D. McCracker, and E. Yoder. KMS: a distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[And93] M. Andreesen. NCSA mosaic technical summary rev. 2.1. Technical report, National Center for Supercomputing Applications, May 1993.

[AP92] A. Analyti and S. Pramanik. Fast search in main memory databases. In *Proceedings of SIGMOD 92 Conference*, pages 215–224, June 1992.

[Ast93] P. Astheimer. What you see is what you hear - acoustics applied to virtual worlds. In *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 100–107, October 1993.

[AVL88] B. Angéniol, G. Vaubois, and J. Le Texier. Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, 1(4):289–293, 1988.

[AWS92] C. Ahlberg, C. Williamson, and B. Schneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *Proceedings of SIGCHI '92*, pages 619–626, May 1992.

[BBK$^+$93] R. Bargar, M. Blattner, G. Kramer, J. Smith, and E. Wenzel. Effective nonspeech audio for virtual reality. In *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 113–116, October 1993.

[BBM93a] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[BBM93b] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993.

[BDG$^+$91] A. Beguelin, J. Dongarra, G. Geist, R. Manachek, and V. Sunderam. A user's guide to PVM parallel virtual machine. Technical Report ORNL/TM-11826, Oak Ridge National Laboratory, July 1991.

[BF90] M. Bolas and S. Fisher. Head-coupled remote stereoscopic camera system for telepresence applications. In *Proceedings of SPIE '90 - Stereoscopic Displays and Applications vol. 1256*, pages 136–146, February 1990.

[BG91] A. Baden and R. Grossman. Database computing in high energy physics. In *Computing in High-Energy Physics 1991*, pages 59–66. Universal Academy Press, Tokyo, 1991.

[BL91] S. Bryson and S. Levit. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings of Visualization '91*, pages 17–24, San Diego, California, October 1991.

[BOYBK90] F. Brooks Jr., M. Ouh-Young, J. Batter, and P. Kilpatrick. Project GROPE: Haptic displays for scientific visualization. *Computer Graphics*, 24(4):177–185, August 1990.

[Bro88] F. Brooks. Grasping reality through illusion - interactive graphics serving science. In *Proceedings of CHI '88*, pages 1–11, May 1988.

[Bus45] V. Bush. As we may think. *The Atlantic*, pages 101–108, July 1945.

[CB88] J. Conklin and M. Begeman. gIBIS: a hypertext tool for exploratory policy decision. *ACM TOOIS*, 6(4):303–331, October 1988.

[CBY89] T. Catlin, P. Bush, and N. Yankelovich. Internote: Extending a hypermedia framework to support annotative collaboration. In *Proceedings of the Hypertext '89 Conference*, pages 365–378, Pittsburg, Penn., November 1989.

[CFCP92] M. Cadisch, M. Farkas, J. Clerc, and E. Pretsch. Spectool: a hypermedia toolkit for structure elucidation. *Journal of Chem. Inf. Comput. Sci*, 32:286–290, 1992.

[CFG91] M. Creech, D. Freeze, and M. Griss. Using hypertext in selecting reusable software components. In *Proceedings of the Hypertext '91 Conference*, pages 25–38, San Antonio, Texas, December 1991.

[CG88a] B. Campbell and J. Goodman. HAM: a general purpose hypertext abstract machine. *Communications of the ACM*, 31(7):856–861, July 1988.

[CG88b] G. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21(3):77–88, March 1988.

[CLB$^+$93] C. Cruz-Neira, J. Leigh, C. Barnes, S. Cohen, S. Das, R. Engelmann, R. Hudson, M. Papka, T. Roy, L. Siegel, C. Vasilakis, T. DeFanti, and D. Sandin. Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment. In *Proceedings IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 59–65. IEEE Computer Society Press, October 1993.

[Con87] J Conklin. Hypertext: an introduction and survey. *IEEE Computer*, 20(9):17–41, September 1987.

[CRM91] S. Card, G. Robertson, and J. Mackinley. The information visualizer: an information workspace. In *Proceedings of CHI '91*, pages 181–188, New Orleans, Louisianna, April 1991.

[Cru93] C. Cruz-Neira, editor. *Course Notes 23, SIGGRAPH '93.* ACM SIGGRAPH, Aug 1993.

[CSD$^+$92] C. Cruz-Neira, D. Sandin, T. DeFanti, R. Kenyon, and J. Hart. The CAVE automatic virtual environment. *Communications of the ACM*, 35(2):64–72, jun 1992.

[CSD93] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[DKO$^+$84] D. DeWitt, R. Katz, F. Olken, L. Shapiro, M. Stonebraker, and D. Wood. Implementing techniques for main memory database systems. In *Proceedings of ACM SIGMOD '84 Conference*, pages 1–8, Boston, Mass, June 1984.

[DL91] P. Delany and G. Landlow. *Hypermedia and Literary Studies.* M.I.T. Press, 1991.

[DR86] J. Deogun and V. Raghavan. User-oriented clustering: A framework for learning in information retrieval. In *ACM Conference on Research & Development in Information Retrieval*, pages 157–163, September 1986.

[DS86] N. Delisle and M. Schwartz. Neptune: a hypertext system for CAD applications. In *Proceedings of ACM SIGMOD '86 Conference*, pages 132–139, Washington, D.C., May 1986.

[DSC93] T. DeFanti, D. Sandin, and C. Cruz-Neira. A room with a 'view'. *IEEE Computer*, 26(10):17–41, October 1993.

[Ele94] Electronic Visualization Lab, University of Illinois at Chicago. *CAVE User's Guide*, February 1994.

[ELK$^+$91] D. Egan, M. Lesk, D. Ketchum, C. Lochbaum, J. Remde, M. Littman, and T. Landauer. Hypertext for the electronic library? CORE sample results. In *Proceedings of the Hypertext '91 Conference*, pages 299–306, San Antonio, Texas, December 1991.

[Ess91] C. Ess. The pedagogy of computing: Hypermedia in the classroom. In *Proceedings of the Hypertext '91 Conference*, pages 277–289, San Antonio, Texas, December 1991.

[FB89] R. Fisher and P. Bandini. Stereoscopic cad and environmental sculpture: enhancement of the design process in the visual arts. In *Proceedings of SPIE '89 - Three-Dimensional Visualization and Display Technologies vol. 1083*, pages 4–17, 1989.

[FB91] N. Fuhr and C. Buckley. A probabilistic learning approachfor document indexing. *ACM Transactions on Information Systems*, 9(3), July 1991.

[FC89] M. Frisse and S. Cousins. Information retrieval from hypertext: Update on the dynamic medical handbook project. In *Proceedings of the Hypertext '89 Conference*, pages 199–212, Pittsburg, Penn., November 1989.

[FGK78] A. Flory, J. Gunther, and J. Kouloumdijan. Database reorganization by clustering methods. *Information Systems*, 3(1):59–62, 1978.

[FJP90] J. French, A. Jones, and J. Pfaltz. Summary of the final report of the NSF workshop on scientific database management. *SIGMOD RECORD*, 19(4):32–40, December 1990.

[FLS89] H. Fuchs, M. Levoy, and D. Santek. Interactive visualization of 3D medical data. *IEEE Computer*, 22(8):46–52, August 1989.

[Fri88] M. Frisse. Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31(7):880–886, July 1988.

[Gol89] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine learning*. Addison-Wesley, New York, 1989.

[Gol93] A. Goldberg. Concert/C tutorial: An introduction to a language for distributed C programming. Technical Report RA218, IBM T. J. Watson Research Center, 1993.

[Gor88] M. Gordon. Probabilistic and genetic algorithms for document retrieval. *Communications of the ACM*, 31(10):1208–1218, October 1988.

[GQV$^+$93] R. Grossman, X. Qin, D. Valsamis, W. Xu, C. Day, S. Loken, J. MacFarlane, D. Quarrie, E. May, D. Lifka, D. Malon, L. Price, A. Baden, L. Cormell, P. Leibold, D. Liu, U. Nixdorf, B. Scipioni, and T. Song. Analyzing high energy physics data using database computing. Technical Report LAC 94-R2, University of Illinois at Chicago, August 1993.

[GS87] P. Garg and W. Scacchi. On designing intelligent hypertext systems for information management in software engineering. In *Proceedings of the Hypertext '87 Conference*, pages 409–432, Chapel Hill, North Carolina, November 1987.

[GSO91] W. Gaver, R. Smith, and T. O'Shea. Effective sounds in complex systems: The arkola simulation. In *Proceedings of CHI '91*, pages 85–90, New Orleans, Louisianna, April 1991. ACM.

[Hal88] F. Halasz. Reflections on notecards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7):836–852, July 1988.

[Hol75] J. Holland. *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* The University of Michigan Press, Ann Arbor, 1975.

[HS89] W. Hibbard and D. Santek. Visualizing large data sets in the earth sciences. *IEEE Computer*, 22(8):53–57, August 1989.

[ILH92] Y. Ioannidis, M. Livny, and E. Haber. Graphical user interfaces for the management of scientific experiments and data. *SIGMOD RECORD*, 21(1):47–53, March 1992.

[ILH$^+$93] Y. Ioannidis, M. Livny, E. Haber, R. Miller, O. Tsatalos, and J. Wiener. Desktop experiment management. *Data Engineering*, 16(1):19–23, March 1993.

[Iwa90] H. Iwata. Artificial reality with force-feedback: Development of desktop virtual space with compact master manipulator. *Computer Graphics*, 24(4):165–170, August 1990.

[Jai92] R. Jain, editor. *NSF Workshop on Visual Information Management Systems.* 1992.

[JE92] R. Jacoby and S. Ellis. Using virtual menus in a virtual environment, in implementation of virtual environments. In *Course Notes 9, SIGGRAPH '92*, pages 12.1–12.9, July 1992.

[JF] A. Johnson and F. Fotouhi. Adaptive indexing in very large databases. *Journal of Database Management (in Press).*

[JF93] A. Johnson and F. Fotouhi. Automatic touring in a hypertext system. In *Proceedings of the 12th IEEE International Phoenix Conference on Computers and Communication*, pages 524–530, Tempe, Arizona, March 1993.

[JF94a] A. Johnson and F. Fotouhi. The SANDBOX: A virtual reality interface to scientific databases. In *Proceedings of the Seventh International Working Conference on Scientific and Statistical Database Management (in Press)*, Charlottesville, Virginia, September 1994.

[JF94b] A. Johnson and F. Fotouhi. The SANDBOX: Scientists Accessing Necessary Data Based On eXperimentation. Demonstration in the VROOM of SIGGRAPH '94, July 1994.

[JFG94a] A. Johnson, F. Fotouhi, and N. Goel. Adaptive clustering of scientific data. In *Proceedings of the 13th IEEE International Phoenix Conference on Computers and Communication*, pages 241–247, Tempe, Arizona, April 1994.

[JFG+94b] A. Johnson, F. Fotouhi, N. Goel, R. Weinand, and J. Lechvar. CASPER: A hypermedia departmental information system. In *Proceedings of ED-MEDIA 94*, page 658, Vancouver, Canada, June 1994.

[JFLD94] A. Johnson, F. Fotouhi, J. Leigh, and T. DeFanti. SANDBOX: an interface to scientific data based on experimentation. In *Proceedings of the Fifth Eurographics Workshop on Visualisation in Scientific Computing*, Rostock, Germany, May 1994.

[JMWU91] R. Jeffries, J. Miller, C. Wharton, and K. Uyeda. User interface evaluation in the real world: a comparison of four techniques. In *Proceedings of CHI '91*, pages 119–124, New Orleans, Louisianna, April 1991.

[Joh92] P. Johnson. *Human Computer Interaction: Psychology, Task Analysis and Software Engineering.* McGraw-Hill Book Company, 1992.

[Kim90] W. Kim. Object-oriented approach to managing statistical & scientific databases. In *Proceedings of the Fifth International Conference on Statistical & Scientific Database Management*, Charlotte, North Carolina, April 1990.

[KM89] M. Kuntz and R. Melchert. Pasta-3's graphical query language: Direct manipulation, cooperative queries, full expressive power. In *Proceedings of 15th International Conference on Very Large Data Bases*, pages 97–105, Amsterdam, Holland, August 1989.

[Koh88] T. Kohonen. An introduction to neural computing. *Neural Networks*, 1(1):3–16, 1988.

[Koh90] T. Kohonen. The self-organizing feature map. *Proc. IEEE*, 78(9):1464–1480, September 1990.

[Koh93a] T. Kohonen. Generalizations of the self-organizing map. In *Proceedings of the 1993 International Joint Conference on Neural Networks*, pages 457–461, October 1993.

[Koh93b] T. Kohonen. Things you haven't heard about the self organizing map. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages 1147–1156, March 1993.

[Koi93] H. Koike. The role of another dimension in software visualization. *ACM Transactions on Information Systems*, 11(3), July 1993.

[KP90] A. Kelley and I. Pohl. *A Book on C.* Benjamin Cummings, 1990.

[KSB93] J. Koh, M. Suk, and S. Bhandarkar. A mulit-layer kohonen's self-organizing feature map for range image segmentation. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages 1270–1276, March 1993.

[LC86a] T. Lehman and M. Carey. Query processing in main memory database management systems. In *Proceedings of ACM SIGMOD '86 Conference*, pages 239–250, Washington, D.C., May 1986.

[LC86b] T. Lehman and M. Carey. A study of index structures for main memory database management systems. In *Proceedings of the 12th VLDB Conference*, pages 294–303, 1986.

[LDL+93] J. Leigh, E. De Schutter, M. Lee, U. Bhalla, J. Bower, and T. DeFanti. Realistic modeling of brain structures with remote interaction between simulations of an inferior olivary neuron and a cerebellar purkinje cell. In *Proceedings of the Society for Computing Simulation, High Performance Computing 1993*, pages 248–254, Arlington, Virginia, March 1993.

[LEB93] R. Loftin, M. Engelberg, and R. Benedetti. Applying virtual reality in education: A prototypical virtual physics laboratory. In *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 67–74, October 1993.

[LK92] G. Landlow and P. Kahn. Where's the hypertext? the dickens web as a system independent hypertext. In *Proceedings of the ACM Conference on Hypertext '92*, pages 149–160, Milano, Italy, November 1992.

[LLL89] M. Long, K. Lyons, and J. Lam. Acquisition and representation of 2D and 3D data from turbulent flows and flames. *IEEE Computer*, 22(8):39–45, August 1989.

[LY77] J. Liou and S. Yao. Multidimensional clustering for database organization. *Information Systems*, 2:187–198, 1977.

[Mar84] G. Marks. How do analysts work? user interface issues. *Data Engineering*, 7(1):10–17, 1984.

[MBP+90] I. McDowall, M. Bolas, S. Peiper, S. Fisher, and J. Humpheries. Implementation and integration of a counterbalanced CRT-based stereoscopic display for interactive viewpoint control in virtual-environment applications. In *Proceedings of SPIE '90 - Stereoscopic Displays and Applications vol. 1256*, pages 136–146, February 1990.

[McL91] P. McLendon. *Graphics Library Programming Guide*. Silicon Graphics, Inc., Mountain View, California, 1991.

[MH88] M. Menon and K. Heinemann. Classification of patterns using a self-organizing neural network. *Neural Networks*, 1(3):201–215, 1988.

[MHR91] S. McWhorter, L. Hodges, and W. Rodrigues. Comparison of 3-d display formats for cad applications. Technical Report GIT-GVU-91-04, Graphics, Visualization & Usability Center, Georgia Institute of Technology, 1991.

[Mic91] Z. Michalewicz, editor. *Statistical and Scientific Databases*. Ellis Horwood, 1991.

[Mon84] A. Monk, editor. *Fundamentals of Human-Computer Interaction*. Academic Press, 1984.

[MRC91] J. Mackinley, G. Robertson, and S. Card. Perspective wall: Detail and context smoothly integrated. In *Proceedings of CHI '91*, pages 173–179, New Orleans, Louisianna, April 1991.

[Nie90a] J. Nielsen. *Hypertext and Hypermedia.* Academic Press, San Diego, 1990.

[Nie90b] R. Nielsen. *Neurocomputing.* Addison-Wesley, New York, 1990.

[Nor88] D. Norman. *The Design of Everyday Things.* Doubleday, 1988.

[OM87] B. Oommen and D. Ma. Fast object partitioning using stochastic learning automata. In *Proceedings of 10th Annual International ACMSIGIR Conference on Research & Development in Information Retrieval*, pages 111–122, June 1987.

[OW93] G. Özsoyoğlu and H. Wang. Example-based graphical database query languages. *IEEE Computer*, 26(5):25–38, May 1993.

[Poh94] I. Pohl. *C++ for C Programmers, Second Edition.* Benjamin Cummings, 1994.

[RCM93] G. Robertson, S. Card, and J. Mackinlay. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):57–71, April 1993.

[RD87] V. Raghavan and J. Deogun. Optimal determination of user-oriented clusters. In *Proceedings of 10th Annual International ACMSIGIR Conference on Research & Development in Information Retrieval*, pages 140–146, June 1987.

[Rib91] B. Ribarsky. Visualization in science and medicine. Technical Report GIT-GVU-91-27, Graphics, Visualization & Usability Center, Georgia Institute of Technology, 1991.

[RMC91] G. Robertson, J. Mackinley, and S. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of CHI '91*, pages 189–194, New Orleans, Louisianna, April 1991.

[RMS92] H. Ritter, T. Martinetz, and K. Schulten, editors. *Neural Computation and Self-Organizing Maps - An Introduction.* Addison-Wesley, 1992.

[RR90] M. Rafanelli and F. Ricci. A visual interface for statistical entities. *Data Engineering*, 13(3):35–43, September 1990.

[RT88] D. Raymond and F. Tompa. Hypertext and the oxford english dictionary. *Communications of the ACM*, 31(7):871–879, July 1988.

[RY92] J. Rasure and M. Young. An open environment for image processing software development. In *Proceedings of SPIE '92 - Symposium on Electronic Image Processing vol. 1659*, February 1992.

[SAD$^+$93] M. Stonebraker, R. Agrawal, U. Dayal, E. Neuhold, and A. Reuter. DBMS research at a crossroads: The vienna update. In *Proceedings of VLDB 1993*, pages 688–692, Dublin, Ireland, August 1993.

[Sae90] M. Saerens. Unsupervised clustering based on a competitive cost function. In *International Joint Conference on Neural Networks*, pages 343–348, San Diego, California, 1990.

[SBM92] R. Springmeyer, M. Blattner, and N. Max. A characterization of the scientific data analysis process. In *IEEE Visualization 92*, pages 235–242, 1992.

[SCG89] F. Shipman, J. Chaney, and A. Gorry. Distributed hypertext for collaborative research: the virtual notebook system. In *Proceedings of the Hypertext '89 Conference*, pages 147–158, Pittsburg, Penn., November 1989.

[Sch83] B. Schneiderman. Direct manipulation: a step beyond programming languages. *IEEE Computer*, 16(8):57–69, August 1983.

[Sch87] B. Schneiderman. User interface design for the hyperties electronic encyclopedia. In *Proceedings of the Hypertext '87 Conference*, pages 189–194, Chapel Hill, North Carolina, 1987.

[SCN$^+$93] M. Stonebraker, J. Chen, N. Nathan, C. Paxon, and J. Wu. Tioga: Providing data management support for visualization applications. In *Proceedings of VLDB 1993*, pages 25–38, Dublin, Ireland, August 1993.

[SGLS93] C. Shaw, M. Green, J. Liang, and Y. Sun. Decoupled simulation model in virtual reality with the MR toolkit. *ACM TOOIS*, 11(3):287–317, July 1993.

[SH89] P. Sellers and F. Hall. FIFE-89 experiment plan, July 1989.

[SHS$^+$90] P. Sellers, F. Hall, D. Strebel, R. Kelly, S. Verma, B. Markham, B. Blad, D. Schimel, J. Wang, and E. Kanemasu. FIFE interim report: Experimentation execution - results - analyses, February 1990.

[Sin88] M. Singhal. Issues and approaches to design of real-time database systems. *SIGMOD RECORD*, 17(1):19–31, March 1988.

[SLGS92] C. Shaw, J. Liang, M. Green, and Y. Sun. The decoupled simulation model for virtual reality systems. In *Proceedings of CHI '92*, pages 321–328. ACM, May 1992.

[SLKB88] J Schnase, J. Leggett, C. Kacmar, and C. Boyle. A comparison of hypertext systems. Technical Report TAMU 88-017, Texas A&M, September 1988.

[SLN$^+$91] D. Strebel, D. Landis, J. Newcomer, J. DeCampo, and B. Meeson. Collected data of the first ISLSCP field experiment, prototype: Satellite, aircraft, and ground measurements may-december, 1987. Published on CD-ROM by NASA, 1991.

[SLN$^+$92a] D. Strebel, D. Landis, J. Newcomer, S. Goetz, B. Meeson, P. Agbu, and J. McManus. Collected data of the first ISLSCP field experiment, volume 3: NS001 imagery, 1987 & 1989. Published on CD-ROM by NASA, 1992.

[SLN$^+$92b] D. Strebel, D. Landis, J. Newcomer, B. Meeson, P. Agbu, and J. McManus. Collected data of the first ISLSCP field experiment, volume 4: ASAS & PBMR imagery, 1987 & 1989. Published on CD-ROM by NASA, 1992.

[SLN$^+$92c] D. Strebel, D. Landis, J. Newcomer, D. van Elburg-Obler, B. Meeson, and P. Agbu. Collected data of the first ISLSCP field experiment, volume 2: Satellite imagery, 1987-1989. Published on CD-ROM by NASA, 1992.

[SN93] D. Song and M. Norman. Cosmic explorer: A virtual reality environment for exploring cosmic data. In *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 75–79, October 1993.

[SNL$^+$93] D. Strebel, J. Newcomer, D. Landis, J. Nickelson, S. Goetz, B. Meeson, P. Agbu, and J. McManus. Images derived from the first ISLSCP field experiment, volume 5: Images derived from satellite, aircraft, and geographic data. Published on CD-ROM by NASA, 1993.

[SNO89] D. Strebel, J. Newcomer, and J. Ormsby. Data management in the FIFE information system. In *Proceedings of International Geoscience and Remote Sensing Symposium*, pages 42–45, Vancouver, Canada, July 1989.

[Sny93] D. Snyers. Clique partitioning problem and genetic algorithms. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms 1993*, pages 352–360, Innsbruck, Austria, April 1993.

[SSAE93] T. Smith, J. Su, D. Agrawal, and A. El Abbadi. Database and modelling systems for the earth sciences. *Data Engineering*, 16(1):33–37, March 1993.

[Ste88] G. Stephenson. Knowledge browsing - front ends to statistical databases. In *Proceedings of IVth International Working Conference on Statistical and Scientific Database Management*, pages 327–337, Rome, Italy, June 1988.

[Ste92] W. Stevens. *Advanced Programming in the UNIX Environment.* Addison-Wesley, 1992.

[Sut65] I. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress 65*, pages 506–508, May 1965.

[Tei90] M. Teitel. The eyephone: a head-mounted stereo display. In *Proceedings of SPIE '90 - Stereoscopic Displays and Applications vol. 1256*, pages 168–171, February 1990.

[TKM$^+$91] I. Tomek, S. Khan, T. Muldner, M. Nassar, G. Novak, and P. Proszynski. Hypermedia - introduction and survey. *Journal of Microcomputer Applications*, 14:63–103, 1991.

[Tri88] R. Trigg. Guided tours and tabletops: Tools for communicating in a hypertext environment. *ACM TOOIS*, 6(4):398–414, October 1988.

[Ulu92] Ö. Ulusoy. Current research in real-time databases. *SIGMOD RECORD*, 21(4):16–21, December 1992.

[Ups89] C. Upson. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[Wal87] J. Walker. Document examiner: Delivery interface for hypertext documents. In *Proceedings of the Hypertext '87 Conference*, pages 307–324, Chapel Hill, North Carolina, November 1987.

[WB85] S. Weyer and A. Borning. A prototype electronic encyclopedia. *ACM TOOIS*, 3(1):63–88, January 1985.

[Web93] J. Weber. Visualization: Seeing is believing. *Byte*, pages 121–148, April 1993.

[Wex93] A. Wexelblat, editor. *Virtual Reality: Applications and Explorations.* Academic Press, Mass, 1993.

[Whi93] D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Colorado State University, November 1993.

[Wic91] A. Wichansky. User benefits of visualizing with 3D stereoscopic displays. In *Proceedings of SPIE '91 - Stereoscopic Displays and Applications II vol. 1457*, pages 267–271, February 1991.

[WK90] D. Winkler and S. Kamins. *Hypertalk 2.0: The Book.* Bantam Books, 1990.

[WRH92] C. Williams, J. Rasure, and C. Hansen. The state of the art of visual languages for visualization. In *IEEE Visualization 92*, pages 202–209, 1992.

[WS82] G. Wyszecki and W. Stiles. *Color Science.* John Wiley and Sons, New York, 1982.

[YHMD88] N. Yankelovich, B. Haan, N. Meyrowitz, and S. Drucker. Intermedia: the concept and the construction of a seamless information environment. *IEEE Computer*, 21(1):81–96, January 1988.

[YSL85] C. Yu, C. Suen, and K. Lam. Adaptive record clustering. *ACM Transactions on Database Systems*, 10(2):180–204, June 1985.

[YSLT81] C. Yu, M. Siu, K. Lam, and F. Tai. Adaptive clustering schemes: General framework. In *Proceedings of IEEE COMPSAC Conference*, pages 81–89, Chicago, 1981.

[ZLB+87] T. Zimmerman, J Lanier, C. Blanchard, S. Bryson, and Y Harvill. A hand gesture interface device. In *Proceedings of CHI '87*, pages 189–192, June 1987.

[ZMSD92] J. Zobel, A. Moffat, and R. Saks-Davis. An efficient indexing technique for full-text database systems. In *Proceedings of the 18th VLDB Conference*, pages 353–362, Dublin, Ireland, August 1992.

Abstract

SANDBOX: A VIRTUAL REALITY INTERFACE TO SCIENTIFIC DATABASES

by

ANDREW JOHNSON

DECEMBER 1994

Adviser: DR. FARSHAD FOTOUHI

Major: COMPUTER SCIENCE (Database)

Degree: DOCTOR OF PHILOSOPHY

Scientific databases contain very large amounts of data accessed by investigators from many disciplines. Much of the data that is stored in scientific databases is collected through experimentation. I propose a new interface to scientific databases: the SANDBOX: Scientists Accessing Necessary Data Based On eXperimentation. The SANDBOX is a virtual reality tool which allows an investigator to recreate the original experiment. The investigator places virtual instruments into a virtual reenactment of the original experiment and collects data from the scientific database in much the same way that the original data was collected. These instruments give visual and auditory feedback, allowing the user to browse through data of any type. I have implemented a prototype of the SANDBOX on a subset of NASA's FIFE scientific database using the CAVE virtual reality theatre.

Autobiographical Statement

Education:

Ph D. -	Wayne State University, Detroit, Michigan, 1994 Major: Computer Science (Database)
M.S. -	Wayne State University, Detroit, Michigan, 1990 Major: Computer Science (Database) Title of Thesis: A Parallel Hash-Based Approach for Computing the Transitive Closure of Database Relations
B.S. -	University of Michigan, Ann Arbor, Michigan, 1988 Major: Computer Engineering

Publications:

A. Johnson and F. Fotouhi. Adaptive indexing in very large databases. To appear in *Journal of Database Management.*

A. Johnson and F. Fotouhi. The SANDBOX: A virtual reality interface to scientific databases. To appear in *Proceedings of the Seventh International Working Conference on Scientific and Statistic Database Management*, Charlottesville, Virginia, September 1994.

A. Johnson and F. Fotouhi. the SANDBOX: Scientists Accessing Necessary Data Based On eXperimentation. Demonstration in the VROOM of SIGGRAPH '94, Orlando, Florida, July 1994.

A. Johnson, F. Fotouhi, N. Goel, R. Weinand, and J. Lechvar. CASPER: A hypermedia departmental information system. In *Proceedings of ED-MEDIA 94*, page 658, Vancouver, Canada, June 1994.

A. Johnson, F. Fotouhi, J. Leigh and T. DeFanti. SANDBOX: an interface to scientific data based on experimentation. In *Proceedings of the Fifth Eurographics Workshop on Visualization in Scientific Computing*, Rostock, Germany, May 1994.

A. Johnson, F. Fotouhi, and N. Goel. Adaptive clustering of scientific data. In *Proceedings of the 13th IEEE International Phoenix Conference on Computers and Communication*, pages 241–247, Tempe, Arizona, April 1994.

A. Johnson and F. Fotouhi. Automatic touring in a hypertext system. In *Proceedings of the 12th IEEE International Phoenix Conference on Computers and Communication*, pages 524–530, Tempe, Arizona, March 1993.

F. Fotouhi, A. Johnson, and S. Rana. A hash-based approach for computing the transitive closure of database relations. *The Computer Journal*, 35:A251–259, 1992.