A NEW PARADIGM FOR EXPLORATION IN COMPUTER-AIDED VISUALIZATION

by

Daryl H. Hepting

M.Sc., University of Regina, 1991

B.Sc. Honours, University of Regina, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
December 1999

APPROVAL

Name: Daryl H. Hepting

Degree: Doctor of Philosophy

Title of thesis: A New Paradigm for Exploration in Computer-Aided Visualization

Examining Committee:

Dr. Kori Inkpen
Chair

Dr. Robert D. Russell
Senior Supervisor

Dr. John C. Dill
Supervisor

Dr. F. David Fracchia
Supervisor

Dr. Tom Calvert
SFU Examiner

Dr. Joe Marks
External Examiner

Date Approved: _____

Abstract

This dissertation examines how the computer can aid the creative human endeavour which is data visualization. That computers now critically aid many fields is apparent, as is evidenced by the breadth of contemporary research on this topic.

Indeed, computers have contributed widely to the whole area of data comprehension, both in performing extensive computations and in producing visual representations of the results. Computers originally aided mathematicians who could both write the instructions necessary to direct the computer and interpret the resulting numbers. Even though modern computers include advanced graphical capabilities, many issues of access still remain: the users of data visualization software systems may not be experts in any computer-related field, yet they want to see visual representations of their data which allow them insight into their problems. For example, today's mathematicians who are generally expert in exploiting computational opportunities for experimentation may lack similar experience in opportunities for visual exploration.

Of particular concern is how a computer-aided visualization tool can be designed to support the user's goal of obtaining insight. There are many visual representations for a given set of data, and different people may obtain insight from different visual representations. Selection of the "best" one for an individual can be exceedingly difficult, as the sheer number of possible representations may be staggering. Current software designs either recognize the possibility of overwhelming the individual and therefore employ some means of restricting the choices that the user is allowed to make, or the designs focus on providing only the raw materials necessary for constructing the representations, leaving the user unrestricted but potentially unaided in searching out the desired representation.

The novel approach presented in this dissertation adapts a genetic algorithm to provide a means for an individual to search alternative visual representations in a systematic and

manageable way. Any visual representation is a combination of elements, each selected from a different component. This approach encourages the individual's creativity without restricting available choices, and leaves the task of bookkeeping to the computer.

A computer-aided visualization system which is driven by the unique preferences of each user has been developed. The efficacy of this system, *cogito*, is demonstrated through a software user study. From an initial specification of components and elements, the system provides a wide variety of visual representations. From within this range of available visual representations, the user pursues the goal of achieving insight by applying personal criteria for effectiveness to the iterative selection and evaluation of candidate representations.

To Jennifer

"In short, it seems worthwhile to avoid argument with (other) enthusiasts for artificial intelligence by conceding dominance in the distant future of cerebration to machines alone. There will nevertheless be a fairly long interim during which the main intellectual advances will be made by men and computers working together in intimate association. A multidisciplinary study group, examining future research and development problems of the [United States] Air Force, estimated that it would be 1980 before developments in artificial intelligence make it possible for machines alone to do much thinking or problem solving of military significance. That would leave, say, five years to develop man-computer symbiosis and fifteen years to use it. The fifteen may be ten or five hundred, but those years should be intellectually the most creative and exciting in the history of mankind"

— *Man-Computer Symbiosis*

J. C. R. Licklider, *1960*

Acknowledgments

I offer my heartfelt thanks to my supervisory committee of Bob Russell, John Dill and Dave Fracchia; they gave me the support and encouragement I needed to complete the journey which this document represents. As my senior supervisor, Dr. Russell gave me the freedom and the confidence to pursue my goal even as it diverged from his own areas of interest. He also played a key role in my education outside of the university. Dr. Dill was my first contact from Simon Fraser University, and he was decisive in my choice to come here, all those years ago. Dr. Fracchia and I have a long history together, going back to our graduate studies at the University of Regina, and I wouldn't trade it for the world. I also thank my internal examiner, Tom Calvert, and my external examiner, Joe Marks of MERL. As director of the Graphics and Multimedia Research Lab, Dr. Calvert always made me feel welcome there. Dr. Marks, who honoured me by travelling a great distance to take part in my examination, was extremely helpful and generous. All their careful comments helped to further improve the quality of this document.

The software user study, which forms a large part of this dissertation, benefitted tremendously from the input of Brian Fisher and Kori Inkpen. Ian Bercovitz and Kori Inkpen helped me to make sense of the results. Frank Manuel, Drew Miners, and Stephen Nix were always there to provide technical support.

I would require a separate tome to thank everyone by name, so it must suffice to say that those who are unnamed are still very much appreciated.

This dissertation has been a long time coming and I want to thank everyone in the School of Computing Science who helped me along the way; I think of Kersti Jaager, Elma Krbavac and Art Liestman in particular. I thank all of the students with whom I shared the qualifying year of the doctoral program, and Micheline Kamber in particular. I thank all the members of the Graphics and Multimedia Research Lab, past and present, for creating

a great environment. Especially, thanks to Armin Bruderlin, with whom I first shared an office and to Lyn Bartram, Sheelagh Carpendale, Dave Cowperthwaite, and Maria Lantin with whom I have shared the trials and tribulations of finishing.

I cannot help but thank again Przemyslaw Prusinkiewicz, supervisor of my Master's thesis, for starting me on my graduate school career and sharing some very exciting times with me. Through him, I made the connection which allowed me the privilege of working for Benoit Mandelbrot at IBM Research, before coming to Simon Fraser University. Dr. Mandelbrot gave me many things, including a lasting appreciation of the power of (computer-generated) pictures as tools for understanding.

John Hart and Ken Musgrave have been great friends and with whom I have shared much, personally and professionally, even though our time together is w usually compressed into our yearly meeting at SIGGRAPH. I first met Jessica Hodgins while at IBM, and I thank her both for continued encouragement and for introducing me to Joe Marks. Alan Law, a supervisor from Regina, has continually encouraged me with regular e-mail messages exhorting me to "work harder!"

No matter where I go, I will always be "from Regina." Thank you to all my friends and family there who have given me so much encouragement. Particularly, I thank my parents, Irvin and Gert, whose love, support, and faith made all this possible.

I have been supported here by my extended family and especially by my parents-in-law Carol and Pat. Although Pat passed away last year, he is remembered. Finally, I thank my wonderful wife, Jennifer, who has shared this journey with me and is ready to share in the next one, along with our child who is expected to arrive in the New Year.

Contents

Approval ii

Abstract iii

Dedication v

Quotation vi

Acknowledgments vii

List of Tables xiii

List of Figures xvi

1 Introduction 1

1.1 Computers, Insight, and Interpretation 2

1.2 Computer-aided visualization 5

1.3 Motivation 6

1.4 Thesis Statement 7

1.5 Outline 8

2 Computer-Aided Visualization 9

2.1 Definition 9

2.2 Historical Development 10

2.3 The Visualization Process 11

2.3.1 Data 14

2.3.2 Visual Representation 15

2.3.3 Evaluation . . . 18
2.4 Computerization . . . 18
2.4.1 Manual . . . 19
2.4.2 Augmented . . . 22
2.4.3 Automated . . . 24
2.5 Implications . . . 25

3 A New Paradigm **27**
3.1 Insight and Problem-Solving . . . 27
3.2 Graphics as Communication Medium . . . 33
3.3 Computer as Enabling Technology . . . 35

4 Implementation of *cogito* **38**
4.1 Basic Organization . . . 39
4.2 Input . . . 40
4.2.1 Abstraction . . . 41
4.2.2 Libraries . . . 46
4.2.3 Database . . . 52
4.2.4 Display . . . 52
4.2.5 Implementation Modules . . . 52
4.3 Configuration . . . 57
4.4 Selection and Evaluation . . . 57
4.5 Output . . . 63
4.6 Implications . . . 63

5 Building applications for *cogito* **65**
5.1 Defining an Abstraction . . . 66
5.1.1 Shapes . . . 66
5.1.2 Graphs . . . 69
5.2 Defining the library . . . 73
5.2.1 Shapes . . . 75
5.2.2 Graphs . . . 79
5.3 Specifying the Display and Configuration . . . 82
5.3.1 Shapes . . . 82

5.3.2 Graphs . . . 83
5.4 Implications . . . 83

6 Assessment **84**
6.1 Test Structure . . . 84
6.2 Software . . . 85
6.2.1 Interface A (Flat) . . . 85
6.2.2 Interface B (Hierarchical) . . . 87
6.3 Hypotheses . . . 90
6.4 Experiment Design . . . 90
6.4.1 Pre-task questionnaire . . . 91
6.4.2 Training Task . . . 92
6.4.3 Main Task . . . 92
6.4.4 Post-task Questionnaire . . . 93
6.5 Analysis of Results . . . 94
6.5.1 Pre-task questionnaire . . . 94
6.5.2 Post-task questionnaire . . . 95
6.5.3 Selected graphs . . . 104
6.6 Discussion . . . 104

7 Conclusions **115**
7.1 Specific Contributions . . . 116
7.2 Future work . . . 117

A User study materials **119**

B User study comments **144**
B.1 Which features did you like the most? (Question 18) . . . 144
B.1.1 Responses from users of Interface A (flat) . . . 144
B.1.2 Responses from users of Interface B (hierarchical) . . . 146
B.2 Which features did you like the least? (Question 19) . . . 147
B.2.1 Responses from users of Interface A (flat) . . . 147
B.2.2 Responses from users of Interface B (hierarchical) . . . 149
B.3 Are there features you'd like to see added to the software? (Question 20) . . . 150

B.3.1 Responses from users of Interface A (flat) 150
B.3.2 Responses from users of Interface B (hierarchical) 151
B.4 Do you have any other comments about the software or the task? (Question 21) 153
B.4.1 Responses from users of Interface A (flat) 153
B.4.2 Responses from users of Interface B (hierarchical) 154

C A sample output file **156**

Bibliography **159**

List of Tables

3.1 Respective capabilities of humans and computers, adapted from Baecker *et al.* [7]. 35

4.1 Summary of Design Principles (**DP**) developed in Chapter 3 40
4.2 Syntax for the specification of compatibilities. 42
4.3 A compatibility called dimension, with two conditions 43
4.4 Syntax for the specification of components. 44
4.5 An example of a component specification. 44
4.6 Syntax for the specification of catalogues. 45
4.7 Two examples of catalogues. The first is descriptive because it will allow different values to be named, and the second is referential because it will allow code segments to be accessed. 45
4.8 Syntax for data set formats. 47
4.9 Designating a format for a data set comprised of segments, each identified by the value of "Time" in the headers. 48
4.10 In an example from the *Graphs* application (see Chapter 5), the `percent100` relation is used to indicate that the percentage in each of the three sectors sums to 100. By testing the number of data fields in the relation, the programmer can verify that the trilinear plot may be used with this data. 48
4.11 Syntax for library specification. 49
4.12 Syntax for library specifications, continued. 50
4.13 Specifying a very small library, which refers to two functions, `CAM_orthoXY` and `linearRamp_RGB`. 51
4.14 Syntax for database specification. 53

4.15 Input of data corresponding to the dataset format given in Table 4.9. Here the dataset is called "sampleGrid" and is an instance of the format called "mesh." . . . 54
4.16 Syntax for the specification of display qualifiers. . . . 55
4.17 Header file for the CogStateInfo class . . . 56
4.18 Header file for the CogStateFuncInfo class . . . 56
4.19 Syntax for the specification of configuration details. . . . 57

5.1 Abstraction for *Shapes* application. . . . 68
5.2 Abstraction for *Shapes* application, continued. . . . 69
5.3 Abstraction for *Graphs* application: compatibilities. . . . 71
5.4 Abstraction for *Graphs* application: components. . . . 72
5.5 Abstraction for *Graphs* application: catalogues. . . . 73
5.6 Abstraction for *Graphs* application: dataset formats. . . . 74
5.7 Colour elements for *Shapes* application. . . . 75
5.8 Colour elements for *Shapes* application, continued. . . . 76
5.9 The code to implement the `MAT_diffuse` module that is referenced in Table 5.7 and 5.8. . . . 77
5.10 An alternative definition for an element which provides diffuse colours in the *Shapes* application. The arguments specify $2^3 = 8$ possibilities. The `no_bw` validator module is used to reduce that to 6 in total, the same ones specified in Tables 5.7 and 5.8. . . . 78
5.11 Validator module for the `MAT_diffuse` element module which eliminates the colours $(0, 0, 0)$ and $(1, 1, 1)$. . . . 78
5.12 An excerpt of elements and designations made available in the *Graphs* application DSO. . . . 79
5.13 A validator module for the "Trilinear plot" representation which uses the CogLib API to test whether the necessary relation exists and has the proper size. . . . 81
5.14 Specification of display parameters for the *Shapes* application. . . . 82
5.15 Specification of display parameters for the *Graphs* application. In this example, the `dataAmplification` component is aliased to the name `sort`. . . . 83

6.1 Analysis of frequencies of response categories for the pre-task questionnaire data for all subjects ($N = 34$). 95
6.2 Pre-task questionnaire responses, grouped by interface. 96
6.3 Post-task questionnaire responses for questions 1 through 9, grouped by interface. 97
6.4 Post-task questionnaire responses for questions 10 through 17, grouped by interface. 98
6.5 Contingency table for exploration and involvement: the relationship between exploration and involvement is significant ($p = 0.031$), so $\mathtt{H_0\text{-}activity}$ can be rejected. 100
6.6 Summary of groups used for the analysis of questionnaire data. 101
6.7 Distribution of cumulative scores, derived from post-task questionnaire responses. The scores are given in the center column with the count from Group A and B displayed to either side. The lowest value on the graph is 34, which corresponds to an average of 2 ("unsatisfied"), 51 corresponds to an average of 3 ("satisfied"), 68 corresponds to an average of 4 ("very satisfied"). 102

A.1 Full text of e-mail message sent to recruit subjects for the study. 121
A.2 Part 1 of data for the main task 138
A.3 Part 2 of data for the main task 139
A.4 Part 3 of data for the main task 140

List of Figures

1.1 The famous "Nine Dots" insight problem [262]. The problem is to draw four straight lines, without lifting pen from paper, so that each of the nine dots on the left is crossed by a line. A solution is shown on the right. 2

2.1 Haber and McNabb's model [100] of the visualization process in terms of abstractions (solid frames on the left) and transformations (dashed frames on the right). 13
2.2 Information flow model described by Nielson [177], expanding on the basic loop of scientific discovery. 14
2.3 The decomposition of scientific data analysis into its process elements, after Springmeyer *et al.* [234]. 14
2.4 Organization of data displays (after Clark [43]) according to primary function, then intended use, then presentation goal. The parenthetical comments indicate the governing principle applied to each type of presentation. 17

3.1 The relationship amongst factors in graphic communication, after DiBiase [156] 34
3.2 The same data represented with a line chart (left) and a trilinear plot (right). 37

4.1 A simplified model of the *cogito* system organization. The 'SELECT AND EVALUATE' step may include details of arbitrary complexity, yet this complexity is hidden from the user. 39
4.2 Schematic look at the interface: the space of available alternatives is grouped according to user-specified criteria. Each group (A – F) has a representative element (a – f) which is displayed to the user. The subspace for the next search iteration is based on the user selection (b and f). 58

4.3 The interface to the *cogito* system, with annotations. The user will always see the same interface, regardless of the particular application being run (here shown with the *Shapes* application, described in Chapter 5). 61
4.4 An illustration of the *cogito* system with an application. At the forefront is the dialogue box which permits editing of the current visual representation. . 62

5.1 The *cogito* system running the *Shapes* application. 67
5.2 The *cogito* system running the *Graphs* application. 70

6.1 Interface A (Flat) with the *Shapes* application. 86
6.2 Interface B (Hierarchical) with the *Shapes* application. 88
6.3 Refine dialogue box for Interface B (Hierarchical) with the *Shapes* application. 88
6.4 Examine dialogue box for Interface B (Hierarchical) with the *Shapes* application. 89
6.5 Four selected images from the *Shapes* application. 105
6.6 Four selected images from the *Graphs* application. 106
6.7 Frequencies of representation element choices. 107
6.8 Frequencies of plotData element choices. 107
6.9 Frequencies of mapData element choices. 107
6.10 Frequencies of dataAmplification element choices. 108
6.11 Frequencies of layout element choices. 108
6.12 Frequencies of annotation element choices. 108
6.13 Frequencies of colourPalette element choices. 109
6.14 Frequencies of colourSelect element choices. 109
6.15 Frequencies of size element choices. 109
6.16 Frequency of selections for interface A. The lines connect pairs of elements chosen from adjacent components. The style in which the line is drawn indicates the frequency that the pair was selected. 110
6.17 Frequency of selections for interface B. The lines connect pairs of elements chosen from adjacent components. The style in which the line is drawn indicates the frequency that the pair was selected. 111

A.1 Ethics approval letter 120
A.2 Information for subjects. 122

A.3 Informed consent form. . . . 123
A.4 Subject feedback form. . . . 124
A.5 Sample graphs given for reference while completing pre-task questionnaire. . . 125
A.6 Pre-task questionnaire. . . . 126
A.7 Introduction to study. . . . 127
A.8 Tutorial for interface A, page 1. . . . 128
A.9 Tutorial for interface A, page 2. . . . 129
A.10 Tutorial for interface A, page 3. . . . 130
A.11 Tutorial for interface B, page 1. . . . 131
A.12 Tutorial for interface B, page 2. . . . 132
A.13 Tutorial for interface B, page 3. . . . 133
A.14 Tutorial for interface B, page 4. . . . 134
A.15 Tutorial for interface B, page 5. . . . 135
A.16 Training task instructions. . . . 136
A.17 Task instructions. . . . 137
A.18 Post-task questionnaire, page 1 141
A.19 Post-task questionnaire, page 2 142
A.20 Post-task questionnaire, page 3 143

Chapter 1

Introduction

In the *Commercial and Political Atlas* [198] of 1786, William Playfair began the modern use of graphics with his visualization of economic variables. Not only could these graphical forms communicate ideas effectively, they also provided a previously unavailable means to check the integrity of data.

Two centuries later, scientific visualization using computers has become commonplace. Following the landmark 1987 *Visualization in Scientific Computing* report by McCormick *et al.* [163], this field has emerged as a means to make vast quantities of data perceivable, and ultimately understandable, through computer-graphical means. A wide array of problems have been aided by these new visualization methods, ranging from the tangible thunderstorm work described by Baker and Bushnell [8] to the abstract minimal surfaces of Callahan *et al.* [30].

In the dozen years which have followed this report [163], scientific visualization has moved beyond the purview of experts in computer graphics both with multidisciplinary "Renaissance" teams [163] and accessible, powerful computational and visual tools.

Regardless of whether one programs directly with a graphics library, uses a turnkey software system, or collaborates with a multidisciplinary team, the success of the visualization is determined by the effectiveness of the resulting visual imagery. Yet "effectiveness" is largely a subjective criterion whose application is related to each user's own experience. The design of a computer-aided tool, which supports the user in exploring new alternatives in a systematic way, will be the focus of this dissertation.

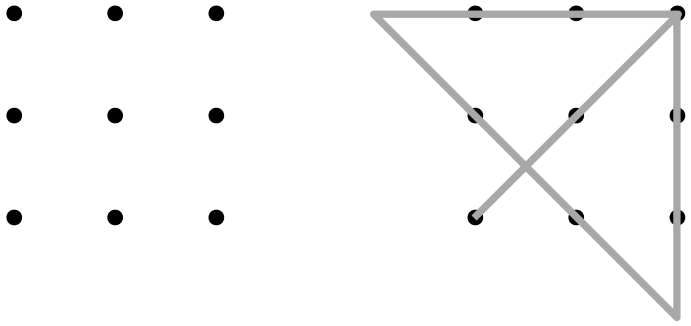


Figure 1.1: The famous "Nine Dots" insight problem [262]. The problem is to draw four straight lines, without lifting pen from paper, so that each of the nine dots on the left is crossed by a line. A solution is shown on the right.

1.1 Computers, Insight, and Interpretation

Visualization in scientific computing can trace its roots back to the beginnings of modern computing. Even before the advent of computer graphics, numerical simulations were an important tool for scientific insight. In the 1940's, John von Neumann [256] wrote about the potential for the use of computers in the study of differential equations. It is this potential of the computer as an experimental tool which caused Richard Hamming [103] to write "the purpose of computing is insight, not numbers" as the motto for his 1962 text, *Numerical Methods for Scientists and Engineers*.

Insight can sometimes be elusive, especially when the subject matter is complex. There are many examples of so-called "insight" problems, the solutions of which require unconventional thinking. A quite famous example of this genre is the "Nine Dots" puzzle, illustrated in Figure 1.1. Attempts to study the process by which people tackle such problems have yielded lively debates, but as yet little agreement [60, 262].

History is filled with examples of creative thinkers who could go "outside the nine dots" of their particular problems, by employing visual thinking unaided by computers. For example, Einstein conducted a famous and well-documented [170] thought experiment in which he visualized what it would be like to ride a wave of light.

Insight from images or numbers, when it occurs, comes through interpretation, which is defined to be "the action of explaining the meaning of something" [190]. The derivation of meaning is an important philosophical question, which has a long tradition in *semiotics* and

hermeneutics. Semiotics is concerned with the function of signs, symbols, and interpretation. It considers a person's relationship with the object and its sign. Semioticians differ about whether or not there is a direct connection between the person and the object (see Fiske [80]). The derivation of meaning is also addressed by hermeneutics (see Palmer [185]), the study of interpretation of texts. Within that community, there is a division between those for whom meaning exists independently of the act of interpretation and those who contend that meaning arises from a process in which the text, its production, and its interpretation all play essential roles.

Of primary interest to the present work is the way in which meaning can be assigned to pictures. Bertin [12] contends that figurative images, in which the meaning of any sign is determined in relation to the other signs, are *polysemic* (several possible meanings); and graphics, in which the unique meaning of each sign is specified through a legend, are *monosemic* (one possible meaning). According to Bertin [12], "graphics owes its special significance to its double function as a storage mechanism and a research instrument. A rational and efficient tool when the properties of visual perception are competently utilized [by the designer of the graphic], graphics is one of the major 'languages' applicable to information processing." Of fundamental importance is Bertin's work on codifying, through visual variables, the expressive possibilities in two-dimensional graphs. Others have explored characterizations based on different criteria [15]. Ware [259] distinguished between sensory and conventional aspects of communication, based on whether the processing is perceptual. Bertin's graphics employ sensory aspects and rely on a legend to fix conventional meanings. Though Bertin contends that both graphics and mathematics are monosemic systems, any articulation of the details surrounding a particular graphic will be, by necessity, incomplete [269]. Furthermore, the objective view of mathematics is currently much less prevalent, even amongst mathematicians [142].

Playfair [198] described how graphs give a lasting impression of the data which they depict. Clark [43] has called this the "Eureka" effect, which may be so strong as to obscure other interpretations of the data. Therefore, care must be taken to ensure that the impression is based on an accurate and meaningful presentation of the data. Wainer [257] discusses how an incomplete graph played a role in the failure to cancel the doomed flight of the U.S. Space Shuttle *Challenger*. Those reading the graph did not see the complete set of data – the graph was made for them by others. That images are very powerful is evidenced by the acceptance of the proverb, "one picture is worth ten thousand words," which still

resonates even with people who know that it originated as an advertising slogan [110].

There is practically an infinite variety of visual representations for any set of data. Bertin wrote that "to construct 100 DIFFERENT FIGURES from the same information requires less imagination than patience. However, certain choices become compelling due to their greater 'efficiency.'" Bertin [12] and Larkin and Simon [144] define efficiency in terms of the effort required to comprehend an image. Following the distinctions of Ware [259], there are both sensory and conventional aspects of efficiency: sensory efficiency is maintained if perceptual cues are used in standard ways, but conventional efficiency is maintained only if the viewer is practiced in interpreting the particular convention. Efficiency may be reduced if sensory cues are used improperly. However, there can be much to gain if conventions are challenged [89].

Wainer [257] finds it difficult to understand why so many "bad" graphics still abound, even two hundred years after Playfair. The works of Sicard and Marcks [225] and Carswell *et al.* [37] have identified some clues by having focused on the context in which the images are interpreted. Specifically, Carswell *et al.* [37] indicate that connotation is as important as denotation at least in certain cases: graphs which showed a third dimension on column charts were more popular because the graphs, and their producers by association, seemed more modern – even though the more modern graphs are likely harder to read accurately.

The image, in whatever form, is the viewer's interface to the data. Though the producer and viewer of an image have a stake in the communication of insight, this alone may not be sufficient to ensure the viewer receives the producer's message from the image. Even for a very simple image, it is not possible to fully articulate for the viewer everything that has gone into it. Some researchers in visualization (see, for example, Rogowitz and Treinish [210]) use the term *metadata* to refer to this information about the data and the visualization problem. The existence of this background, or pre-understanding [269], which is not always available to the viewer, highlights the need to consider the context in which scientific images are created [225]. Even when the same person is producer and viewer and has access to all the metadata, it still may be difficult to ensure that insight is communicated. If an image does not facilitate insight, it is not effective for that viewer.

The uniqueness of each viewer would seem to preclude an optimal, objective method for image selection which is applicable for all viewers. It is still possible, though, to apply the criterion of efficiency to the graphs that a particular individual will view. The question

for each viewer individually is then how to find, or search for, an efficient visual representation of the data. Following the evolutionary epistemology of Campbell [32], one should iteratively create variations, select from them, and retain what is useful from them. One need not find the *most* efficient visual representation, even if it exists. Rather, one can be concerned with those visual representations which satisfy most criteria (Simon [226] called these solutions *satisficing*). Over time, it is quite likely that one's needs from the data, and thereby the visual representation which will satisfy them, will change [174]. However, even with such simplifications, the activity of visualization will benefit from the application of computer technology.

1.2 Computer-aided visualization

To visualize is to "form a mental image of, imagine" or "to make (something) visible to the eye" [190]. Though the computer is not essential to this activity, it can be of substantial aid. According to Haber and McNabb [100], visualization is profitably viewed as a process which takes data, from whatever source, and transforms it into a displayable image. Bertin [11] described comparable transformations in his model of visual decision making, comprising matrix analysis of the problem (questions are defined); graphic information-processing (answers are discovered); and graphic communication (answers are communicated). For Bertin [11], it was clear that this work could never be automated because no machine would ever be able to solve this problem of imagination.

In 1992, Jessup [127] reported that practitioners of computer-aided visualization agreed that it involved computational graphics but they differed as to whether it is "a mental process aided by powerful computational tools, the computer processes that create the images, or the computer images themselves." These relate to the ways in which a user might employ a computer for visualization (adapted from the classification scheme of Kochhar *et al.* [137]):

Manual corresponds to a view of the computer as machinery operated by the human. In this mode, the user must handle all the details necessary to have an image displayed.

Automated corresponds to a view of the computer as a black box which will produce the correct output for the specified input. In this mode, all processing is handled by the computer.

Augmented corresponds to a view of the computer as a contributing partner. In this mode, the user and the computer share tasks based on the abilities of each.

Engelbart [66] viewed the augmentation of man's intellect as "increasing the capability of a man to approach a complex problem situation, gain comprehension to suit his particular needs, and to derive solutions to problems." Exemplars of the modern application of this principle are found in the area of Computer-Supported Cooperative Work (CSCW) [95]. For Winograd and Flores [269], "the ideal of an objectively knowledgeable expert must be replaced with a recognition of the importance of background. This can lead to the design of tools that facilitate a dialog of evolving understanding among a knowledgeable community."

One always wants to find the best possible visual representation for one's self, given the task at hand. If such a visual representation is not immediately available, for whatever reason, some search amongst alternatives must be done. Manual approaches will leave the user unsupported, automated approaches will carry out this search without direct specification from the user, and augmented approaches will involve the user in some way. Generally, a search can be made locally or globally, with different computer algorithms suited to each. For global searching, a genetic algorithm can be very effective, especially when the user is in control of the evaluation phase by interactively providing the objective function [231]. Different implementations of this approach are known collectively as *interactive evolution*.

1.3 Motivation

This work began from specific efforts to create imagery to aid the study of numerical methods [115], dynamical systems [111], and fractals [109, 116]. Fractals, in particular, have proven to be a prototypical example of the power of mathematical visualization.

Hanson *et al.* [104] defined mathematical visualization to be "the art of creating concrete understanding from abstract mathematical objects and their transformations." Such a definition can well be applied to all types of visualization. However, the importance of visualization, imagination, and intuition is often poorly communicated, particularly in mathematics. In the inaugural edition of the *Journal of Experimental Mathematics*, Epstein and his colleagues [68] addressed this issue: "Experiment has always been, and increasingly is, an important method of mathematical discovery. (Gauss declared that his way of arriving at mathematical truths was 'through systematic experimentation'.) Yet this tends to be concealed by the tradition of presenting only elegant, well-rounded and rigourous results."

The development of a new paradigm for using computers in support of visualization grew out of a simple question: how can (mathematical) visualization software tools be made directly accessible to the practitioner? Winograd and Flores [269] indicated the philosophical direction: "We can create computer systems whose use leads to better domains of interpretation. The machine can convey a kind of 'coaching' in which new possibilities for interpretation and action emerge." Beginning with Sims' work on artificial evolution in computer graphics [231] and inspired by Engelbart's earlier work on augmentation [66], a vision of a computer-aided visualization tool emerged. The *cogito*[1] system as it now stands is a robust realization of the core of that vision.

1.4 Thesis Statement

This work addresses the general question of how a computer can productively assist the human intellectual activity of visualization. Specifically it examines how software can be designed to provide direct access to images for anyone, regardless of computer expertise. Direct access to images may be a key in advancing the democratization of visual thinking, which Jessup [127] saw as a benefit to be realized from computer-aided visualization.

As part of a general movement which considers how best a computer can aid human intellectual activities, the solution described herein is unique both in its application and its conception of the potential relationship between human and computer. Specifically, it gives the user full control over all available visual representations in a structured way so that the user is not overwhelmed.

The *cogito* system is not intended to make any interface easier to use, nor any image simpler to interpret. Rather, it is intended to give people meaningful access to alternatives, from which each person can choose. Although the concept is straightforward, the implications are far-reaching, because each user has the capability of conducting systematic exploration in search of insight. Though some form of local exploration is commonly supported, the same is not true for global exploration. In the *cogito* system, this powerful ability is provided through a genetic algorithm that is controlled by the user in an easy and direct way.

The thesis of this work is that insight is encouraged by a software system which supports exploration and involves the user in image creation. The *cogito* system was designed and

[1] The name is a reflection of the process of human thought: to think, to shake up.

built both to gain experience with the practical implications of this paradigm and to validate it, through a user study of the software system.

1.5 Outline

The dissertation continues in Chapter 2 with a discussion of computer-aided visualization and the visualization process. The respective roles of human and computer are examined through examples of several current software systems.

Chapter 3 details the foundation, primarily from the areas of problem-solving, semiology, and human-computer interaction, upon which the implementation has been built. From these areas, several design principles for a computer-aided visualization tool are presented.

Chapter 4 describes the design and implementation of the *cogito* system, both in terms of the structures used and the implications of the choices provided to the user. The original descriptions of this project [113, 114] have been validated through implementation in a usable prototype system, called *cogito*.

Chapter 5 highlights the use of *cogito* to build two separate example applications.

Chapter 6 describes a user study conducted to evaluate the underlying system concepts and the access provided to alternative representations. The same applications developed in Chapter 5 were used for this study.

Chapter 7 describes what has been accomplished in the development and implementation of the *cogito* system, and the avenues for further work.

Chapter 2

Computer-Aided Visualization

Of all the areas to which computers provide substantial aid, visualization is conspicuous because of its visual nature. Interestingly though, the adjective "computer-aided" is applied far less frequently to visualization than design, for example. This chapter will explore the use of computers to aid visualization and its supporting activities.

2.1 Definition

The 1987 *Visualization in Scientific Computing* report by McCormick *et al.* [163] has been a catalyst for a great deal of research in scientific visualization over the past dozen years, expanding upon a tradition which can claim roots back to the tenth century [47]. That report unequivocally states the importance of visualization: "the ability of scientists to visualize complex computations and simulations is absolutely essential to ensure the integrity of analyses, to provoke insights and to communicate these insights to others." Although the purpose of visualization is clearly insight, it is difficult to find a concise definition of visualization in the modern era.

Haber and McNabb's 1990 paper [100] exemplifies the usual, data-centric, view by defining scientific visualization to be the "use of computer imaging technology as a tool for comprehending data obtained by simulation or physical measurement." A more user-centered outlook is signalled with Blinn's assertion [127] that the human mind is the final destination of any visualization. From this perspective, visualization is a human activity and the purpose of visualization software is to foment insight in its users.

These two aspects of visualization are distinguished in many different ways. In *A Critique*

of Pure Reason, Kant [170] did so by separating visual imagery generated from scientific theories from that which is abstracted from perceived phenomena. The data-centered aspect is concerned with making images from simulation data, whereas the user-centered aspect is concerned with making sense of the images and gaining intuition about them. The dictionary [190] expresses these two sides as "making something visible to the eye" and "forming a mental image." However, as "the art of creating concrete understanding" [104], visualization can be an involved process which may not always yield insight.

In their introduction to *Readings in Information Visualization: Using Vision to Think*, Card, Mackinlay and Schneiderman define [computer-aided] visualization as "the use of computer-supported, interactive, visual representations of data to amplify cognition" [35]. It is clear that computer-graphics can make many otherwise unseen things visible to the eye. Computer support for the cognitive aspects of visualization is a current research topic, with which this dissertation is chiefly concerned. It also has a considerable past, which is discussed in the next section.

2.2 Historical Development

The history of computer-aided visualization begins before the development of computer graphics. It starts with the very first use of computers when much of that computer technology was devoted to solving mathematical problems. The new computational power enabled a new experimental approach [256]. Whether or not the results of these calculations were used by the researchers to form mental imagery, insight and intuition were encouraged. As numerical methods allowed the solution of more complex problems, graphical displays provided more powerful ways to develop intuition about those problems and solutions.

It was at about this time that Vannevar Bush [29] described his vision of an information appliance, called the Memex, which would support humans' cognitive activities. That article would serve to inspire many who came after him.

As computational power was directed towards the creation of images, new tools with new interfaces became available to scientists. In the late 1950's, the United States' military was involved in the Semi-Automatic Ground Environment (SAGE) air defense system [157], which featured computer graphics in its human-computer interface and notably introduced the light pen. In 1963, Sutherland's Sketchpad system [240] introduced the concept of conversation between man and computer through the medium of line drawings. Also at about

this time, General Motors was involved with development of its DAC/1 (Design Augmented by Computer) system [157], which became a major effort in computer-aided design. All these early systems helped to revolutionize the way that people used and interacted with computers. Computer-Aided Design (CAD) remains an exemplar of the very effective use of computation and computer graphics.

New display techniques and representations suited for the capabilities of the computer began to appear from diverse areas like crystallography [56] and statistics [3]. Beyond particular visual representations, software and hardware systems were being developed which came to embody new ways of working with computers. Two pioneers in this area were Engelbart [66] and Culler [51]. Each, in their own way, shaped the systems which are becoming commonplace only today.

The 1987 *Visualization in Scientific Computing* report [163] focused primarily on developing capabilities for computer-generated imagery as a means to deal with an explosion of data from many new sources of increasingly high volume. Zabusky from physics [275, 277] and Tukey from statistics with his 1977 book entitled *Exploratory Data Analysis* [251], are two prominent examples of more inclusive approaches towards the use of computers in scientific discovery.

As tools become more widely available, more people from different disciplines are discovering computer graphics, as indicated by the increasing breadth of applications reported in the literature each year. The availability of computational graphic tools is changing how people use the computer as a tool. The ways in which people approach visualization with tools, and the tools themselves are discussed in the following sections.

2.3 The Visualization Process

At the time of the *Visualization in Scientific Computing* report [163] in 1987, the reality of the visualization process was very much influenced by hardware. In 1988, Fangmeier [70] presented a view of the visualization process which clearly reflected the lengthy computation and demands of the video production process. Computational power continues to grow a remarkable rate. As the access to computational tools has improved, the visualization steps are becoming better integrated into applications and the quality of "working visualizations" [164] approaches that previously reserved for publication.

In 1990, Haber and McNabb [100] presented a process view of visualization with a theoretical orientation. They begin with the consideration of a simulation process comprising modelling, solution, and interpretation/evaluation phases. The modelling phase is the most creative, since the scientist must make crucial decisions regarding how a loosely stated problem can be transformed into something which can be handled computationally. The solution phase includes making the appropriate approximations to the idealized mathematical model obtained in the modelling phase. Visualization and interaction can be useful at this stage both to define and evaluate the discrete models used in computation. The final phase of interpretation and evaluation provides the opportunity to consider any changes which might be needed in the modelling phase after which the simulation can be run again. Visualization can be invaluable as an aid to this evaluation. Haber and McNabb think about the simulation process as a collection of abstractions implemented through transformations. They apply this same view to a more detailed definition of the visualization process, where the transformations convert raw data into an *Abstract Visualization Object* (AVO) and finally into a viewable image (see Figure 2.1). The details of the data and transformations which finally result in the image are encapsulated in the notion of a *visualization idiom*, analogous to the definition of an idiom as "a group of words established by usage as having a meaning not deducible from those of the individual words" [190].

Although Haber and McNabb [100] present the notion that a particular visualization idiom may be tuned, they do not really consider evaluation/interpretation of the visual representation apart from the larger simulation process. Upson *et al.* [252] present a similar model:

- **filtering** simulation (or other) data into a more informative or tractable form
- **mapping** resulting data into geometry to be rendered
- **rendering** geometric data into images

This is a general model which can be adapted to statistics, where Cleveland [44] puts emphasis on the need for both graphing (mapping and rendering) and fitting (filtering) for data analysis.

These examples of process pertain to a single visual representation which is used within the larger context of scientific discovery. Nielson [177] demonstrates the power of incorporating scientific visualization into the iterative process of scientific discovery (see Figure 2.2).

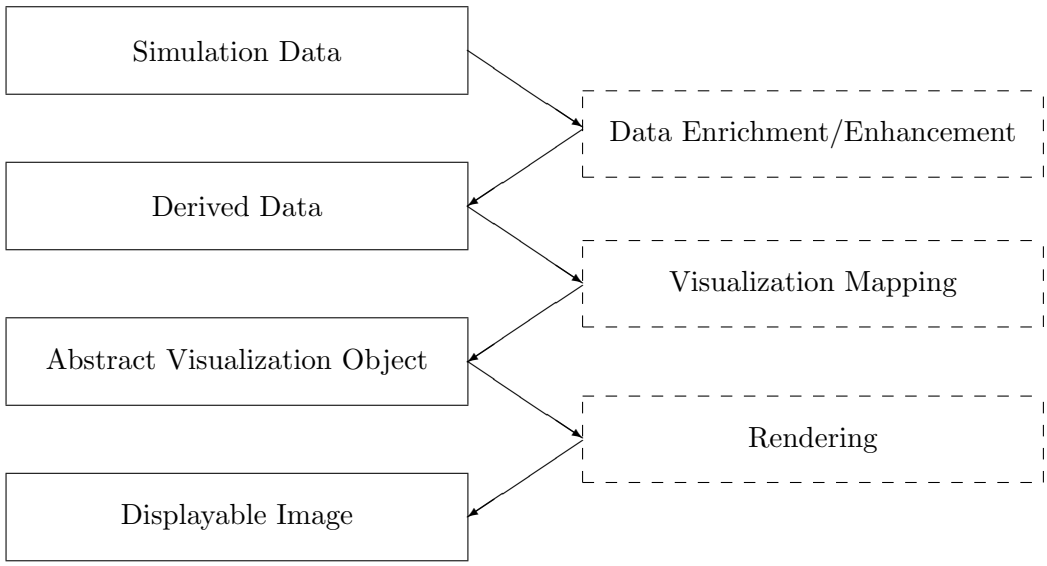


Figure 2.1: Haber and McNabb's model [100] of the visualization process in terms of abstractions (solid frames on the left) and transformations (dashed frames on the right).

Haber and McNabb [100] indicate three different ways to situate visualization in the regular activities of scientists:

1. post-simulation analysis (completely in interpretation/evaluation phase of simulation)
2. runtime monitoring (nominally in solution phase of simulation)
3. interactive steering (fully integrated into solution phase of simulation)

Springmeyer *et al.* [234] lament that visualization tools are more often directed at producing images rather than supporting researchers in gaining insight. In response, they suggest a new basis for future research which takes a more complete accounting of the activities involved in analysis (see Figure 2.3). Their separation of analysis into the data-centric branch of "investigation" and the user-centric branch of "integration of insight" corresponds to the aspects of visualization.

Within any visualization process, there are the issues of how to deal with data, how to select a visual representation to depict the data, and how to evaluate the effect of those

decisions.

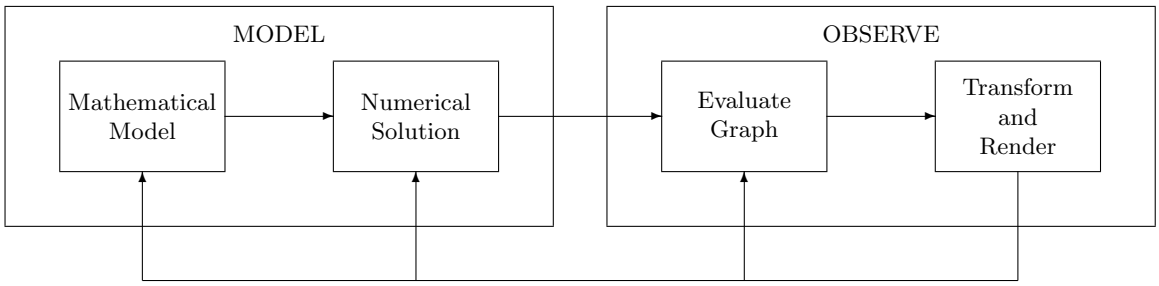


Figure 2.2: Information flow model described by Nielson [177], expanding on the basic loop of scientific discovery.

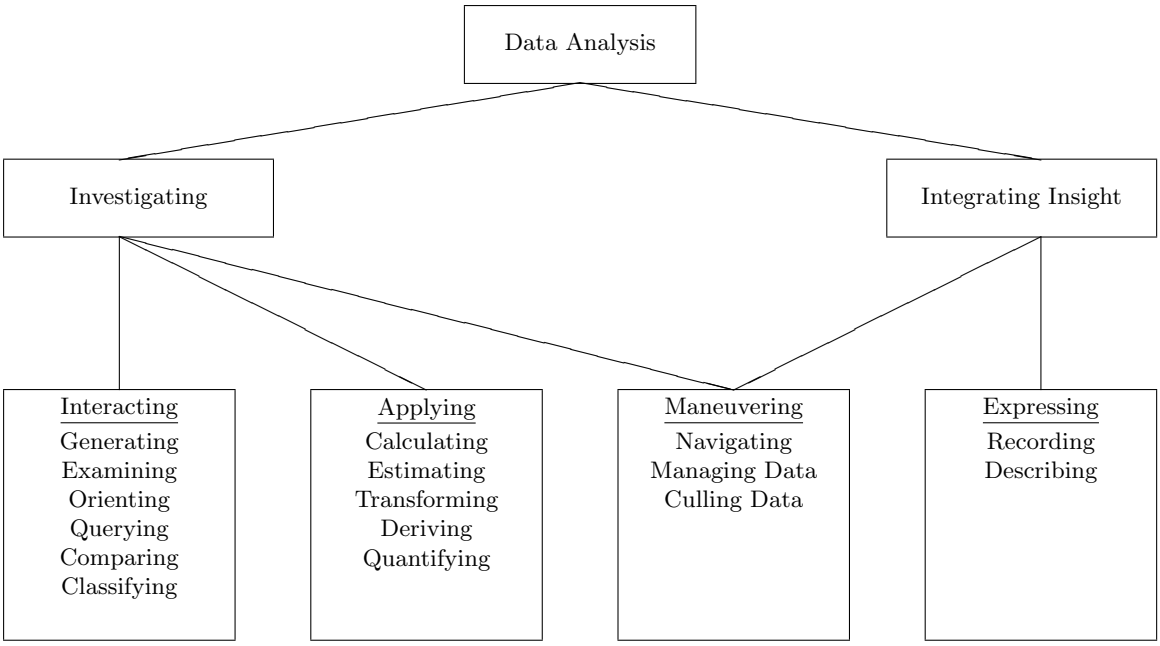


Figure 2.3: The decomposition of scientific data analysis into its process elements, after Springmeyer *et al.* [234].

2.3.1 Data

The desire to comprehend data is the primary force behind scientific visualization. Only in the cases when the amount of data is small and the computations are relatively simple, does one have a real option about whether or not to use a computer. Treinish [246] cited the ten terabytes of data per day output of NASA's (the National Aeronautic and Space

Administration of the United States of America) Earth Observing System (EOS) as a prime example of the need to handle huge quantities of data. Issues of data content, such the methods to easily handle a wide variety of data types, are also important [99].

In an experimental situation, the first issue is to decide which phenomena to measure and which data to collect. The importance of this phase is related to the expense and repeatability of the experiment; whereas it is fairly straightforward to make adjustments and repeat a numerical simulation which can be run in minutes, it is another matter when the time frame is weeks or months. Once data is collected, it must be made accessible before decisions about how to process it can be made. The data model provides the logical structure which defines the interface between the physical representation on disk and the visual representation on screen or film. Such data models may be unified, with a single, generalized data model and an Application Programming Interface (API); diffuse, using multiple models and multiple API's; or focused, with a single domain-specific data model which is not sufficiently abstracted to have its own API [246].

In their preface to the proceedings of the *Database Issues for Data Visualization* workshop from the IEEE Visualization '93 conference, Lee and Grinstein [145] remark that "*data-centric* visualization is necessary to allow researchers to interact with data, aided by visualization" and "we believe that as visualization systems evolve, they will begin to look more like database management systems (DBMS) with advanced visualization capabilities." Data, and metadata, needs to be accessible [246]. Therefore, effective data management techniques are essential for scientific visualization.

The process of transforming data into a visual form is not always a straightforward process because several decisions must be made. For example, it is possible to filter or otherwise transform the data but this must be done carefully because the revised data may tell a different story, with potentially tragic results [257].

2.3.2 Visual Representation

The visual representation is the means used to graphically depict the data to the user. It may comprise several issues, such as geometry, colour, and viewing projection. The actual construction and display of a representation is a fairly mechanical matter compared to a user finding an effective visual representation. Beyond storage of an image file, there are new formats for output, such as OpenInventor [264] and VRML [106], which can help to encode more information about a representation.

For computer scientists, much of the work in visualization is focused on the precise mapping or interpretation from data to representation [145]. In developing an effective visual representation, a data-centered approach will look to the data for clues whereas a user-centered approach will look beyond the data to consider the user. The more general issue underlying both approaches is whether there is a firm basis which can be used to guide the selection of an efficient, or *most* efficient visual representation to evoke insight in its viewer.

As the availability of affordable software tools and methods for output to video and other media improves, a user has much more flexibility to explore alternative representations in a meaningful way. It is not difficult to construct a very large number of different visual representations for some particular set of data. As Bertin [12] wrote, this would require "more patience than skill."

Robertson [208] notes that "the appropriate representation can provide the key to critical and comprehensive appreciation of the data, thus benefitting subsequent analysis, processing, or decision making." Providing appropriate representations and relevant techniques for particular problems is still very much a matter for research. The ideas of 'appropriate' and 'relevant' are embodied by the adjective *apposite* [190], which, however, does not provide any measurable criteria. In fact, determining if a representation is apposite requires some consideration of how and by whom the representation will be used.

Casner [38] says that there is no such thing as a "best" representation without considering how the image will be used. Clark [43] has presented a characterization of uses of imagery, ranging from data storage to communication (see Figure 2.4).

Sicard and Marck [225] distinguish cognitive, didactic, and aesthetic logics in scientific pictures, which are not separable without knowledge of the author's intent. For them, scientific pictures are "imbued with the 'view' of the author which claims to be objective. But, in fact, it is attached to 'thought history', technological history, scientific history and is marked by aesthetic choices, cultural bias, and perceptional practices."

Although there is much excitement [21] about three-dimensional graphic techniques such as iso-surface construction [152, 273], two-dimensional graphics are quite interesting in their own right. Many of the interesting and useful techniques for data display now commonly in use were developed long before computers [47, 257]. Wainer [257] presents an interesting comparison between the pie chart, developed by Playfair [198], and the Nightingale

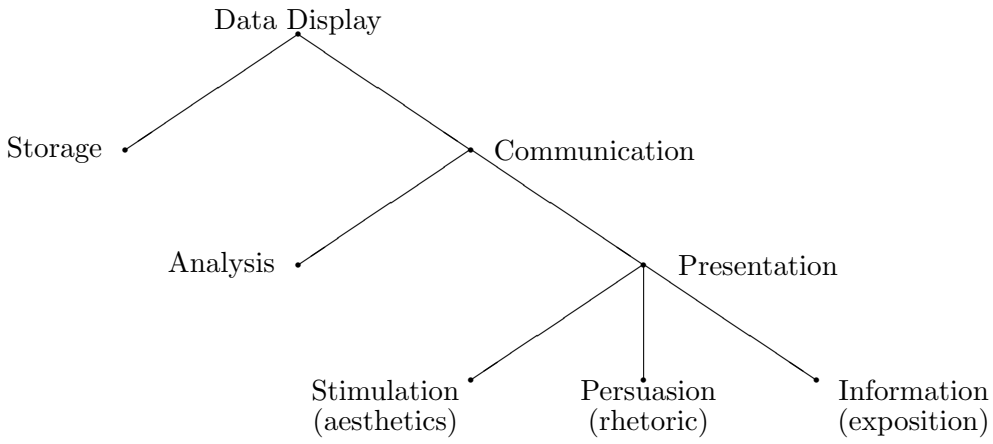


Figure 2.4: Organization of data displays (after Clark [43]) according to primary function, then intended use, then presentation goal. The parenthetical comments indicate the governing principle applied to each type of presentation.

rose [12, 257], first used by Florence Nightingale. Trilinear plots [12, 257] and various nomograms [133], first developed in 1899, require varying amounts of computer time to calculate and somewhat more time to learn as well. With so many types of graphs available, it may be difficult for the user to select between them. The experience and training of the user will be a definite factor, and some users will choose not to expend the effort to learn to read a new visual representation [38].

According to Wainer [257], the medium of graphics is so powerful because facility with pictures is such a prevalent trait across cultures. Messaris [168] agrees and makes the important distinction between facility with pictures and facility with the particular interpretations imposed by a culture. Messaris' defines "visual literacy" as something that enables exposure of the media and the messages which it contains. One can see the utility of a notion of "visualization literacy", which would work to expose the ways in which visual representations can be made to mislead.

Whether or not graphics can permit a monosemic interpretation, the tremendous contribution by Bertin [12] lies in his systematic exploration of marks on the plane, by considering

components and categories of retinal variables, and how those marks can be used to construct diagrams, networks, and maps. There can be many complicating factors, some of these are discussed by contemporaries such as Schmid and Schmid [219], Tufte [248], Cleveland and McGill [45], Tukey [251] and Wainer [257]. There are, however, other ways to think about images [15].

2.3.3 Evaluation

However one has arrived at a visual representation, the effectiveness with which that representation can be used to communicate its message is of primary importance. Development of a visual representation from data is done with a focus on the data, the interpretation of that visual representation relies on the user. It is important to ensure that the user can develop sufficient confidence in his or her interpretation of the image [209].

2.4 Computerization

The preceding section identified three key parts of the visualization process. Most directly, the computer can be applied to render the visual representation for some set of data. The computer may also be used to calculate and manage the data, and to evaluate the effectiveness of the visual representation. The choice of whether, or to what extent, each of these three parts is handled by the computer helps to describe the different approaches to computerization.

The mere fact that a computer is used to aid the visualization process does not mean that visualization has been democratized. Each choice for computerization of the visualization process requires competencies from the user. This may involve anything from learning a simple graphical interface to a complex programming language.

Following from the classification developed by Kochhar *et al.* [137], the design of visualization software can be classified as either manual, automated, or augmented. Manual systems require complete involvement of a human to construct a visualization application; automated systems require no involvement of a human; and augmented systems support some notion of the visualization process as a collaborative effort between human and computer.

Each of these system design choices also represents an understanding about the problem of articulation. In general, both manual and automated systems rely on a sufficiently

complete articulation of the problem such that a useful solution can be found. Augmented systems take a view which is closer to that of Winograd and Flores, who said, "The effort of articulation is important and useful, but it can never be complete" [269].

Just as is true for computer information systems [2], the design of a computer-aided visualization tool may be either data-centered or user-centered. In the present case, the distinction is based on whether the visual representation is determined by the data to be depicted or by the user who will view it.

What follows is a discussion of several systems which indicate the breadth of research in this area, grouped according to the way in which they relate human and computer.

2.4.1 Manual

Software belonging to this category requires the user to completely describe and control the operation of the visualization application. Licklider [149] saw that this represented the idea of a mechanically-extended user [181]; the software is an extension of the user's control. This control can be exerted at a variety of points. If the average programmer needs to deal with issues at a low-level, he or she may have less opportunity to make the application very general. At the most basic level to be undertaken today, one might choose to write code directly with a graphics library such as OpenGL [270] and a windowing system such as X Windows Motif [108].

The first efforts to provide high-level capabilities in visualization software systems came from the animation production Environment (apE) [61] and the Advanced Visualization System (AVS) [252], which were both developed during the same time period with little contact between groups. The design goals were:

- ease of use (through direct-manipulation interfaces)
- low cost
- completeness (in implementing the entire visualization process)
- extensibility
- portability

Both systems use a dataflow model accessed through a visual programming interface [267]. Cyclic dataflow graphs allowed these systems to be used for computational steering [162].

This software design approach has become dominant in the scientific visualization community. Systems like apE and AVS, and the more recent entries like Khoros [201], Iris Explorer [63] and IBM's Data Explorer [153], are collectively known as Modular Visualization Environments (MVE) [31]. MVE's are characterized by:

- modularity, which includes building blocks of code, an object-oriented approach, and user-extensible libraries
- an emphasis on visualization, possibly as a means of prototyping
- programming environments which allow integrated and interactive program creation and manipulation

The GRASPARC [22] system uses the basic MVE architecture and adds additional capabilities by providing support of a more complete data analysis cycle. A large component of this support is through a history mechanism. Such a history mechanism was key for Patten and Ma [189] which allowed them to meaningfully represent the results of their visualization efforts.

The OpenInventor [264] toolkit provides a high-level interface to OpenGL [270] and relieves the programmer of many details of the graphics. The toolkit works with the notion of a scene graph, which encapsulates the ideas needed to create an image and can be constructed from many built-in objects and actions. Images are displayed by traversing the scene graph. OpenInventor closely follows the state-based approach of the underlying OpenGL library and is therefore not truly object-oriented. It does provide facilities for input and output and data-handling, and is extensible. The application's interface remains the responsibility of the programmer and requires separate coding in a native windowing system such as X Windows Motif, or the simpler platform-independent Graphics Library Utility Toolkit (GLUT) [270]. This former approach was used in previous work by Hepting *et al.* [111, 115].

The Visualization Toolkit (vtk), first described in 1996 by Schroeder *et al.* [222], is more specialized towards visualization tasks than the OpenInventor toolkit and features a more complete object-oriented design. Although many of the MVE's profess to use an object-oriented design, Schroeder *et al.* [222] found them difficult to use apart from their graphical environments and in response designed vtk with a dataflow model. Their design

goals included implementation of a toolkit philosophy to permit complex applications to be built from small pieces, provision of an interpreted language interface, and simplicity.

Another trend has been towards graphical environments for more specialized purposes. The EXVIS system for exploratory visualization described by Grinstein *et al.* [97] was intended, amongst other things, to provide a testbed for the design, execution, and analysis of perception experiments as a means to study new visualization technologies. The DAVID (DAta, VIsualization, and Diagnostics) system, described in 1990 by Bitz and Zabusky [14], provided some of the same features as the early MVE's, but with a more limited programming capacity. The system embodied Zabusky's ideas of "visiometrics" [277], defined as "the process of visualizing, quantifying, and mathematizing amorphous objects observed in time-dependent multidimensional data sets" [14]. The SCIrun [188] system, reported in 1997, is a scientific programming environment specialized for the steering of large-scale computations and, like MVE's, it employs a dataflow programming model.

DataDesk, the statistical graphics package first described by Velleman and Pratt [254] in 1989, provides a direct-manipulation interface to statistics. Theus [242] relates this, and other statistical packages, as examples of Tukey's Exploratory Data Analysis [251]. It builds on the idea that multiple, connected views of data can greatly enhance the power of data analysis. For Velleman and Pratt [254], graphical interfaces are seen as ways to specify "like this one, only different in the following ways." Insight, however, is acknowledged as important.

MATLAB (see, for example, Nakamura [173]) began as a specialized environment for mathematics, which can trace its roots to Culler's On Line System [51], and it has grown to include substantial general computer graphics support. It has its own high-level language, and interfaces to FORTRAN and C/C++.

The Spreadsheet for Information Visualization (SIV) [42], based on work presented by Levoy [148], is a novel use of the spreadsheet programming paradigm. The focus provided is on operators rather than operands. The user can explore the effect of the same operation on several related images. The interface does not employ direct manipulation as in the visual programming paradigm described earlier. However, it is still a style of visual programming.

In 1992, Treinish [247] noted that there was still a need for programming help even for the visualization-literate scientist. Despite continual advances in the programming interface provided to researchers, programming may always be a necessity. Many feel that this is too great a burden to place on the scientist or researcher, who will either have the programming

done by someone else, or abandon the system and perhaps the practice of scientific visualization altogether. The use of a staff programmer is not an ideal solution to this problem, since the scientist becomes removed from his or her data, making communication about it that much more difficult. Rogers [209], for example, has proposed a solution which would shift the burden of programming away from the user. As the interface moves closer to the problem and further from the hardware, designs can become less data-centered.

2.4.2 Augmented

Augmented systems aid the user by allowing certain well-defined tasks to be performed primarily by the computer, with the effect of increasing the capabilities of people to tackle complex problems. West [265] sees such augmentation as the best response to Weiner's 1948 prediction [261] that the computer revolution is "bound to devalue the human brain, at least in its simpler and more routine decisions":

> ... we can clearly see that some time in the not-too-distant future, machines will be the best clerks. Given this situation, we must learn, as teachers and workers, to maximize in ourselves and in our students what is most valued among human capabilities and what machines cannot do. It seems clear that, increasingly, many of these most valued skills will involve the insightful and broadly integrative capabilities often associated with visual modes of thought, skills that can perhaps be taught and exercised most effectively using graphics-oriented computers.

In 1960, Licklider [149] also saw that the realization of what he called a "man-computer symbiosis" could mark a very intellectually-stimulating era. Today these ideas are embodied, for example, in work on problem-solving [79], collaboration [95], and performance [129].

In 1986, Dawkins published the *Blind Watchmaker* [55], along with his biomorph software, the first example of interactive evolution. Although not so strictly an example of scientific visualization software, it does enable the user with the capability of visual exploration. Following on this work, Sims [231] presented a method for the use of artificial evolution in computer graphics which employed both a genetic algorithm [91] and genetic programming [140]. Both of these "genetic" methods work by simulating the cellular-level processes of cross-over and mutation. The former does this as a means to search a space whereas the latter works to transform the space itself. For Sims, the goal was to evolve

images and textures. Peterson [193] added more control to Sims' original method by allowing the program which generated an image to be modified. This approach is very powerful since the user can be directly involved with images, which allows a very user-centered approach. However, because it is surprising to see images from different generations with no apparent connection between them, it can defeat the user's control. In that case, one is really faced with a dilemma of how to make sense of the diversity of images. In 1992, Todd and Latham [245] also discussed a genetic approach to the generation of new images, theirs being more restrictive by not including genetic programming. Boden [17] argues that the latter is more relevant to artistic creativity because it allows a more deliberate and disciplined exploration of alternatives. This sort of approach can be very effective when the computations can be done at rates sufficient to maintain interactivity.

Rogowitz and Treinish [210] describe a rule-based visualization architecture which is a modification to the standard model of visualization processes. Such rules can make the visualization higher-level because of its characterization of the data and human-centered rules. The user can choose to employ the rules or not, the interface is the same either way. In this way, the user is more in control and can avail him or herself of the rules as need be.

The VISTA (VISualization Tool Assistant) environment described by Senay and Ignatius [223] attempts to aid users with support for five types of visualization knowledge:

- data characteristics
- visualization vocabulary
- primitive visualization techniques
- composition rules
- rules of visual perception

The system will analyse, as much as possible, the input data and suggest a visualization. At that point, the user can make changes or specify further constraints on the design before rendering the final image. Although one would like to ensure the "accurate interpretation of the image", such a goal seems beyond the control of any such system however detailed. Even after perceptual issues are addressed, there are still a lot of free choices. This system is considerably biased towards the "automated" end of the "augmented" spectrum but is

considered in the "augmented" category because it allows the user to adjust the final image produced.

The SageTools [211] system allows users to work in a context of past graphics and allows them to modify what has already been done. This is a more user-centered approach but it is constrained by the examples which are in the system.

The Integrated Visualization Environment (IVE) [136] is an implementation of the co-operative computer-aided design (CCAD) paradigm [134]. It uses a generative approach by applying an appropriate grammar. The user can intercede after each iteration to select promising designs for further development. The CCAD paradigm forms, along with the work of Dawkins [55] and Sims [231], an important aspect of what constitutes interactive evolution.

Design Galleries [161] works to provide a sampling of the whole range of alternatives, based on the principle of maximal diversity. The user specifies a metric function which is used to evaluate the pairwise diversity between alternatives. The system works offline to generate alternatives and test them with the supplied function. Similar alternatives are grouped and representatives are displayed.

GADGET (Goal-oriented Application Design Guidance for modular visualization EnvironmenTs) [83] increases the accessibility of the MVE paradigm by reducing the requirements for the user to program for him or herself.

2.4.3 Automated

There has been a great deal of work in the area of automating the design of graphics. Much of that work goes beyond the scope of interest in scientific visualization. Two excellent papers which deal with these issues and the related literature are Kochhar *et al.* [137] and Fischer and Reeves [79].

As Licklider described, these systems might be more like humanly-extended machines where the human does the input and some other specifications work. These systems appear to the user as black boxes which are given input and produce output. The rationale behind them is that the number of alternative visual representations is so large that the user would be overwhelmed if he or she had to deal with the space in its entirety. In accepting this guidance from the computer, the user relies more on the computer for its application of design rules and gives up more freedom to exercise personal choices about what the visual representations will contain.

In 1986, APT (A Presentation Tool) by Mackinlay [158] contributed a formalization of the design rules for two-dimensional static graphs, based in part on Bertin [12]. It was prescriptive system because it chose graphics on the basis of expressiveness and effectiveness criteria.

In 1991, Casner added task information to his presentation system, called BOZ [38]. This resulted in a noticeable improvement in user performance with the graphs his system generated. The AutoVisual [13] system described by Beshers and Feiner in 1993 expanded earlier work by dealing with the design of animated virtual worlds for multivariate data visualization. This approach was more exploratory than Feiner's earlier work with APEX system [71], for example, which had the different objective of generating explanations.

2.5 Implications

The various implementations described here represent viable solutions for the particular aspects of the visualization problem which they address. There are still many concerns about the development of current visualization systems.

As noted by Springmeyer *et al.* [234], some of these manual systems, based on the filter-map-render paradigm do not include a complete model of the scientific data analysis process. The automated systems rely on a particular articulation of the design rules which can make them inflexible. In an effort to protect the user, they assume the role where the user would exercise his or her creative judgement. Augmented systems seem to be the most promising alternatives. The presentation here has touched on the important issues of how the different systems make use of the user's input to drive the development of visual representations.

As yet though, the difficulties remain substantial when it comes to finding effective visual representations: "There is really no practical way to decide, other than to make the same choices that others have made in the past, or to experiment painstakingly with a small number of the possibilities. This is an unsatisfactory solution" [81].

Gahegan [88] identified four obstacles to effective exploratory visualization tools:

- rendering speed: general approaches suffer in terms of rendering speed
- the effects of visual combination: though there is a great deal understood about human perception in particular situations, the perceptual effects of combining various factors are not known

- the size of the "solution space": considering all possible combinations or permutations leads to an immense space in which there is no guidance as to whether a good representation has been found.

- user orientation may be difficult because the idea of manipulating objects in a virtual space is difficult; construction of an appropriate legend or annotation is complex; and users need to be trained

The potential for an access problem becomes quickly apparent. How is it possible to manage the many possible visual representations of a data set and be able to access those which will encourage insight? Indeed, there are more choices beyond the representation of data, since each representation can be presented in a variety of ways [36].

The next chapter will explore whether, as Winograd and Flores [269] contend, that recognition of the importance of background can "lead to the design of tools that facilitate a dialog of evolving understanding among a knowledgeable community."

Chapter 3

A New Paradigm

Having established that the goal of visualization is insight, the task is to build a computer-aided visualization tool which encourages it. Following an examination of insight, problem-solving, and graphic communication, this chapter presents the design principles (**DP**) for such a tool.

Even after considerable study, insight remains somewhat of an elusive concept. An operational sense of insight is understanding, which can be readily seen as the goal of a problem-solving process. Visualization depends on the graphical communication of insight. Cartography will be used as the basis for a discussion of graphic communication, since it has a rich history and is almost universal in its scope. Thrower [244], quoting McLuhan [166], says: "the map is one of a select group of communications media without which, ... 'the world of modern science and technologies would hardly exist.' "

Petchenik [192] has welcomed a shift in cartography from an emphasis on the behavioural, focused on isolated perceptual results about how well people can read particular map symbols, to the cognitive, focused on how people understand whole maps. Scientific visualization has begun with a data-centric view, encouraged by the *Visualization in Scientific Computing* report [163] and studies of perception [96] have provided valuable knowledge. This dissertation focuses on how that knowledge can provide an effective tool.

3.1 Insight and Problem-Solving

The *New Oxford Dictionary of English* [190] defines insight to be "the capacity to gain an accurate and deep intuitive understanding of a person or thing." It is important to study

insight as a means of clarifying what, if any, explicit features a computer-aided visualization tool may provide to support its development.

A ready source of information about insight comes from personal accounts, which are quoted extensively in this section. That these people have benefitted from insight is clear from the work which they have produced. However, the mechanism of insight is difficult to discern from such subjective sources of information. Kekulé's famous account of the dream that led him to discover the ring structure of benzene, in which he saw a snake bite its own tail, was fabricated [170]. However, French mathematician Henri Poincaré is considered to be a much more reliable source. The following account deals with his revelation regarding Fuchsian functions [101]:

> Just at this time, I left Caen, where I was living, to go on a geologic excursion under the auspices of the School of Mines. The incidents of the travel made me forget my mathematical work. Having reached Coutances, we entered an omnibus to go some place or other. At the moment when I put my foot on the step, the idea came to me, without anything in my former thoughts seeming to have paved the way for it, that the transformations I had used to define the Fuchsian functions were identical with those of non-Euclidean geometry. I did not verify the idea; I should not have had time, as, upon taking my seat in the omnibus, I went on with a conversation already commenced, but I felt a perfect certainty. On my return to Caen, for conscience' sake, I verified the result at my leisure.

A more recent example is the story, recounted by Schooler and Melcher [221], is that of a medical graduate student, Yung Kang Chow, who discovered the first apparently successful method for eliminating the AIDS virus from cells in a test tube.

> I was reading during dinner, which is a bad thing to do . . . but I had to because I had so much to do that evening. I was thinking of ways to explain the phenomenon, and the idea just came to me in an instant. It was an inspiration, almost like 'Eureka', I was ecstatic jumping up and down and telling my wife that I think this was the most exciting thing I ever came up with because right away I realized the implications of the work.

Both of these accounts, and many like them, indicate that the moment of insight arrives

quickly and without warning. Its suddeness indicates that the process is, to a large extent, unreportable. There is increasing evidence, in part reported by Fallshore and Schooler [69] and Schooler and Melcher [221], which indicates that attempts to verbalize descriptions of such non-reportable phenomena may overshadow the original information. Especially for those whose verbal ability is not at par with their perceptual ability, the words of the description do not capture the observation and so the two diverge. Although one may not be able to describe something, one may still be able to identify it visually. For example, verbalizing the appearance of a previously seen face interfered with the ability to later recognize that face from a group of similar ones [220].

DP 1 *Support the user in working directly with images, as much as possible*

Perhaps, as Koestler [138] wrote, "Language can become a screen that stands between the thinker and reality. That is the reason why true creativity often starts where language ends." Hadamard reported the accounts of several mathematicians, including himself, who shared this view. The following quotation is from Einstein [101]:

> The words or the language, as they are written or spoken, do not seem to play any role in my mechanism of thought. The psychical entities which seem to serve as elements in thought are certain signs and more or less clear images which can be 'voluntarily' reproduced and combined.
>
> There is, of course, a certain connection between those elements and relevant logical concepts. It is also clear that the desire to arrive finally at logically connected concepts is the emotional basis of this rather vague play with the above mentioned elements. But taken from a psychological viewpoint, this combinatory play seems to be the essential feature in productive thought – before there is any connection with logical construction in words or other kinds of signs which can be communicated to others.

The combinatory play to which Einstein referred was quite tangibly experienced by Poincaré, who made the following observation: "One evening, contrary to my custom, I drank black coffee and could not sleep. Ideas rose in crowds; I felt them collide until pairs interlocked, so to speak, making a stable combination." According to the poet Valéry[1]

[1]T. S. Eliot once described Valéry as a poet in a lab coat. Regarded as one of the premiere poets of the twentieth century, he is most famous for *Le cimetière marin* and *Ebauche d'un serpent*.

> It takes two to invent anything. The one makes up combinations; the other one chooses, recognizes what he wishes and what is important to him in the mass of the things which the former has imparted to him.
>
> What we call genius is much less the work of the first one than the readiness of the second one to grasp the value of what has been laid before him and to choose it.

DP 2 *Support the user in combining images, and elements of images*

The commonalities amongst various accounts have led to a general, four-stage model of the process which leads to insight. This model was developed by Hadamard [101], and refined by Csikzentmihalyi and Sawyer [50].

1. preparation: study and analysis of data in a conscious, focused manner, preparing the raw material for processing by the subconscious. The preparation phase may occur before, or include, the examination of images.

2. incubation: occurs in the subconscious, can take seconds or years

3. insight: occurs when the subconscious combines or selects an idea which emerges into consciousness and results in a "Eureka!" experience

4. evaluation/elaboration: final conscious step which verifies the idea and elaborates it for presentation to others

The mechanisms at work during the incubation and insight stages are a matter of contention. They are of concern to this work only insofar as they manifest themselves in activities to be supported. For example, Hadamard contended that the subconscious is actively guiding the processing. Ohlsson [183] proposed the notion of restructuring, which is akin to searching for a new way to look at the problem. Weisberg [262] concedes that restructuring plays a role in some cases, though others may only require a reconsideration of alternatives. Simon [226] has the idea of selective forgetting; in the preparation stage, structures are built up in long-term memory and the problem details are kept in short-term memory. If a problem is abandoned, then the short-term memory is forgotten but the long-term memory structures are maintained, allowing a new tack to be taken. Langley and Jones [143] suggest

that the processes are more memory-based and that it really amounts to finding appropriate analogies.

However the phenomenon of insight is conceived, some form of searching is indicated. Most often, one considers that the solution exists in some space determined by the problem, and problem-solving is the search for the solution [176]. An exhaustive search is almost always completely impractical. Instead, human problem-solvers rely on heuristic methods which are not likely to find any optimal solutions, but rather to find an acceptable solution in a reasonable amount of time. Simon [226] calls these *satisficing* solutions.

DP 3 *Allow the user to employ heuristic search techniques*

As one searches for insight, it can be adopted as the goal of a problem-solving process. Csikszentmihalyi traces many of the current three-stage models to the evolutionary epistemology of Popper [199] and Campbell [32]. In those schemes, variations are created, then selected and possibly retained. Brainstorming [184] is included in this category.

DP 4 *Support a model of problem-solving*

A slightly different emphasis is found when considering design problems, which are close analogues to the visualization process. In these models, more emphasis is placed on the generation of ideas: one limits the design space, makes up combinations within this space and then evaluates. Researchers have observed that, especially in design, problems and solutions coevolve [79]. One usually does not, perhaps cannot, have a clear statement of the problem without an idea of the solution. Rittel [207] described this as follows: "you cannot understand the problem without having a concept of the solution in mind; and that you cannot gather information meaningfully unless you have understood the problem but that you cannot understand the problem without information about it." The evolutionary nature of the design process is well-described by the model of evolutionary epistemology [232]. Allen [2] uses this formulation, based on Neill [174], for information seeking.

According to Fischer and Boecker [76], "design is concerned with how things ("artifacts") *ought to be*, in order to attain goals and to function." As the solution evolves, it is necessary and desirable to keep a record of the design changes.

DP 5 *Record all aspects of the design of visual representations*

According to Perkins [191], a problem space can appear either as clue-rich (homing) in which the solution is evident, or clue-poor (Klondike) in which the solution must be found by prospecting. Insight can be viewed as the transformation of a space from Klondike to homing.

DP 6 *Support restructuring and reorganizing of alternatives*

Csikzentmihalyi and Sawyer [50] distinguish between problem-solving and problem-finding. Like problem-solving, problem-finding can also be construed as a search through a space of alternatives. One begins problem-finding with a less-clear problem at the beginning, and it can take longer to get through. Problem-finding has been connected to creative productivity and it may be distinct and more difficult than problem-solving, as described by Einstein and Infeld: "The formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill. To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science" [107]. Both problem-solving and problem-finding can be social activities, to which many people can contribute.

DP 7 *Support collaboration*

Boden [17] distinguishes between psychological and historical creativity, depending on whether it has significance for a society. Creativity at the psychological level is very important because one invents things for oneself. Hadamard [101] said that mathematicians may "prefer, when studying any previous work, to think it out and rediscover it by themselves. This is my approach, so that finally I know, in any case, of only one inventor, who is myself."

DP 8 *Encourage the user to discover things personally*

Boden [17] also discusses conceptual spaces defined by stylistic conventions, and how to expand those spaces with three rules: "drop a constraint", "consider the negative", and "vary the variable." On the one hand, constraints are necessary when the design space is large. However, to constrain or to rely on examples [75] too much might lead one to fixate too soon on an ineffective design [233].

DP 9 *Support the systematic exploration of a conceptual space*

The annexation of new territory in the conceptual design space may meet with resistance [194]: "The people with most influence on the future are those designing new visualizations today, who should be conscious of the conflict between their desire for unfettered originality and the value of widely accepted conventions."

DP 10 *Support the use of current solutions as the basis for future enquiries*

3.2 Graphics as Communication Medium

In Subsection 2.3.2, the importance of visual representations was discussed. Now, this section considers how those visual representations fit into the larger scheme.

Insight from images produced through scientific visualization is gained through the interpretation of those images and so graphics becomes the communication medium for that insight. A general model of communication comprises a sender who encodes and transmits a message and a receiver who receives and decodes the message. To ensure communication of the complete message, it is not sufficient that the sender and receiver have the same goal. Although Bertin [11] considers all graphic communication to be monosemic, in practice it seems that this will only occur for simple situations when the whole problem can be articulated through the legend to the graph.

Bertin's *Semiology of Graphics* brought a structure to graphics and cartography. Tufte [248, 249], Schmid [219], Cleveland [44, 45], Wainer [257], and Tukey [251] have all made contributions to the area of graphics, though not in such a systematic way. Bertin distinguishes three successive forms of graphic application in decision making:

1. matrix analysis of the problem (the problem is defined, the data table constructed, and questions are defined)

2. graphic information-processing based on the reorderable image (the graphic processing language is adopted, the comprehensive data is classified, and answers are discovered)

3. graphic communication (interpret and decide based on simplified data, answers are communicated)

For Bertin [11], it is the first step which requires the most creativity. Is it possible to support people in being creative? Csikzentmihalyi and Sawyer [50], discussing the requirements for creativity, suggest that motivation is a key factor for individual creativity; and for

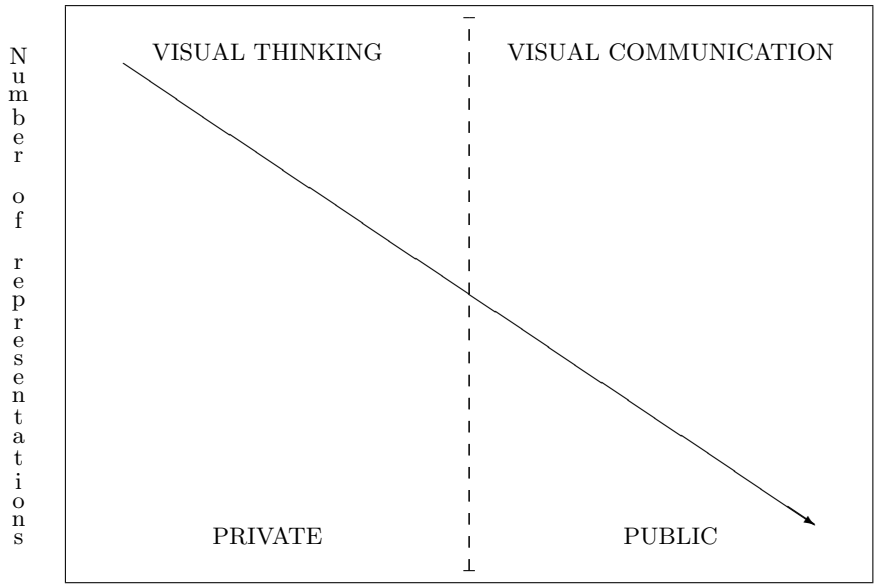


Figure 3.1: The relationship amongst factors in graphic communication, after DiBiase [156]

the motivated individuals a dialogue with the computer can help to formulate and articulate questions.

DiBiase [156] considers a continuum of the visual (see Figure 3.1), where thinking can be private with very many representations and communication can be public with very few representations. The decrease in the number of visual representations used for visual communication is likely the byproduct of someone preparing information for another to consume. In this age of interactive computer graphics, each person can be empowered to engage in visual thinking with many visual representations. The potential of interactivity is a very exciting aspect of computer graphics.

DP 11 *Support the user in choosing images*

DP 12 *Allow the user to interact with the images*

Blackwell and Engelhart [15] provide a taxonomy of taxonomies which gives a better context for monosemic models. Another alternative view of cognition in cartography [192] can be expressed in terms of Piaget's concepts of organism-environment interaction through assimilation and accommodation [40].

Ultimately, though, this work is concerned with empowering each user with images, such that they become participants in shaping the imagery instead of just receiving the graphic;

Human	Computer
holistic pattern matching	precision and repeatability
creativity	fast and accurate calculations
initiative and exception handling	reliable memory
ability to learn from experience	tirelessness
ability to handle ill-defined problems	objectivity
good motor skills	patience
judgement	physical robustness
sense of ethics and responsibility	
ability to apply social context to task	
ability to fail gracefully	
flexibility and adaptability	

Table 3.1: Respective capabilities of humans and computers, adapted from Baecker *et al.* [7].

so the graphic stops being an end in itself and becomes a "moment in the process of decision-making." The means by which the computer can help this process are examined in the next section.

3.3 Computer as Enabling Technology

The problem of articulation has been considered at various points in this dissertation. It would be best to leave the researcher to relate directly with the images which he creates. However, the researcher alone may have trouble dealing with an unfamiliar system in which he or she has difficulty formulating the commands necessary to produce a visual representation and difficulty interpreting the results of his or her actions [179].

In a "good" interface, according to Baecker *et al.* [7], human and computer "should augment each other to produce a system that is greater than the sum of its parts. Computer and human share responsibilities, each performing the parts of the task that best fit its capabilities. The computer enchances our cognitive and perceptual strengths, and counteracts our weaknesses. People are responsible for the things the machine cannot or should not do." The capabilities attributed to each are displayed in Table 3.1.

Visualization naturally has both data-centered and user-centered aspects. Between the extremes of the user managing all the details or the computer automatically generating solutions, there exist a wealth of possibilities for supporting the user in his or her interactions

with the computer [136, 161, 210]. The support which is offered depends on how the user is modelled, from someone who is overwhelmed by the immensity of the design space to someone who knows precisely what is needed for him or herself and can tell the computer to produce it.

At a general level, there are two issues related directly to the design of a visualization tool: support for individual differences and the allocation of work between human and computer.

Clearly, the computer can aid visualization beyond mere image production. The two goals of Licklider's human-computer symbiosis [149] were to bring the computer into the "formulative parts of technical problems" which might be too difficult for humans alone; and to create a direct link between human and computer, perhaps like that between colleagues, which would enable the computer's involvement in "real time" situations. One can see that these goals point to a computer system which enables human problem-solving, and problem-finding. The focus of the visualization process is the iteration performed in selecting and evaluating candidate visual representations.

The user will contribute his or her creativity to the equation. Schank's comments, though made with respect to computer systems are equally valuable for humans: "We are not proposing that people simply loosen the constraints they use when searching for and applying knowledge. A system [perhaps, user] that worked in this way would not be creative but would instead progress from schizophrenia (as it leaped from one random idea to another) to catatonia (as it found itself buried under a combinatorial avalanche of attempts)" [216].

In the context of cooperative human problem-solving systems, Fischer [79] observed several characteristics of successful dialogues between people working on problems. If that success is to be extended to dialogues between humans and computers, the design of computer systems must consider: natural communication which people often use (perhaps different from grammatically-correct natural language); multiple specification techniques for communicating ideas about a problem; mixed-initiative dialogues which allow the user to both direct the operation of the system and query the system about its actions; management of trouble when breakdowns occur; simultaneous exploration of problem and solution spaces; how a user operates within the physical world to find sources of information and extend his or her knowledge and reasoning; and how a user benefits from distributed intelligence, in a cooperative process or collaboration. Consideration of all these factors will lead to a system

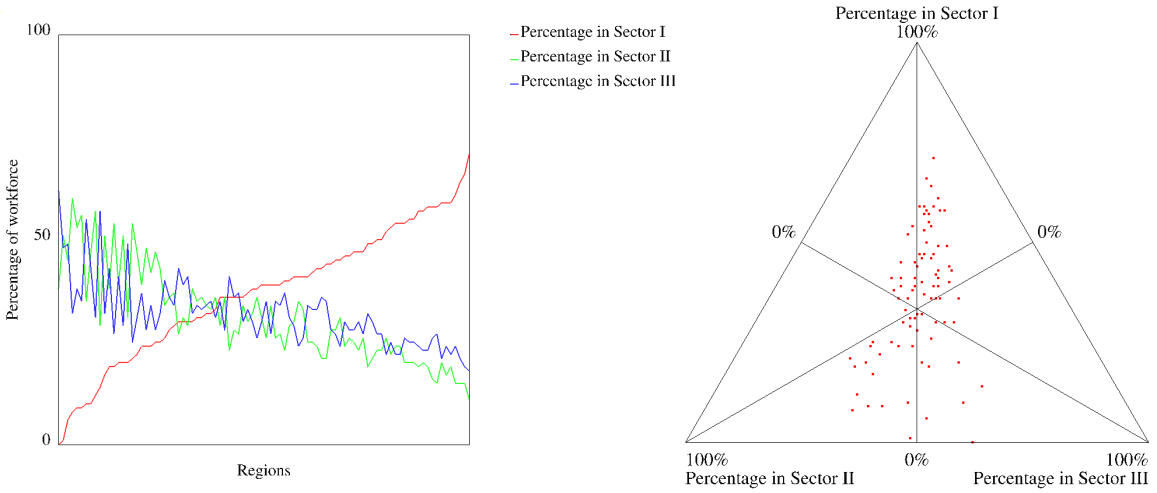


Figure 3.2: The same data represented with a line chart (left) and a trilinear plot (right).

which will ideally keep the user focused on the task at hand.

DP 13 *Allow the user to concentrate on the task*

Specifically, the computer can be used to maintain a representation of the space, in the physical world, which can also be used to permit a variety of specification techniques in a natural way.

DP 14 *Provide an external representation of possible choices*

Different people may prefer different representations (see Figure 3.2). Although this consideration is becoming integrated into most user interfaces, the implications of the concept of the visual representation as the interface to the data is not generally supported. Clearly then, support for multiple visual representations is a requirement of a user-centered tool [26].

DP 15 *Support multiple visual representations*

DP 16 *Allow the user to apply personal judgement*

All these factors are considered as the development of the new tool is laid out in the next chapter.

Chapter 4

Implementation of *cogito*

Chapter 3 developed the design principles for a computer-aided visualization (CAV) system from the perspective of a user in search of insight. Those design principles, characterized by the notions of involvement and exploration, reflect the underlying theme that a user-centered tool should provide the user with the means to make an informed choice from a potentially large space of available alternatives. Conversely, many existing software systems described in Chapter 2 generally either limit user involvement and exploration by assigning search tasks solely to the computer, or encourage involvement by providing tools to support programming and ignore the issue of exploration.

The *cogito*[1] system supports both user involvement and exploration. The system considers every visual representation to be the product of *elements* from each of several *components* and it relies on the user to choose these elements. This conception of components and elements is not unlike the modules familiar from MVE systems like AVS [252] (Application Visualization System), the toolkit philosophy of the Visualization ToolKit [222], or the components and categories used by Bertin to define his visual variables [12]. The entire space of available visual representations is structured as the Cartesian product of available components, and it is this structure which supports exploration. Consider each visual representation to be an n-tuple, where e_i is an element in component C_i as follows:

$$\langle e_1, e_2, \ldots, e_N \rangle \in C_1 \times C_2 \times \ldots \times C_N$$

[1]The name of the system, *cogito* is taken from the Latin verb "to think", which etymologically means "to shake together". It was chosen to acknowledge the role of the combination of ideas in various models of human inventive thought.

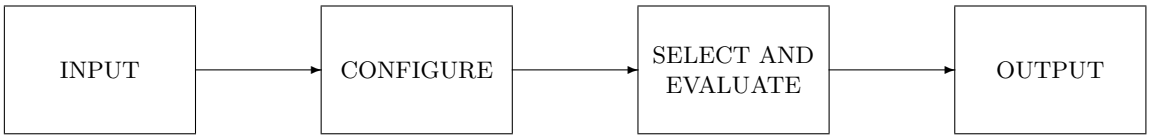


Figure 4.1: A simplified model of the *cogito* system organization. The 'SELECT AND EVALUATE' step may include details of arbitrary complexity, yet this complexity is hidden from the user.

The fundamental decision to structure the space in this way enables the concept of combinatory play from **DP 2**. Visual representations are defined as combinations of elements in *cogito* and all levels of the system architecture, reviewed next, reflect this.

4.1 Basic Organization

A standard model for the use of visualization [100] emphasizes the power of graphics as an aid to the evaluation of simulation results, but it does not emphasize the evaluation of individual visual representations. Conversely, the *cogito* system accounts for both concerns in its design, illustrated schematically in Figure 4.1.

Underlying all four conceptual phases (*input*, *configure*, *select and evaluate*, and *output*) of *cogito* is the OpenInventor [264] toolkit, which has proven very valuable both because of its portability and self-documenting features.

The discussion of the implementation details of *cogito* will follow the organization depicted in Figure 4.1. Features of the system design which extend beyond a single session are not considered here. This discussion focuses on the implementation of *cogito* (with references to the appropriate design principles – see Table 4.1) and includes an input syntax reference. Chapter 5 will present the development of two non-trivial applications for *cogito*.

The **input** phase allows specification of the problem, the data to be examined, and the tools to be used. The **configure** phase tailors the inputs to a particular situation. The **select and evaluate** phase begins in the space as configured and works always with the current space of available alternatives. This space can become exceedingly large very quickly. For example, with only 6 components each with 6 elements, there can be as many as 46,656 alternatives. The **output** phase deals with the export of the images which have encouraged insight, or provoked more questions.

Design Principle	Description	Page Where Defined
1	Support the user in working directly with images	29
2	Support the user in combining images, and elements of images	30
3	Allow the user to employ heuristic search techniques	31
4	Support a model of problem-solving	31
5	Record all aspects of the design of visual representations	31
6	Support restructuring and reorganizing of alternatives	32
7	Support collaboration	32
8	Encourage the user to discover things personally	32
9	Support the systematic exploration of a conceptual space	32
10	Support the use of current solutions for future enquiries	33
11	Support the user in choosing images	34
12	Allow the user to interact with the images	34
13	Allow the user to concentrate on the task	37
14	Provide an external representation of choices possible	37
15	Support multiple visual representations	37
16	Allow each user to apply personal judgement	37

Table 4.1: Summary of Design Principles (**DP**) developed in Chapter 3

4.2 Input

The system has been designed to allow each user the flexibility to specify all the details of his or her application from outside of the *cogito* system. The user's specification includes the components and their organization, the elements in each component and their implementations, and the compatibilities between components and elements. In general, the elements will either operate on the data to amplify it in some way, or contribute to a visual representation of the data. Particular problems may require the definition of several specific new elements, which can be easily integrated into the existing application where the applicability of these elements will be determined by the standard compatibility mechanisms. The data, and the access provided by the data model, are central to the visualization process. For this reason, considerable effort was made to ensure an expressive and modular input specification system to deal with data, metadata, and other aspects of the problem definition.

The input to *cogito* specifies all details of the application, which includes data[2], the tools to operate on that data, and context within which the data is to be interpreted. The goal is to support the flexible input of data and functions; and for this purpose the input language and parser were developed using flex and bison [147]. Input to *cogito* consists of five separate, yet related, logical pieces: *abstraction*, *libraries*, *database*, *display*, and *implementation modules*. These may be contained in a single file or split over several files.

The motivation for this separation of input is to maximize the flexibility afforded to each user of the system. Within a community which considers the same sorts of general problems, there may be a great deal to reuse. A new application in *cogito* needs only for one to create or customize as much of this input as necessary (**DP 7**). The following discusses each part of input in turn.

4.2.1 Abstraction

The *abstraction* encodes a particular conception of a problem or a class of problems, to define an interface, document it, and make it available as a basis for collaboration. This mechanism will provide metadata for other users (**DP 5**). The abstraction comprises four parts:

- compatibilities: define the basis for testing whether two elements from different components can be used together to produce a visual representation
- components: define the set from which the space of available visual representations is formed
- catalogues: define the names used to facilitate the use of the *cogito* system
- dataset format: define the permissible formats for data

Different individuals may have very different ideas about which things are important, even for something like the choice of colours in a graph-drawing application. The user will be dissatisfied if the interface either provides a control which is not wanted or omits a control which is expected. This issue came up during user testing (see Chapter 6). An individual's conception of a problem exposes the key points where he or she expects to have

[2] an application may not always have external data, as illustrated by the *Shapes* application (see Chapter 5).

influence. Having this information both expressed in and accessible through the abstraction will ensure a good match between a user and the interface which he or she has defined. As one's understanding and experience grows, his or her changing conception of the problem will warrant changes to the abstraction. This capability to evolve the articulation of the problem is a manifestation of **DP 5**.

If an abstraction exists for a problem which a new user wishes to tackle, he or she may use the existing abstraction directly, adapt it, or create an entirely new one. The work of Furnas *et al.* [86], which studied the lack of regularity and consistency with which people name things, indicates that two people might rarely choose to use the same abstraction without at least some modification.

The abstraction is developed with a programmer, who will write new code or customize existing code to implement the various elements of the components in the abstraction. The abstraction provides the necessary structure for adding functionality to the application. The precise mechanisms for doing this are detailed in the following sections.

Compatibilities

```
compatibilities compatibility-list

compatibility-list
        : compatibility-list-elem
        | compatibility-list compatibility-list-elem

compatibility-list-elem
        : compatibility-name ( condition-list )

condition-list
        : condition-list-elem
        | condition-list condition-list-elem

condition-list-elem
        : condition-name descriptive-text
```

Table 4.2: Syntax for the specification of compatibilities.

In most applications, not every possible combination of elements will be meaningful or unique. In order to make stipulations about how elements from different components can

be combined, the facility of *compatibilities* and *conditions* is provided (see Table 4.2 for the syntax). For example, to distinguish between two-dimensional and three-dimensional data, one might use the syntax illustrated in Table 4.3.

```
compatibilities
        dimension ( "2D" "two-dimensional data"
                    "3D" "three-dimensional data")
```

Table 4.3: A compatibility called dimension, with two conditions

This capability allows a person to enforce certain rules in *cogito*, but this is a very limited facility. In accordance with **DP 9**, the design of *cogito* is flexible about the application of constraints.

Components

This section of the abstraction indicates the dimensions of the space of alternatives, with each component representing a dimension (N components create an N-dimensional space). See Table 4.4 for the syntax. Components are required to match the conditions of one or more compatibilities. See Table 4.5 for an example.

Another view of components is that each implements some aspect of the traditional 'filter, map, render' visualization process. From this perspective, the importance of the ordering of the components is clear. There is more than one ordering possible, however, since only a partial order is imposed by the processing dictated by OpenInventor [264].

The in-order list of all components describes a "template", implemented with an OpenInventor scene graph. Components either represent nodes or attributes, meaning that they will modify nodes in the scene graph. The scene graph is an important data structure in *cogito*: it can be read or modified by all components. The `getstate` and `setstate` commands are intended as a documentation feature in the abstraction. The state information is actually carried in the graph and accessed by a user-extensible library (see Subsection 4.2.5). The collection of this state information in a central place reduces the complexity required in each module, as indicated by **DP 13**.

components component-list

component-list
: component-list-elem
| component-list component-list-elem

component-list-elem
: component-name descriptive-text
match compatibility-name-list
[*attribute*]
[*getstate* state-info-list]
[*setstate* state-info-list]

state-info-list
: state-info-list-elem
| state-info-list state-info-list-elem

state-info-list-elem
: state-info-name

Table 4.4: Syntax for the specification of components.

```
components
        colourSelect "Methods to select colours from a palette"
        match base
        attribute
        setstate colourSelect
```

Table 4.5: An example of a component specification.

catalogues *catalogue-list*

catalogue-list
: *catalogue-list-elem*
| *catalogue-list catalogue-list-elem*

catalogue-list-elem
: *catalogue-name descriptive-text*
⟨*data-type* | *reference state-info-name* ⟩(*value-list*)

value-list
: *value-list-elem*
| *value-list value-list-elem*

value-list-elem
: *value-name descriptive-text value*

Table 4.6: Syntax for the specification of catalogues.

```
catalogues
        aspects "Aspect ratios for drawing graphs" float
        clrsel "Methods for selecting colours" reference colourSelect
```

Table 4.7: Two examples of catalogues. The first is descriptive because it will allow different values to be named, and the second is referential because it will allow code segments to be accessed.

Catalogues

Each catalogue is an enumeration of some quantity. A catalogue can be either descriptive or referential, see Table 4.6 for the complete syntax. The purpose of descriptive catalogues is to provide a layer between machine-implementation details and the user so that the parameters to elements have intuitive names which can be queried and understood easily. The purpose of referential catalogues is to provide a mechanism which allows references to *sampling* modules (see Subsection 4.2.5) added from different libraries to be registered with the system and used by the application. See Table 4.7 for an example.

Catalogues may be named with any valid string, except for two: the system presently reserves the names ‘relation’ and ‘amplifier’ for special purposes. A user is still free to add

designations to these catalogues.

Data Set Formats

This section describes the formats for data accepted by the system on behalf of an application; two sample applications (*Shapes* and *Graphs*) are described in Chapter 5. An application like *Shapes* might not use any external data, whereas an application like *Graphs* could use data of practically all formats. The format names facilitate the input of the actual data. The syntax is given in Table 4.8.

Conceptually, a data set is divided into one or more logical segments, which are related to one another by some key value. The data sets for the *Graphs* application all have a single logical segment. More complex data will arise from, say, measurements taken on a grid of positions over time. Here, the measurements at each time step constitute a segment and the segments for all time steps measured form the whole data set. An index may be necessary to make the proper connections and interpret the grid in visual form. The system now allows for a single index file to be associated with all segments. This example would be expressed to *cogito* with the sample code in Table 4.9.

Relationships which exist between fields within a data set can be indicated by listing those fields as arguments of various *relations*. The relation names are defined and described in the 'relation' catalogue. For example, the *Graphs* application makes available the trilinear plot, discussed in Chapter 2. It can be used only to plot three data fields whose sum is always constant. As part of the data set format, one can indicate whether this condition exists in the data. See Table 4.10 for an example.

4.2.2 Libraries

This section describes the libraries, each of which contains the descriptions of the component elements and catalogue designations made available by the indicated dynamic shared object (DSO) [124]. See Tables 4.11 and 4.12 for the syntax. The library establishes a mapping between the code modules in the DSO and the structures of elements and designations used by *cogito*. Matching of specifications to modules is done by extracting all local functions from the symbol table of the DSO and testing for a symbol name which completely contains the code name from the specification. Because the structure of *cogito* is combinatorial, one new element for a component will result in many new combinations, based on specified

datasets *dataset-format-list*

dataset-format-list
: *dataset-format-list-elem*
| *dataset-format-list dataset-format-list-elem*

dataset-format-list-elem
: *dataset-format-name descriptive-text*
⟨ ***sequential*** | ***indexed*** ⟩
[*heading-description*] *body-description relations-description*

heading-description
: ***header*** (*data-description-list*)

body-description
: ***body*** (*data-description-list*)

data-description-list
: *data-description-list-elem*
| *data-description-list data-description-list-elem*

data-description-list-elem
: *data-field-name data-type* ***units*** *units-name*

relations-description
: ***relations*** (*relations-description-list*)

relations-description-list
: *relations-description-list-elem*
| *relations-description-list relations-description-list-elem*

relations-description-list-elem
: *relation-name* (*data-field-list*)

data-field-list
: *data-field-list-elem*
| *data-field-list data-field-list-elem*

data-field-list-elem
: *data-field-name*

Table 4.8: Syntax for data set formats.

```
datasets
        mesh indexed
        header (
                "Time" float units "Time"
        )
        body (
                "X coord" float units "position"
                "Y coord" float units "position"
                "Z coord" float units "position"
        )
```

Table 4.9: Designating a format for a data set comprised of segments, each identified by the value of "Time" in the headers.

```
datasets
defset sectors sequential
body (
        "Percentage in Sector I" float units "Percentage of workforce"
        "Percentage in Sector II" float units "Percentage of workforce"
        "Percentage in Sector III" float units "Percentage of workforce"
)
relations (
        percent100(
                "Percentage in Sector I"
                "Percentage in Sector II"
                "Percentage in Sector III"
        )
)
```

Table 4.10: In an example from the *Graphs* application (see Chapter 5), the `percent100` relation is used to indicate that the percentage in each of the three sectors sums to 100. By testing the number of data fields in the relation, the programmer can verify that the trilinear plot may be used with this data.

```
libraries library-list

library-list
        : library-list-elem
        | library-list library-list-elem

library-list-elem
        : library file-name descriptive-name
          [elements element-list ]
          [designations designation-list ]
```

Table 4.11: Syntax for library specification.

compatibilities. An example is given in Table 4.13.

All the functionality of any *cogito* application is encapsulated in the DSO libraries which are loaded as that application is run. Once standard libraries are available, it is easily foreseeable that the only work to get started on a new application will be the preparation of the abstraction files.

The evolution of problem conceptions is also supported by the interface to these code libraries because one can edit the descriptions of elements without having to recompile anything. As new functionality is added or specialized, it can be placed in its own separate DSO with its own library file (**DP 5**).

Each element is identified with a particular component and its relationship to other elements in other components is specified by a condition from each of the compatibilities associated with the component. The *cogito* system uses this information to determine which combinations of elements are valid. This is done by comparing the compatibility conditions of the current element with those already in the partially-assembled combination. Only those compatibilities for the component are tested. The element may have an optional validator module specified with it. If the element is compatible with the combination, the validator module is invoked before the element module, to test any conditions on the arguments.

Any arguments for the module are specified next. Each argument to the element module may be either a single value, an enumerated list of values, or a continuous range which is sampled. If the parameter is continuous, some number of samples must be specified. The net effect is to create another, smaller space of alternatives based on the possible values of each argument. Consider each specified invocation of an element module to an m-tuple,

element-list
: *element-list-elem*
| *element-list element-list-elem*

element-list-elem
: *descriptive-name element-module-name* [*validator-module-name*]
component *component-name*
[**arguments** *argument-list*]
matches *compatibility-condition-list*
description *descriptive-text*

argument-list
: *argument-list-elem*
| *argument-list argument-list-elem*

argument-list-elem
: *argument-name catalogue-name value-specification*

value-specification
: [**range** *Val Val*] [**samples** *number*] [**value** *typed-value*]

compatibility-condition-list
: *compatibility-condition-list-elem*
| *compatibility-condition-list compatibility-condition-list-elem*

compatibility-condition-list-elem
: **group** *compatibility-group-name condition-name*

designation-list
: *designation-list-elem*
| *designation-list designation-list-elem*

designation-list-elem
: *catalogue-name designation-name descriptive-name*

Table 4.12: Syntax for library specifications, continued.

```
"Graphs in 2D" graphs.so

elements
        "Parallel projection onto "XY plane" CAM_orthoXY
                component camera
                matches dimension "2D"
                description "Orthographic camera looking at XY plane"

designations
        vary_by_data clrsel "Vary colour selection with data"
                "linearRamp_RGB 0"
```

Table 4.13: Specifying a very small library, which refers to two functions, `CAM_orthoXY` and `linearRamp_RGB`.

where v_i is a value for argument A_i, as follows:

$$\langle v_1, v_2, \ldots, v_M \rangle \in A_1 \times A_2 \times \ldots \times A_M$$

The declaration of the element, which establishes the link to its implementation, is responsible for correctly indicating the interface to the module. Different elements of the same component do not need to have the same interface.

Control over the appearance of the space of alternatives is exercised by manipulating the mapping between elements and modules. One can define a single element which samples the space of its arguments, according to a specification, or one can define several elements with each one associated with a particular point in that parameter space. A rule of thumb is that qualitative differences can be associated with elements and quantitative differences associated with parameter changes. One can choose to identify discrete values of a continuous parameter by creating designations in a catalogue.

Each element has a description, the text of which is processed to reflect the arguments at invocation. This information is stored in the scene graph, and available for query through the interface.

Designations specified in the library are generally of the reference type, which makes sampling modules (see Subsection 4.2.5) available to the system. The reference is actually a quoted string which specifies the name of the module and any other required details.

Parsing of this reference string is done by the state information class associated with the catalogue, which defines the requirements of the interface.

4.2.3 Database

The data available to a particular invocation of an application is indicated in this part of the input. By associating the data with a format specified in the abstraction, one is relieved of the need to describe the metadata for each file. The existence of a format also makes it easy to develop a standard file format or support several different ones. The new dataset becomes a named instance of the specified format (see Tables 4.14 and 4.15). All data is converted to OpenInventor types upon input. This gives access to all of OpenInventor's data capabilities, including the use of disk-based data management if required.

The data fields are treated as individual variables, to which operations may be applied. This view is like that of *Data Desk* [254], however, *cogito* adds the concept of relations amongst the fields.

4.2.4 Display

Input also includes indications of viewing and display parameters. See Table 4.16 for a guide to this syntax. However the user chooses to think of and view the components (in which order), they are processed according to the order given in the abstraction.

4.2.5 Implementation Modules

The *cogito* system employs three different types of modules, distinguished primarily by their purpose. The most prevalent type of module is that which implements the behaviour specified by an element. Each **element module** has a standard interface which allows it to accept a variable number of arguments. This interface is implemented using the protocol of name/value pairs provided by the 'Arg' functions of X Window Toolkit [182]. Each of these modules accepts three arguments: an Arg structure, the number of arguments, and a reference to the scene graph currently under construction. This last argument enables communication between components, so the complexity needed by any one module is reduced. Each component can read and write state information to and from the scene graph, which can be accessed by later components. Because the order of components is fixed in

```
database dataset-list

dataset-list
        : dataset-list-elem
        | dataset-list dataset-list-elem

dataset-list-elem
        : dataset-name format-name
          [index ( index-record-list )]
          ( data-segment-list )

index-record-list
        : index-record-list-elem
        | index-record-list index-record-list-elem

index-record-list-elem : index-record

data-segment-list
        : data-segment-list-elem
        | data-segment-list data-segment-list-elem

data-segment-list-elem
        : segment ( segment-record-list )

segment-record-list
        : segment-record-list-elem
        | segment-record-list segment-record-list-elem

segment-record-list-elem
        : data-set-record
```

Table 4.14: Syntax for database specification.

```
database
set sampleGrid mesh
index (
#include "time-data/mesh.idx"
)(
segment (
#include "time-data/solu0.000"
)
segment (
#include "time-data/solu0.001"
))
```

Table 4.15: Input of data corresponding to the dataset format given in Table 4.9. Here the dataset is called "sampleGrid" and is an instance of the format called "mesh."

the abstraction, programmers can use it to communicate whatever state information they need when writing their modules.

The programmer can create new state information, as required, by creating subclasses of the general state-info class CogStateInfo (Table 4.17) or the specialized class CogStateFuncInfo which allows a *sampling module* to be invoked (Table 4.18).

Components can either represent nodes or attributes in the scene graph. The modules which implement nodes return a reference to an OpenInventor group structure [264]. There are two children in such a group: the first contains the descriptive information from the associated element and the second contains the implementation details. Those modules which implement attributes return a boolean value to indicate whether or not the modification was successful.

If the execution of either variety of element module is not successful for whatever reason, construction of the current combination is stopped. This mechanism is the last check of the compatibility of an element.

Each element may optionally specify a **validator module** to verify that a particular collection of parameter values will constitute a valid invocation of the element module. These modules return a boolean and are invoked with only two arguments (the Arg structure and the argument count) since this module is called before a scene graph is constructed.

Each designation in a reference catalogue has an associated **sampling module**. These

views *views-list*

views-list
: *views-list-elem*
| *views-list views-list-elem*

views-list-elem
: *view-key-list*

view-key-list
: *view-key-list-elem*
| *view-key-list* **and** *view-key-list-elem*

view-key-list-elem
: *component-name*

display *display-qualifier-list*

display-qualifier-list
: *display-qualifier-list-elem*
| *display-qualifier-list display-qualifier-list-elem*

display-qualifier-list-elem
: *samples number*
⟨ *blocked* | *interleaved* ⟩
policy policy-specifier
gridsize x-number y-number aspect-ratio aspect-ratio

Table 4.16: Syntax for the specification of display qualifiers.

```
#ifndef __CogStateInfo__
#define __CogStateInfo__

class CogStateInfo {
protected:
        CogString token;
        CogString csi;

public:
        CogStateInfo(void);

        void display(void);
        CogString& getInfo(void);
        CogBool store(SoSeparator*);
        CogBool retrieve(SoSeparator*);
};
#endif
```

Table 4.17: Header file for the CogStateInfo class

```
#include "CogStateInfo.h"

#ifndef __CogStateFuncInfo__
#define __CogStateFuncInfo__

class CogStateFuncInfo : public CogStateInfo {
protected:
        CogString func_name;
        void* funcptr;
public:
        CogStateFuncInfo(void);

        CogBool storeFunc(SoSeparator*,CogString&);
};
#endif
```

Table 4.18: Header file for the CogStateFuncInfo class

modules are related to some particular piece of state information. In that file, the details of the function interface are defined.

4.3 Configuration

```
configuration configuration-spec-list

configuration-spec-list
      : configuration-spec-elem
      | configuration-spec-list configuration-spec-elem

configuration-spec-elem
      : ⟨ include | exclude ⟩
        ⟨ condition | designation | element ⟩
        name
```

Table 4.19: Syntax for the specification of configuration details.

The configuration stage of processing is used to limit the size of the root space from that which is determined only by the modules described in the library files. The user can tailor the space for a particular invocation of an application, without having to create separate versions of the library files. An example is illustrated in Table 4.19.

The first selection is made in the configuration part and this determines which elements, conditions, and designations will be used in a particular invocation of an application.

4.4 Selection and Evaluation

The architecture of the *cogito* system thus far described shares much in common with the design of more common modular visualization environments. The key difference between *cogito* and other systems is its focus on the space of alternative visual representations (**DP 14**) and its support for the user in structuring and managing that space (**DP 9**). For any application, the composition of a space may very well change over time, as elements and even components are added or modified (**DP 5**). This flexibility is essential in supporting the entire problem-solving process, as indicated by **DP 4**.

The selection of alternatives is facilitated through a visual interface which is reminiscent of Sims' system for artificial evolution in computer graphics [231]. A key factor in choosing

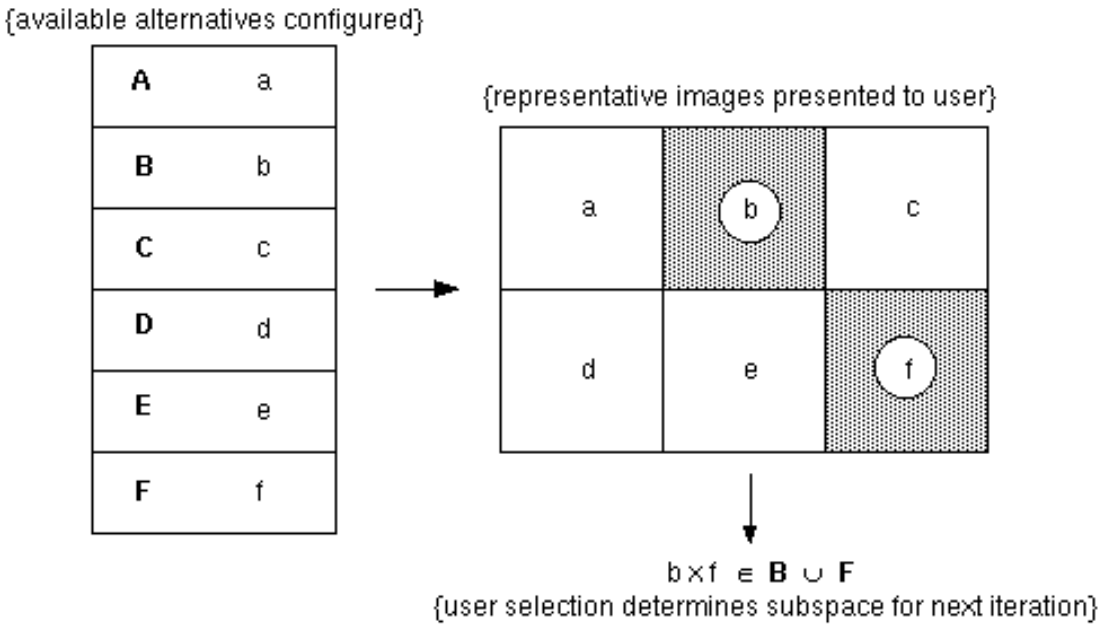


Figure 4.2: Schematic look at the interface: the space of available alternatives is grouped according to user-specified criteria. Each group (A – F) has a representative element (a – f) which is displayed to the user. The subspace for the next search iteration is based on the user selection (b and f).

this design is its ability to fulfill **DP 1** by allowing users to select visual representations directly from the screen, through the use of embedded menus [224]. Users of any application in *cogito* always see the same interface, which is provided by the system, and can benefit from that consistency by focusing more on their task (**DP 13**).

The interface (see Figures 4.2 and 4.3) displays a subset of available representations (sampled according to the selected organization of the search space), generated from the current data, with which the user can interact. The creation and traversal of this hierarchical set of spaces is the result of decisions at various points. From a design perspective, all these choices provide valuable information about the design and should be recorded [76]. An annotation facility has been implemented within *cogito* to meet this need (**DP 5**). To help the user to productively use the notion of the space of alternatives, the interface actively supports this model (**DP 14**). The concept of a space is made more manageable by several additional structures:

- each space is examined through a *view*, which establishes how the alternatives are presented to the user. Elements in the view's key component(s) are sampled sequentially from the current space. Elements from the other components are sampled from the current space in a pseudo-random fashion. For example if the view key is the 'Colour' component and the current space has 'Red', 'Green', 'Blue', and 'Yellow' elements from that component, one will see Red, Green, Blue, and Yellow objects always in sequence, with the other elements of the combination completed in a pseudo-random way. A user may choose to establish several different views for the same space (**DP 6**). The view partitions the space and alternatives from each partition are seen in sequence. Elements from the other components are chosen in a pseudo-random fashion.

- each view comprises one or more *screens*, which are generated only at the request of the user.

- each screen comprises two or more[3] *cells*, each of which displays a visual representation, determined by the view and other parameters. The visual representation in a cell can be examined or its contents queried by the user.

The collection of these structures provides a context in which to interpret the visual representations. The system manages the combination of elements and organizes any records of experiments. Multiple cells on the screen at once support the user in evaluating alternatives. The environment is well-suited to experiments, where individuals can invent their own ideas or recreate the work of someone else; they become the only inventor they know [101].

The elements in the *cogito* system provide the functionality of the filter, map, and render pipeline of Haber and McNabb. The difference with the *cogito* system is that once the elements have been specified to the system, the user does not have to explicitly assemble that pipeline (**DP 13**).

The design is focused on supporting the user in selection and evaluation, which requires a meaningful access to the space of alternatives. A user might want to see all alternatives in a space by looking at successive screens although this may easily result in hundreds of screens to look at. To allow this definiteness, no alternative is allowed to be displayed more

[3]Ten is a practical upper limit on this number, so that the screen-size of individual alternatives is not too small.

than once in a particular view. A more powerful approach supported by the interface is to review only a small fraction of the alternatives. This allows the user to explore a more confined region of space. The user may see the desired representation, or parts of it, and use it as the basis for further selection and evaluation (**DP 10**). An alternative displayed in a cell is selected by clicking anywhere in the cell. The colour of the cell border changes to reflect its selected state. Once the user has evaluated sufficient alternatives, selections made in the current space are used to define a new space, using the equivalent of a genetic crossover operation. The new space is consistent with the selections made in the previous space. Figure 4.3 displays an annotated version of the *cogito* system interface. Figure 4.4 illustrates the system use with an application.

Although it follows an interactive evolution paradigm in a sense, the possible solutions are not evolving, as would be the case in a genetic programming system [193, 231]. The first such artificial evolution system accompanied Dawkins' *Blind Watchmaker* [55]. More recently, Sims [231] used this idea to generate computer-graphical imagery. Kochhar [136] applied the idea to a generative system. The *cogito* system follows in this tradition with the user providing the fitness function through his or her selections. One navigates and searches the space by a genetic algorithm (**DP 3**) which creates new spaces consistent with selections. It is a technique which can be applied to especially good advantage when the satisficing solutions to a problem are not known. Proceeding in a directed random way, promising alternatives can be quickly identified. In less ideal circumstances, a good strategy for exploration of the whole search space is to pursue a particular direction with each search and perform many searches. Left without user guidance, the system will present images taken from the entire search space. In this case, however, a higher proportion of the alternatives presented may be deemed unusable. And since only a relatively small number of alternatives will be displayed, it is more likely that useful ones will be missed. Alternatively, since one sees a good sampling in one dimension, it may be possible to find a partial solution which can be further developed. The possibility that the selected alternatives will come from combinations generated outside the user's experience is a very powerful aspect of this approach, which was also noted by Kochhar *et al.* [136]. Once the promising alternatives are identified as such, the nature of the navigation may change.

The user may either be sure of the modification needed to create an effective representation, or he or she may choose to explore the similar combinations in an effort to better understand the neighbourhood of the combinations and ultimately allow some insight.

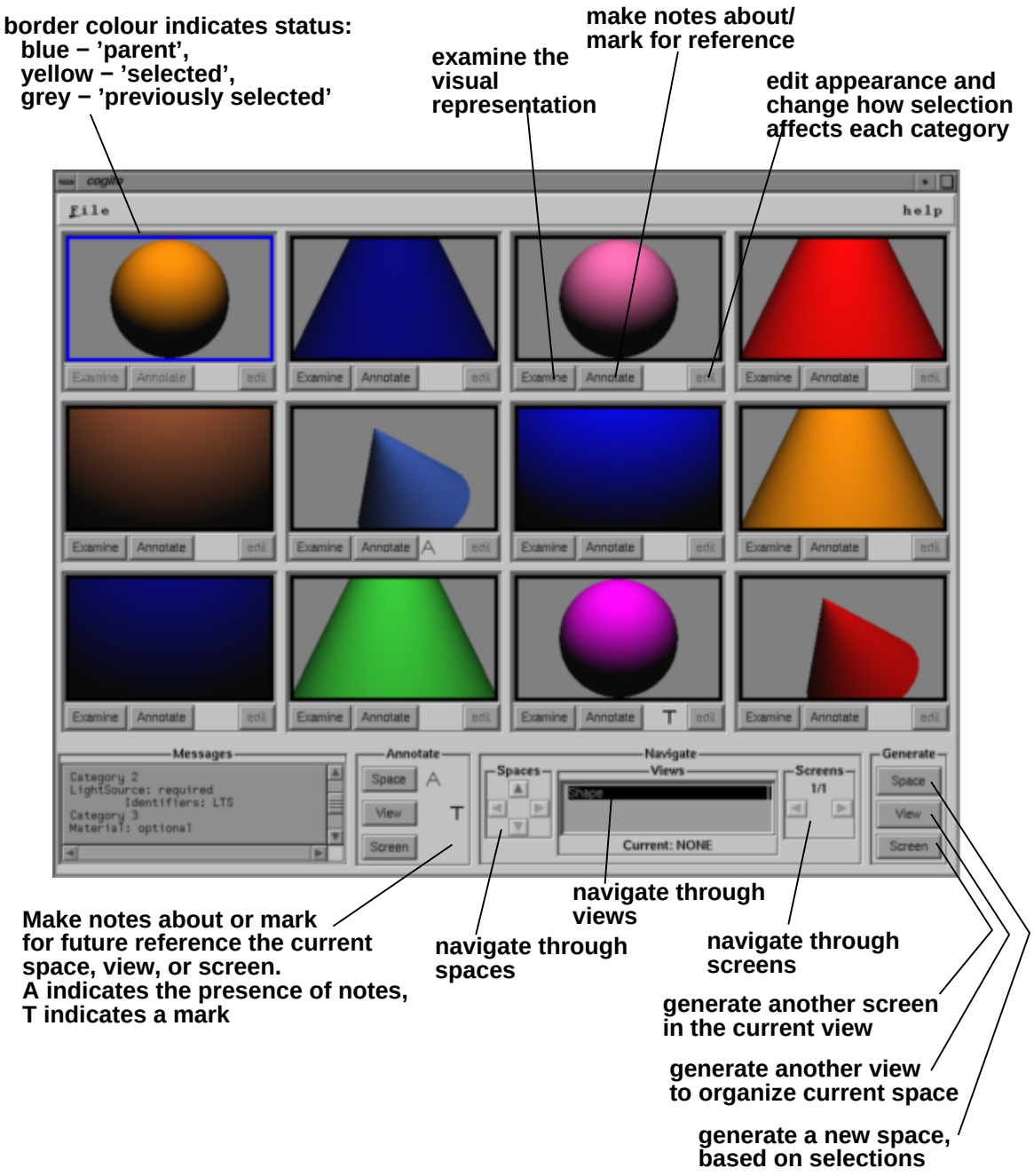


Figure 4.3: The interface to the *cogito* system, with annotations. The user will always see the same interface, regardless of the particular application being run (here shown with the *Shapes* application, described in Chapter 5).

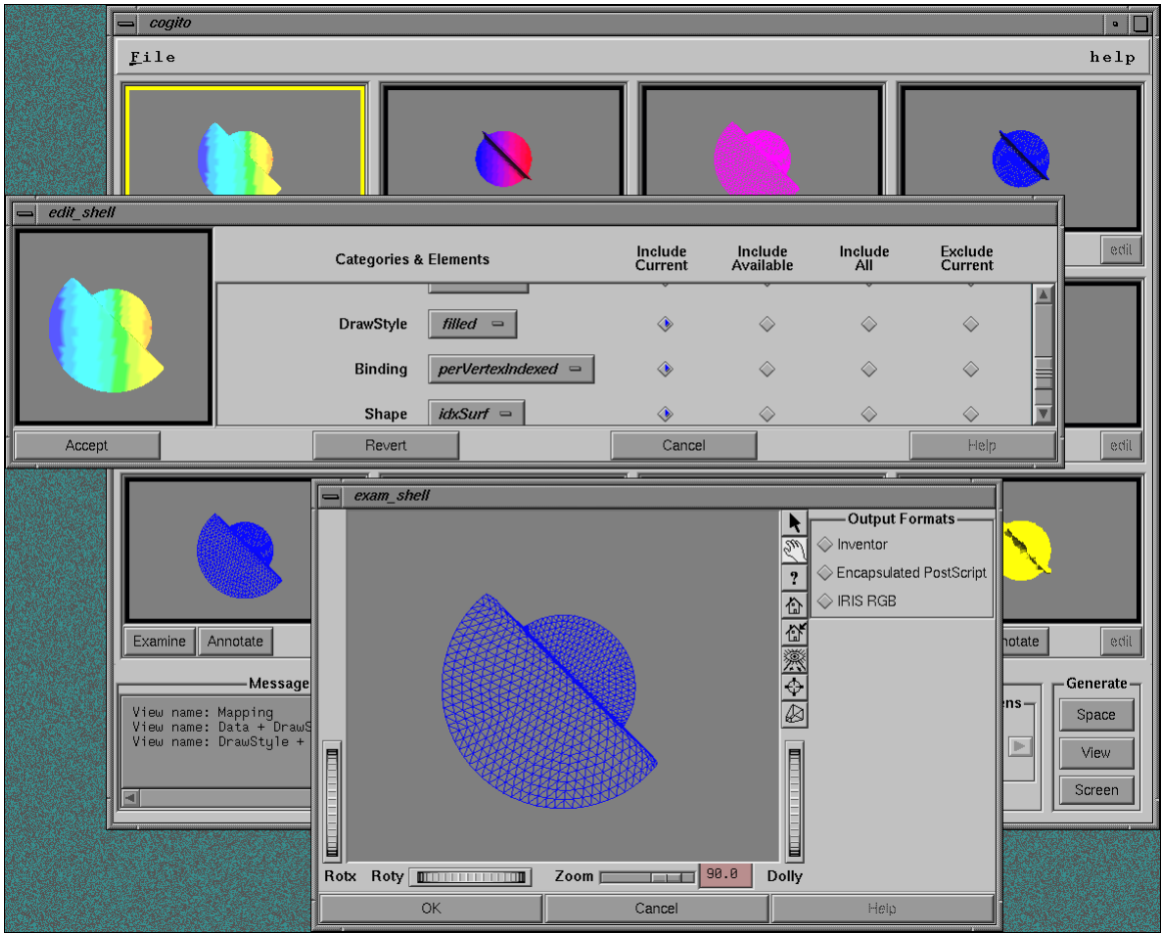


Figure 4.4: An illustration of the *cogito* system with an application. At the forefront is the dialogue box which permits editing of the current visual representation.

The *cogito* system provides mechanisms to deal with both situations. The former requires an "edit" capability that allows individual elements to be changed to different particular elements. The latter requires a means to return elements to the search, by implementing a "match any" capability that can be expressed for particular components.

The user identifies features of interest in a relatively non-specific way. These identified features may lead to the perception of the space as homing, in which case a directed refinement approach may be taken. If the space remains as Klondike, the user may be advised to continue generating combinations randomly. The topographies proposed by Perkins [191] correspond well to the sorts of search strategies which can be effectively applied [91]. If a space is homing, then a gradient-based local search method like hill-climbing can be applied.

If the space is Klondike, a genetic algorithm provides a robust method. The same space can appear in different ways to different people and this is based on each person's personal experience and knowledge of the task. The *cogito* system uses this notion to allow a space to be examined through different organizations (**DP 6**).

The approach to selection and evaluation, though different from MVE's, could be used to develop networks for use in those systems. The focus of this work differs from those, as well as Chi's Spreadsheet for Information Visualization (SIV) [42] and Levoy's Spreadsheet for Images [148], because it concentrates on the implicit rather than explicit expression of operators.

4.5 Output

Generativity is an important feature of a problem solution [191]. Annotated logs of space explorations could serve as important documentation in the process of design, and a foundation for insight. Output to OpenInventor [264] or VRML [106] could serve to encourage much wider collaborations, though sharing of libraries will also be an important activity. Appendix C presents a sample OpenInventor output file.

The results of the visualization process will be the selected visual representation(s) and a map of the space of alternatives that was traversed in the process of finding the visualization. The intent is to provide the researcher with different physical and conceptual views of the problem data. This is an important aspect which is advocated by Brown and Sedgewick [26].

4.6 Implications

The goal of providing access to a personal paradigm for visualization has created a system which affords choice at every alternative, and one that does not do away with the role of programmer but rather shifts the focus to the relationship between researcher and computer.

The concept of a hierarchical organization of spaces contrasts with the flat view which is otherwise used. By ordering the components, one can think of the combinations as paths in a tree from root to leaf. A change or variation early in the tree will potentially affect quite a number of leaves, whereas a change in a leaf node is a very local change. This idea is applicable when reorganization of the space occurs.

As noted by Treinish [247], there is still a need for programming help even for the the

visualization-literate scientist. An intuitive means of accessing alternatives puts the scientist in direct control, further alleviating the need for specialized programming assistance. The new paradigm proposed herein allows that direct access by the research scientist. The reverse is true that such a specification system could be used as a programming tool for a dataflow visualization system such as AVS [252]. Programming support will be required at those times when the need arises from problem-finding activities in *cogito* for new visual elements to be added to the library. This aspect of support for the addition of elements will be discussed in Chapter 5 with respect to particular applications.

Although it would be interesting to explore whether it is beneficial to keep the dimensions of the space also hidden from the user, here they are more visible and are used as a means for the user to interact with the system. Since there is an expectation that users will have sufficient expertise with the application, this decision will give the users more control. This choice may in some ways be at odds with the idea of keeping the user away from language-related details of the problem, though alternatively, the user must have some grasp of the problem, and this can be expressed in terms of this interface. If the user did not have a role in defining the abstraction, then he or she must gain familiarity with the components and elements before proceeding. One might soon see how well the abstraction has been constructed by observing the user interact with it.

It might also be interesting to explore whether there are popular choices for visual representations which relate to specific tasks. As such, multidimensional scaling [141] analysis might prove useful in exposing any patterns which might in turn be used to provide better support to the user navigating the space, if he or she chooses to use that support.

The rationale for building the *cogito* comes from the goal to empower both user and programmer in a visualization environment. The next chapter describes, with concrete examples, the programmer's interface to *cogito*.

Chapter 5

Building applications for *cogito*

To illustrate the power of the *cogito* system, this chapter deals with the construction of two *cogito* applications, *Shapes* and *Graphs*. These same applications were used to conduct the user study, described at length in Chapter 6.

The user of a *cogito* application deals with the *cogito* interface, tailored by the current application. Therefore, the user will not need different computer skills for each application in *cogito*. Rather, once the user is acquainted with the *cogito* interface, only the task-specific knowledge relevant to the application will be required.

The prototype *cogito* system permits applications to be built in a distributed fashion with the description and implementation expressed in several logical parts. It is easy to see that new applications may be built by mixing new and existing pieces with a minimum of effort. In this description and use of the prototype *cogito* system, effort has been focused on the manual creation of the pieces which constitute the applications.

This distributed model of an application in *cogito* reshapes the traditional relationship between user and programmer when solving visualization problems; it places the user in direct contact with the system which is generating images and that user is in control of the process.

A person may interact with the *cogito* system either as a user or as an application programmer. Generally, these two roles may be assumed by the same person, two different (groups of) people who interact while developing an application, or two different (groups of) people who do not communicate at all. The last situation may arise when an application has been sufficiently developed that its use spreads beyond those who defined it.

The remainder of the chapter will present each step of the application-building, with

illustrative examples from both the *Shapes* and *Graphs* applications.

5.1 Defining an Abstraction

As discussed in Subsection 4.2.1, the abstraction defines the user's interface to the application. The programmer and user must first decide which aspects of the problem will be put under the control of the user, then secondly, how the user will access those controls. There are many ways to describe any problem, and it is worthwhile to consider them carefully. Three pairs of distinctions are of interest:

- **components** identify the logically distinct and indivisible units in a problem. Some of these may be almost universal while others are very specialized
- **elements** identify the *qualitatively* different choices within each component. Each element can be parameterized to enables a range of *quantitatively* different choices to be explored for each element

- **compatibilities** identify the areas where coordination is needed
- **conditions** identify the particular cases within each compatibility

- **catalogues** identify the themes for the collections of names that are made accessible throughout the different files of an application
- **designations** identify the particular values and references

The whole process of defining an abstraction may go through several refinements before the user is satisfied.

5.1.1 Shapes

The *Shapes* application (see Figure 5.1) gives the user some means to experiment with the appearance of simple three-dimensional shapes with various colours, transformations, and lighting. In keeping with this intent, `camera`, `lightModel`, `lightSource`, `lightPosition`,

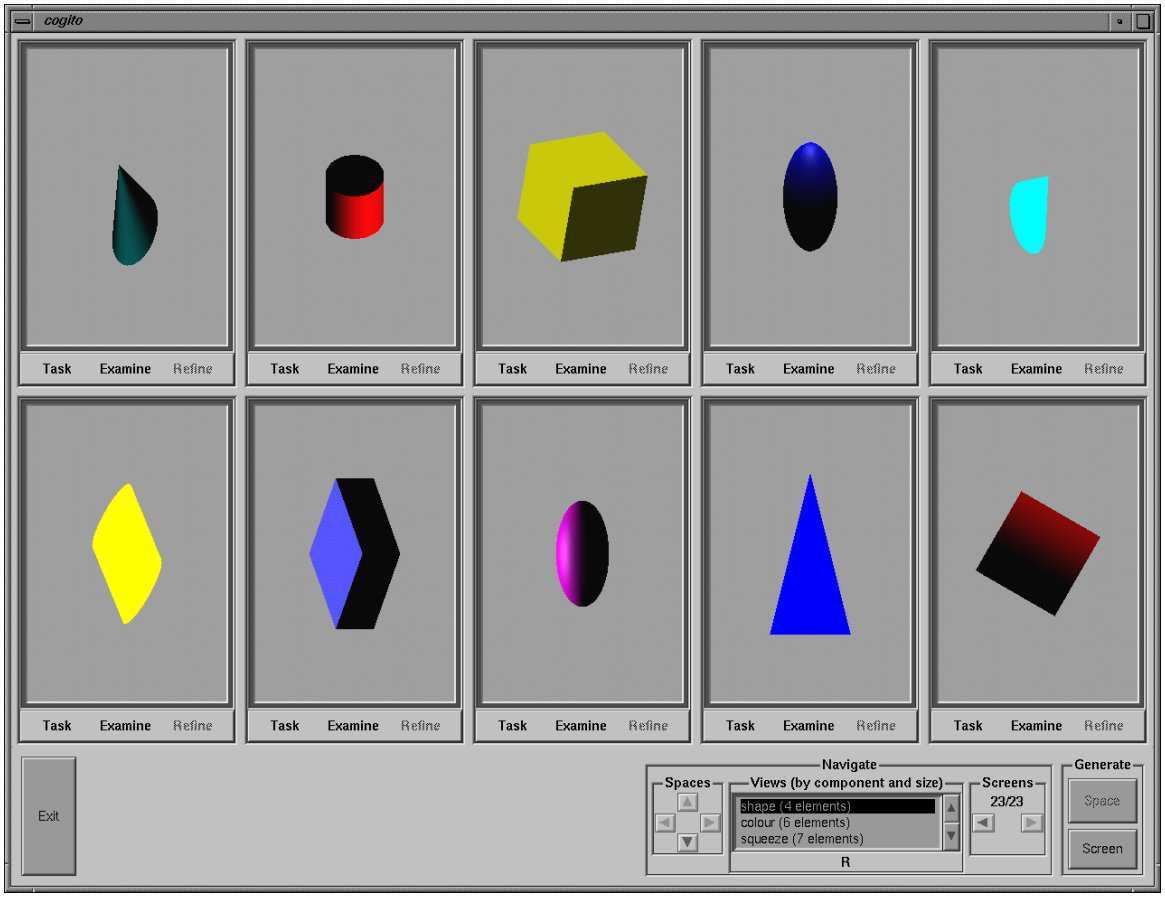


Figure 5.1: The *cogito* system running the *Shapes* application.

```
abstraction
compatibilities
        base "Base compatibility" ( base "Base condition" )
        dimension "Dimensionality of shapes"
                ( "3D" "Three-dimensional shapes" )
        lightingProperties "Use of lighting information"
                ( base "Base colour without any light",
                phong "Phong illumination with light" )
        lightingPosition "Position of lights"
                ( none "No light position",
                spec "Light position specified by user" )
        lightingDirection "Direction of lights"
                ( none "No light direction",
                spec "Light direction specified by user" )

components
        camera "Viewing information" match dimension
        lightModel "Model for the way light is used"
                match lightingProperties
        lightSource "Type of light source"
                match lightingProperties lightingPosition lightingDirection
                setstate lightSource
        lightPosition "Position of point light source"
                match lightingProperties lightingPosition
                attribute getstate lightSource
        lightDirection "Direction of directional light source"
                match lightingProperties lightingDirection
                attribute getstate lightSource
        colour "Colour of shapes" match base
        squeeze "Shrinking of shape along some axes" match base
        orientation "Rotation of shape around some axes" match base
        shape "Three-dimensional shape" match base
```

Table 5.1: Abstraction for *Shapes* application.

```
catalogues
        lm_cat "Available light models" descriptive integer
                ( base "Base colour light model" 0,
                phong "Phong light model" 1 )
        ls_cat "Available light sources" descriptive integer
                ( none "No light source" 0,
                point "Point light source" 1,
                directional "Directional light source" 2 )
```

Table 5.2: Abstraction for *Shapes* application, continued.

`lightDirection`, `colour`, `squeeze`, `orientation`, and `shape` components are defined. See Tables 5.1 and 5.2 for the actual specification.

The abstraction does not give a great deal of control over the transformations, since neither the squeeze nor orientation component reveals very much about how the transformation will be applied. As implemented, the order of these components will affect the displayed image. The `lightModel`, `lightSource`, `lightPosition`, and `lightDirection` components work together to determine the lighting of the shape. Because of this interdependence, compatibilities are defined to control the combination of elements between these components. For example, both `lightSource` and `lightPosition` are coordinated by the `lightingPosition` compatibility. This means that the elements from the `lightSource` and `lightPosition` components must both have the same condition from the `lightingPosition` compatibility in order to form a valid combination. Additionally, state information is set by the `lightSource` component and used by the `lightPosition` and `lightDirection` components to determine the type of light source in the scene. Two catalogues are defined to provide the user with more mnemonic names. The abstraction for this application is made simpler because it does not use any external data.

5.1.2 Graphs

The *Graphs* application (see Figure 5.2) provides the user with access to a variety of plotting methods in two space dimensions. The following components are defined: `camera`, `lighting`, `layout`, `size`, `texture`, `colourPalette`, `colourSelect`, `plotData`, `mapData`, `dataAmplification`, `representation`, and `annotation`. See Tables 5.3 through Tables 5.6

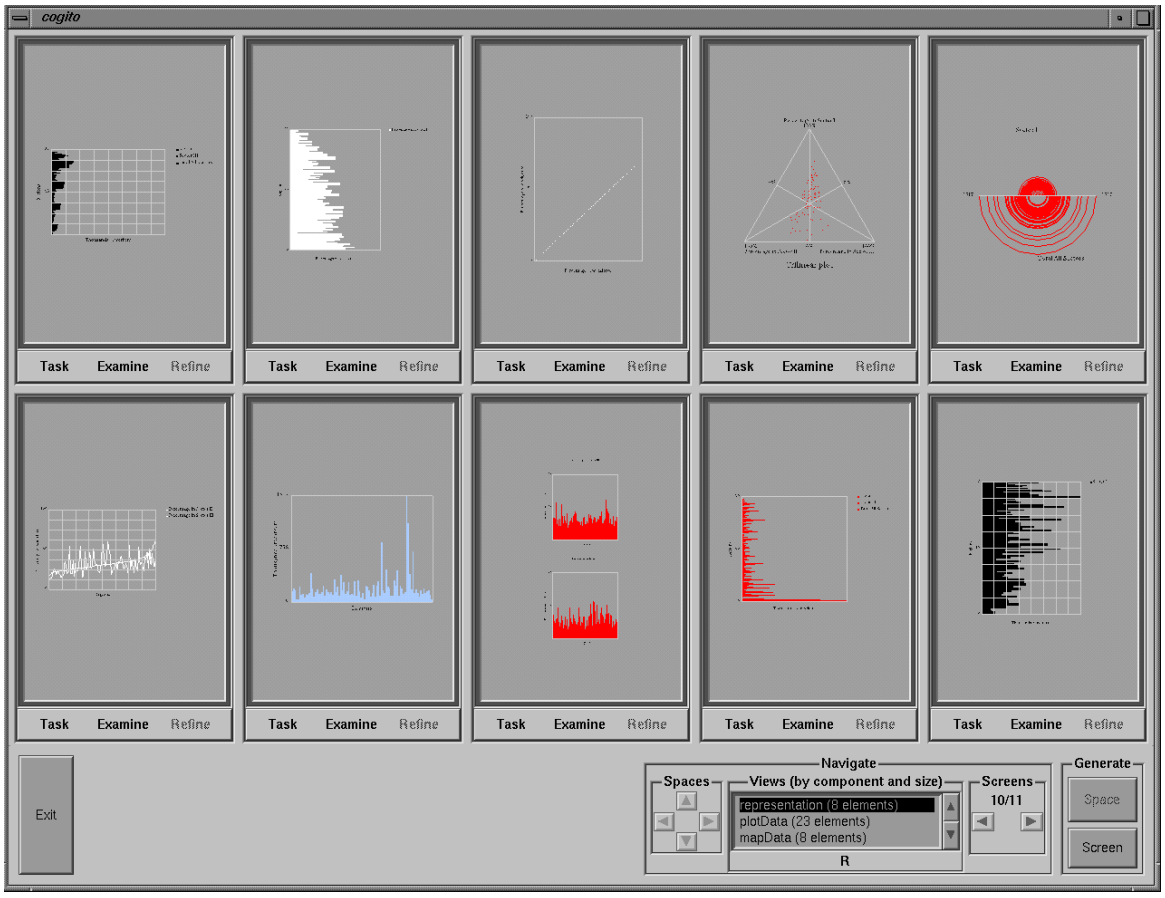


Figure 5.2: The *cogito* system running the *Graphs* application.

for a detailed example of the syntax involved.

```
abstraction
compatibilities
        base "Base compatibility" ( base "Base condition" )
        dimension "Dimensionality of graphs"
                ( "2D" "Two-dimensional graphs" )
        rep "Representation used in graphs"
                ( chart "Recti-linear chart", rose "Circular",
                tri "Triangular" )
        amplify "Use of amplified data"
                ( permitted "Amplified data is permitted",
                forbidden "Amplified data is forbidden" )
        anntype "Annotation type"
                ( scatter "Annotate scatter plot", col "Column chart",
                bar "Bar chart", line "Line chart",
                trilinear "Trilinear plot", rose "Nightingale rose" )
        lay "layout requirements"
                ( other "no special requirements",
                overSquare "overlaid with square aspect ratio" )
        dataRestrict "Restrictions on data"
                ( none "No restrictions",
                tri_100  "Three fields sum to unity" )
        implantation "coverage of mark on plane"
                ( point "point", line "line", area "area" )
```

Table 5.3: Abstraction for *Graphs* application: compatibilities.

Compatibilities and conditions are defined to enforce restrictions on how the elements from these components can be combined. In addition to eliminating nonsensical combinations, the `lay` compatibility is used to eliminate visually non-unique combinations by imposing the `overSquare` condition, which indicates that the layout should only be overlaid with a square aspect ratio. In addition to the named values used in the *Graphs* example, this application uses three reference catalogues. The contents of the catalogues are determined at run-time. However, because these catalogues are defined in the abstraction, elements may use the contents of the catalogue to define their arguments.

```
components
        camera "Viewing information" match dimension
        lighting "Lighting information" match dimension
        layout "Layout information for graphs"
                match base present lay
                attribute setstate graphPosition presentation aspectRatio
        size "Size of pieces"
                match base implantation lengthNeeds
                attribute setstate lineWidth pointSize
        texture "textures"
                match base
                attribute setstate linePattern
        colourPalette "Palette from which colours are chosen"
                match base
                attribute setstate colourPalette
        colourSelect "Method to select colours from palette"
                match base
                attribute setstate colourSelect
        plotData
                match dataRestrict
                attribute setstate plotData
        mapData
                match base lengthNeeds
                attribute setstate mapData
        dataAmplification
                match base amplify
                attribute setstate dataAmplification
        representation
                match dimension rep dataRestrict implantation
                      amplify lay anntype
                getstate plotData mapData lineWidth linePattern
                         colourPalette colourSelect presentation aspectRatio
                setstate boundingBox graphCount
        annotation
                match rep present anntype
                getstate graphCount
```

Table 5.4: Abstraction for *Graphs* application: components.

```
catalogues
        aspects "Graph aspect ratios" float
                ( square "Square" 1.0, vertical "Portrait" 1.333333,
                horizontal "Landscape" 0.75 )
        lm_cat "Available light models" integer
                ( base "Base colour" 0, phong "Phong illumination" 1 )
        pres_type "Presentation type" integer
                ( single "Graph showing a single plot" 0,
                overlaid "Several plots overlaid on a single graph" 1,
                separate "Several plots each on a separate graph" 2 )
        clr_space_type "Colour space type" integer
                ( RGB "Red Green Blue" 0,
                HSV "Hue Saturation Value" 1 )
        over_mode_type "Overlapping mode for charts" integer
                ( multi "Multiple symbols side by side" 0,
                stacked "Multiple symbols stacked one on another" 1 )
        relation "Relations in data" string
                ( percent100 "Fields in relation sum to 100 percent" )
        amplifier "Amplify data" string
                ( sort "sort data" srt_check )
        positions "Graph positioning function" reference graphPosition
        linewidths "Line width functions" reference lineWidth
        dotsize "Dot size functions" reference pointSize
```

Table 5.5: Abstraction for *Graphs* application: catalogues.

5.2 Defining the library

In general, there may be several libraries used for an application. Without loss of generality, this discussion will deal with only a single library.

The abstraction lays out the components which define the dimensions of the space of alternatives and the catalogues which define the structures for using the sampling modules. The library file realizes the abstraction by putting elements into the components and by adding designations to the reference catalogues. The library file describes the elements and designations which the DSO adds and then it connects the parts of the abstraction to executable code.

```
datasets
sectors "Sectors in an economy by region"
        sequential
        body (
                "Record Number"
                        float units ordinal sorted
                "Sector I"
                        float units "Thousands of workers"
                "Sector II"
                        float units "Thousands of workers"
                "Sector III"
                        float units "Thousands of workers"
                "Total All Sectors"
                        float units "Thousands of workers"
                "Percentage in Sector I"
                        float units "Percentage of workforce"
                "Percentage in Sector II"
                        float units "Percentage of workforce"
                "Percentage in Sector III"
                        float units "Percentage of workforce"
        )
        relations (
                percent100 (
                        "Percentage in Sector I"
                        "Percentage in Sector II"
                        "Percentage in Sector III"
                )
        )
```

Table 5.6: Abstraction for *Graphs* application: dataset formats.

One may think of the *cogito* structure as a three-tiered hierarchy: the components define the different parts of the visual representation, the elements define the distinct aspects of each component, and the parameters to the each element can control the quantitative aspects of an element's behaviour.

Both the *Shapes* and *Graphs* applications use elements which represent a single sample. This was done to provide more precise and equal control in the user study. Some alternative formulations are illustrated in following text.

5.2.1 Shapes

```
"Red" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 1.0
                green float range 0.0 1.0 value 0.0
                blue float range 0.0 1.0 value 0.0
        matches base base
        description "Red diffuse colour"

"Green" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 0.0
                green float range 0.0 1.0 value 1.0
                blue float range 0.0 1.0 value 0.0
        matches base base
        description "Green diffuse colour"

"Blue" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 0.0
                green float range 0.0 1.0 value 0.0
                blue float range 0.0 1.0 value 1.0
        matches base base
        description "Blue diffuse colour"
```

Table 5.7: Colour elements for *Shapes* application.

```
"Cyan" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 0.0
                green float range 0.0 1.0 value 1.0
                blue float range 0.0 1.0 value 1.0
        matches base base
        description "Cyan diffuse colour"

"Magenta" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 1.0
                green float range 0.0 1.0 value 0.0
                blue float range 0.0 1.0 value 1.0
        matches base base
        description "Magenta diffuse colour"

"Yellow" MAT_diffuse
        component colour
        arguments
                red float range 0.0 1.0 value 1.0
                green float range 0.0 1.0 value 1.0
                blue float range 0.0 1.0 value 0.0
        matches base base
        description "Yellow diffuse colour"
```

Table 5.8: Colour elements for *Shapes* application, continued.

The element descriptions in Tables 5.7 and 5.8 provide the functionality of several different elements by calling the same module (see Table 5.9) with different parameters. The benefit of this approach is that it is easy to evaluate a function at a few interesting discrete points and name them as such. Table 5.10 illustrates a more compact approach, which is beneficial especially when there are very many desirable alternatives in an element's parameter space. A validator module (Table 5.11) can be used to reject certain combinations of parameters. Therefore, the two implementations have the same effect, although they do have a slightly different interface. The modules for the *Shapes* application are quite straightforward. A more complex example will be given with the *Graphs* application.

```
SoGroup*
MAT_diffuse(Arg* args,int argc,SoSeparator*)
{
SbColor start;

        SoGroup* mat_group = new SoGroup;
        mat_group->setName("material");

        SoInfo* info = new SoInfo;
        mat_group->addChild(info);
        if (argc > 0) {
                info->string = SbString((char*) args[0].value);
        }

        SoMaterial* mat = new SoMaterial;
        if (argc > 3) { // load starting colour
                start = SbColor(*((float*)(args[1].value)),
                                *((float*)(args[2].value)),
                                *((float*)(args[3].value)));
        }
        mat->diffuseColor = start;
        mat->specularColor = SbColor(0.3,0.3,0.3);
        mat_group->addChild(mat);

        return(mat_group);
}
```

Table 5.9: The code to implement the `MAT_diffuse` module that is referenced in Table 5.7 and 5.8.

```
"Diffuse colour" MAT_diffuse no_bw
        component colour
        arguments
                red float range 0.0 1.0 samples 2
                green float range 0.0 1.0 samples 2
                blue float range 0.0 1.0 samples 2
        matches base base
        description "Diffuse colour"
```

Table 5.10: An alternative definition for an element which provides diffuse colours in the *Shapes* application. The arguments specify $2^3 = 8$ possibilities. The `no_bw` validator module is used to reduce that to 6 in total, the same ones specified in Tables 5.7 and 5.8.

```
/*
 * return TRUE only if the colour is not black nor white
 */
CogBool
no_bw(Arg* args, int argc)
{
CogBool rval;
        rval = FALSE;
        if ((args[1].value != 0) |
            (args[2].value != 0) |
            (args[3].value != 0)) {
                if ((args[1].value != 1) |
                    (args[2].value != 1) |
                    (args[3].value != 1)) {
                        rval = TRUE;
                }
        }
        return(rval);
}
```

Table 5.11: Validator module for the `MAT_diffuse` element module which eliminates the colours $(0, 0, 0)$ and $(1, 1, 1)$.

There is only one instance where state information is required and the CogLib API is not used at all. These features will be discussed through examples from the *Graphs* application.

5.2.2 Graphs

The library for the *Graphs* application contributes designations for referential catalogues, as well as elements. The syntax which accomplishes this is shown in Table 5.12.

```
elements
        "Trilinear plot" REP_trilinear
                component representation
                matches dimension "2D"
                        rep tri
                        dataRestrict tri_100
                        implantation point
                        amplify forbidden
                        lay overSquare
                        anntype trilinear
                description "Trilinear plot"

designations
        vertical_layout positions
                "Position graphs vertically to one another" GPOS_vert_sq
        horizontal_layout positions
                "Position graphs horizontally to one another" GPOS_horiz_sq

        constant_reg linewidths
                "Single, regular width lines" "line_constant_reg 1"
        constant_thick linewidths
                "Single, thick width lines" "line_constant_thick 1"
        varyExponentially linewidths
                "Vary line width exponentially" "line_exp 2"
        varyLinearly linewidths
                "Vary line width linearly" "line_linear 4"
```

Table 5.12: An excerpt of elements and designations made available in the *Graphs* application DSO.

Compatibility conditions were again used in this application for the benefit of the user

study. If one was to use a validator module instead, it would have the general form of the one in Table 5.13.

```
CogBool
trilinear_tester(Arg* args, int argc)
{
int i, count = argc - 1;
CogBool tf;
DataIndex di;

        tf = TRUE;
        di[0] = 0; di[1] = 0;
        if (count == 3) {
                for (i = 1;i <= argc-1;i++) {
                        di[2] = (int) args[i].value;
                        if (CogLib::hasRelation("percent100",di) == FALSE ||
                            CogLib::getRelationSize("percent100",di) != 3) {
                                tf = FALSE;
                                break;
                        }
                }
        }
        else {
                tf = FALSE;
        }
        return(tf);
}
```

Table 5.13: A validator module for the "Trilinear plot" representation which uses the CogLib API to test whether the necessary relation exists and has the proper size.

5.3 Specifying the Display and Configuration

A user may choose to establish any viewing order for the components. The order in which the components are viewed through the interface is completely separate from notion of the processing order which was fixed in the abstraction. The `views` keyword handles reordering the components, and it can also be used specify new names for the components in the interface. Examples are shown in the following subsections in Tables 5.14 and 5.15.

5.3.1 Shapes

```
views
        shape
        colour
        squeeze
        orientation
        lightModel
        lightSource
        lightPosition
        lightDirection

display
        interleaved
        gridsize 2 5 1.29
        policy all
```

Table 5.14: Specification of display parameters for the *Shapes* application.

5.3.2 Graphs

```
views
        representation
        plotData
        mapData
        (dataAmplification sort)
        layout
        annotation
        colourPalette
        colourSelect
        size

display
        interleaved
        gridsize 2 5 1.29
        policy all
```

Table 5.15: Specification of display parameters for the *Graphs* application. In this example, the `dataAmplification` component is aliased to the name `sort`.

5.4 Implications

The process of developing the two applications, *Shapes* and *Graphs*, for the *cogito* system has underscored the flexibility of the interface for both user and programmer. Each is provided with multiple means of accomplishing their task. Although the basis for the interface is now established, it can be quite tedious to use. There remains a need for tools which can support the programmer in developing code for particular applications with particular abstractions. Before further development takes place, it is important to assess the present design. Such an assessment is presented in the next chapter.

Chapter 6

Assessment

The design of the *cogito* system [113, 114] began as a new expression of the possibility of improved access to computer-based visualization technology [127]. Chapters 3 and 4 have described the development of that design and its implementation. Chapter 5 has demonstrated the expressiveness of the system for building applications. This chapter completes the presentation by describing the user study which was conducted to assess how well the software meets its goals.

The foundation of the *cogito* system design was validated through informal evaluation of early prototypes by peers. A detailed user study was conducted in order to assess the effectiveness of the paradigm in the context of a real application, and to collect empirical data about how people visualize with the aid of a computer. The following sections discuss all aspects of the test, from design to analysis of the results.

6.1 Test Structure

One would like to determine whether or not the *cogito* software system encourages insight. The challenge, especially in studying such an ineffable quantity as insight, is to capture relevant information in a tractable way. In Rubin's practical guide to usability testing [212], he describes four different test types: an *exploratory* test evaluates the effectiveness of design concepts, early in the development cycle; an *assessment* test examines and evaluates the implementation of design concepts, early or midway through the development cycle; a *validation* test evaluates software against predetermined benchmarks, late in the development cycle; and a *comparison* test is used at various points in the development cycle to better

understand the advantages and disadvantages of different designs.

In the case of this work, the role of the exploratory tests was taken by the informal evaluations from peers. The *cogito* system has now matured to the point at which a robust implementation of the concepts exists. However, it has not yet reached the point for a validation test, and the benchmark against which the software could be compared does not yet exist. Rather, the present study is primarily an assessment test of the basic system theory and design. The study includes a comparison test of two interface approaches, flat (A) and hierarchical (B), which provide different forms of access to the space of alternative visual representations, produced by the same underlying system. Subjects were asked to complete a task, whose solution may require insightful and creative thinking, using one of the two interfaces.

6.2 Software

The full *cogito* system as it is envisioned and implemented contains many features which differ to varying degrees from popular current software tools, meaning that substantial training of users may be required. Two simplified versions of the full interface were developed in order to focus attention on the means of accessing alternatives. Facilities to log user interaction were added to both interfaces. The most explicit of these additions was the "Task" button, with which users were asked to indicate their use of a particular visual representation. Other features of the software were hidden from the subjects as much as possible in order to create the situation that access to the space of alternatives was the sole condition of the study.

6.2.1 Interface A (Flat)

This interface provides the user with an unstructured, flat view of the space of alternative visual representations (see Figure 6.1). This interface evolved from the edit dialogue box in the full *cogito* system, illustrated in Figure 4.4. The user is allowed to make incremental changes to the image and see the results immediately. Additionally, a history of those changes is kept and can be accessed at any time.

Its display of a single visual representation at a time and its requirement for the explicit specification of elements make it similar to existing systems. However, it employs a kind of interaction more closely related in style to "form fill-in" [224] than the visual programming

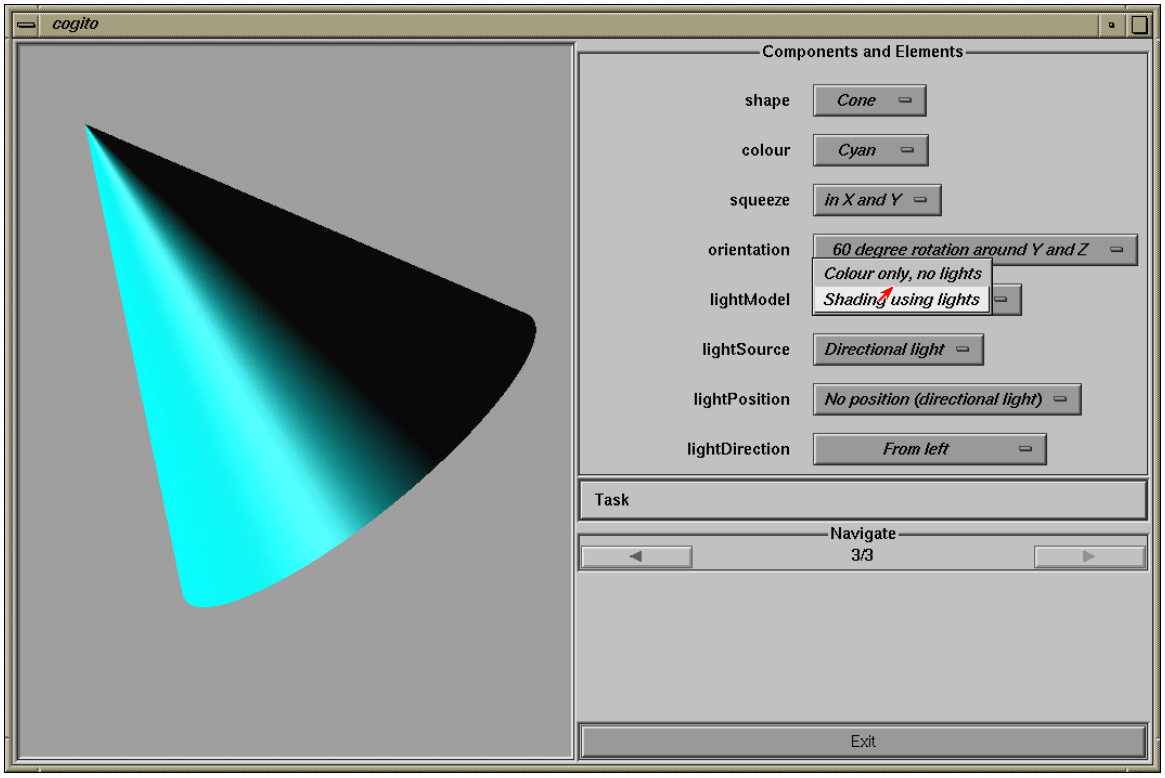


Figure 6.1: Interface A (Flat) with the *Shapes* application.

familiar from modular visualization environments. The left side of the window displays the visual representation corresponding to the settings of components and elements listed on the right side of the window. The user may pan and zoom the visual representation using the mouse. Compatibilities are checked from the top of the list to the bottom. The user indicates the use of an element for a component by selecting it in the option menu next to the component name. Only elements compatible with those earlier in the list are available for selection. If selection of a new element for some component is incompatible with the current element for another component later in the list, the element from the latter component is changed to make a compatible combination. It is assumed that this is always possible to do.

The user reference material for this interface, which includes a tutorial, can be reviewed in Appendix A on pages 128–130.

6.2.2 Interface B (Hierarchical)

This interface (see Figure 6.2) provides the user with a structured, hierarchical view of the space of alternative visual representations. This interface evolved from the main display of the full *cogito* system. It makes the user aware of options by showing a collection of visual representations from a regular sample of the space of available representations. Based on his or her own requirements, the user possibly examines (see Figure 6.4) then chooses suitable candidate visual representations from the collection of those displayed. The system uses this information to determine which specific elements are of interest and composes a new space of alternatives consistent with those selections.

The user is allowed to change the organization of the space using a predefined list of views, and navigate the hierarchy of spaces which he or she might create. For any selected visual representation, the user may exercise additional control by bringing up the "Refine" dialogue box (shown in Figure 6.3). The usual semantics of a selection are that only the elements in the selected combination are used in the definition of the next space. This may be altered so that for a particular component, all elements from the current space are included in the next (no change) or all elements from the root space are included in the next (possibly increasing the number of alternatives). This capability is also present in the full *cogito* system. The capability to directly edit visual representations is hidden in this interface, since it forms the basis of Interface A.

The user reference material for this interface, which includes a tutorial, can be reviewed

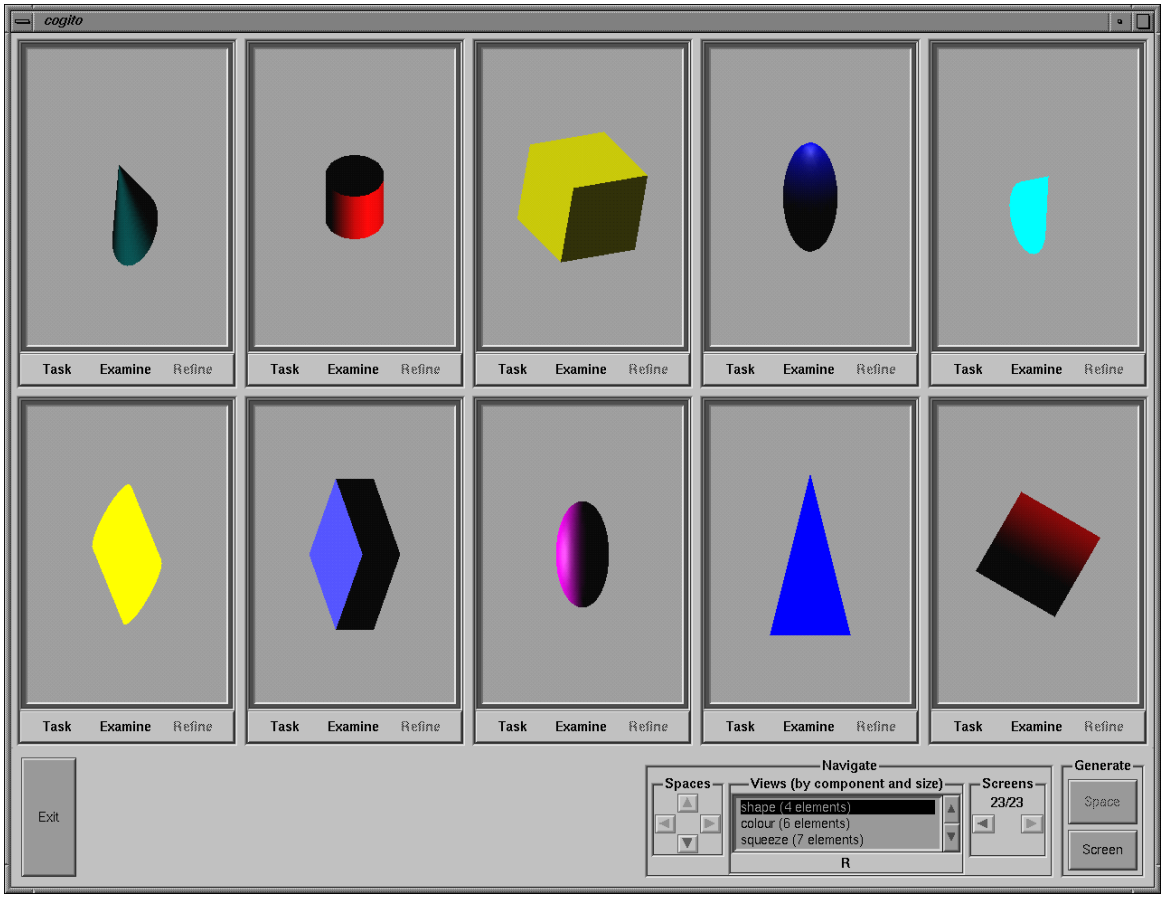


Figure 6.2: Interface B (Hierarchical) with the *Shapes* application.

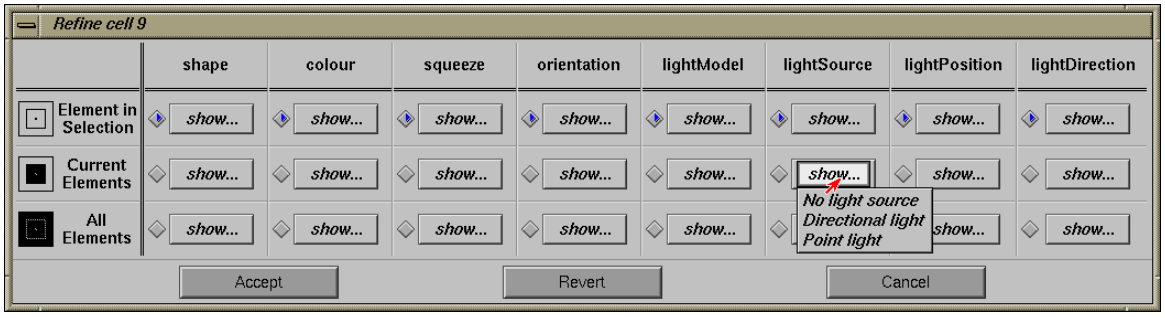


Figure 6.3: Refine dialogue box for Interface B (Hierarchical) with the *Shapes* application.

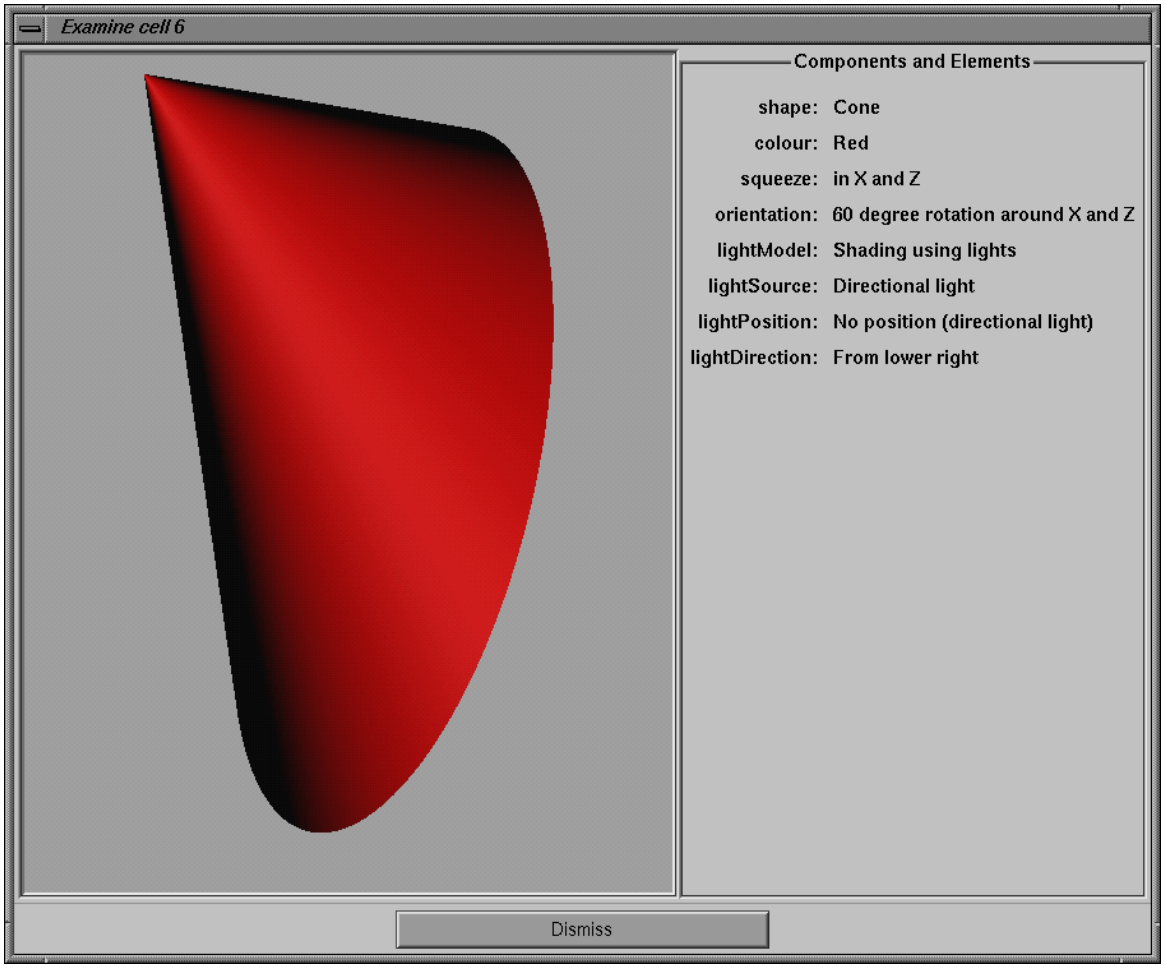


Figure 6.4: Examine dialogue box for Interface B (Hierarchical) with the *Shapes* application.

in Appendix A on pages 131–135.

6.3 Hypotheses

This dissertation has argued that the effectiveness of an image is determined by personal criteria, and it has presented a computer-aided visualization system designed to help individuals use those criteria to find effective imagery. The thesis of this dissertation is that involvement and exploration are important activities for finding personally-effective imagery and that software can be designed to encourage these activities. This leads immediately to two hypotheses for testing.

The first testing hypothesis deals with whether the activities of involvement and exploration are important to visualization. The null hypothesis for this case is:

$$H_0\text{−}\texttt{activity} = \text{there is no preference for involvement and exploration}$$

The second testing hypothesis concerns the design of the software. The same underlying system was used for each of the two interfaces, so the interfaces were closer in function than would have been the case if an existing and separate commercial package was used for comparison. Considering the importance of searching, the two interfaces correspond broadly to different search methods. The flat interface (A) supports a local search, and the hierarchical interface (B) supports a global search. Both interfaces support the user in choosing visual representations which meet personal criteria for effectiveness. The flat interface (A) gives the user explicit control over the content of each visual representation, and the hierarchical interface (B) gives the user explicit control over the content of spaces of available alternatives. The null hypothesis for this case is:

$$H_0\text{−}\texttt{interface} = \text{there is no preference for either interface}$$

6.4 Experiment Design

Subjects were recruited through e-mail, primarily from the undergraduate population in the Faculty of Applied Sciences at Simon Fraser University. The e-mail was sent to the lists of Computing Science, Mathematics and Statistics, Engineering Science, and Business undergraduate and graduate students. The text of that message is presented in Appendix A, on page 121. The goal was to recruit forty computer-literate adult subjects.

Subjects were asked to volunteer approximately one hour of their time. The fact that subjects received no remuneration was one of the constraints of the experiment design. It encouraged subjects with a high level of interest, and it limited the length of the test session. All subjects were required to sign a consent form at the start of the session. The subjects were randomly assigned to one of two groups (A or B). Each group used only one of the two software interfaces.

This experiment design [46] (also known as between-subjects, as opposed to within-subjects) was chosen for two main reasons:

- the underlying interface of components and elements was the same and it was felt that in a within-subjects design, subjects' performances with their second interface would benefit from what was learned while using their first interface
- the time required for each volunteer subject to use both interfaces would have been prohibitive

The primary instrument for data collection was the questionnaire, because of the largely subjective nature of the study and difficulty of quantifying the data by other means. Charlton [41] served as a useful reference in the design of the questionnaires, which were reviewed extensively before the study began.

The experiment session comprised the pre-task questionnaire, training task, main task, and post-task questionnaire.

6.4.1 Pre-task questionnaire

This questionnaire was designed to gain background information about the subjects, which could be used for subsequent analysis. An exact copy of this questionnaire is presented in Appendix A, on pages 125–126. Two types of information were requested: objective information about gender, age, years of education, and main area of interest. Then the subjects were asked to make more subjective judgements about their use of graphs, their creation of graphs, and their level of expertise with graph-generating software and computer graphics generally.

Although it is a popular notion that people may have a preference for visual or verbal thinking, it has been difficult to construct a test which can effectively isolate this preference. For example, Richardson [206] published the Verbalizer/Visualizer Questionnaire (VVQ),

but attempts to substantiate its effectiveness have not succeeded [94]. Another possible approach to gather this information is through learning style preferences, where Kolb's Learning Style Inventory [139] has become a widely used metric. The additional cost of collecting this information and the potential negative impact of the questions on the subjects were the prime factors in the decision not to collect it.

6.4.2 Training Task

The requirement for the training task was to train the subjects in the software without training them in the main task problem. In order to minimize the task-specific learning effects, the training task was designed to use a completely different application with a different character of questions. The application selected for this task was *Shapes*, described in Chapter 5. The subjects first completed a tutorial which gave them experience using the different features important to the study, including the "Task" button. They were then asked the following two questions, which were intended to focus the subjects on using the system to explore alternatives and to assess those alternatives based on their own criteria.

1. which combination do you like the best?
2. which combination has the most 3D character?

6.4.3 Main Task

The requirement for the main task was to engage the subjects with an accessible problem motivated by a real application. The application selected for the main task was *Graphs*, also described in Chapter 5. The graphing task was considered to be sufficiently realistic to motivate the subjects, yet not so specialized or difficult that it would restrict the pool of potential subjects. It was considered to be a good match to the skills expected in the population from which the subjects were recruited.

The task for this application focused on using graphs to answer questions about data. The data is taken from page 100 of *Semiology of Graphics* [12] and it provides a view of the French economy from the early 1960's. For each département in France, the data provides: the workforce (in thousands of workers) for each of the three sectors (primary, secondary, and tertiary) in the economy; the total workforce (the sum of the three sectors); and the percentage of the workforce in each sector. The raw data is provided in Appendix A,

beginning on pages 138–140. Although its source was explained to the subjects, they were instructed to focus on the data itself. The questions posed were chosen to emphasize the three different types of reading one might use with a graph, identified by Bertin [12].

1. elementary, where one is interested in a specific fact
   (a) Is there a region whose workforce is evenly distributed amongst the sectors?
   (b) What is an estimate for the median total workforce of the regions?
2. intermediate, where one is interested in characterizations of facts
   (a) Can you characterize the regions in which Sector I predominates?
   (b) Can you describe, in general, the size of Sector I in regions where Sector III predominates?
3. overall, where one is interested in relationships between different characterizations
   (a) Is there a discernible relationship between the 3 sectors?
   (b) Can you find any other relationships in the data?

Participants had access to the reference manual and were encouraged to ask any questions regarding the components, the elements, the interface, or how to accomplish a goal using the interface. As much as possible, the participants were left to work alone on the questions. However, since the goal was not to test the participants' ability to answer the questions, help was given as needed.

6.4.4 Post-task Questionnaire

A total of seventeen Likert-scale (requesting a rating on a scale of the form: "Strongly Disagree", "Disagree", "Agree", "Strongly Agree") and four open-ended questions were posed on this questionnaire. The full questionnaire is provided in Appendix A, on pages 141–143.

The questionnaire was designed to assess subjects' feelings about their completion of the task; the degree to which the software helped them; and their impressions of the task. The written questions were meant to solicit which features were liked and disliked, which features were missing, and any other comments about the system.

6.5 Analysis of Results

Study session data was collected both from questionnaires and through interface selections. All responses to multiple-choice and Likert-scale questions were coded with integers beginning with 1 for the first response on the question. Missing responses were coded with 0. For example, on a question which asked for a rating of "Low", "Medium", or "High", the responses would be coded as follows:

Response	Code
No response	0
"Low"	1
"Medium"	2
"High"	3

This data is *ordinal* because the response categories for each question can be ordered. For every visual representation marked by the "Task" button during the main task, its combination of elements was recorded. This data is *nominal* because each element is named, with no quantitative connection to other elements. The *chi-square* test was used to analyse both the questionnaire responses and the selections of visual representations because both sets of data are categorical and this test allowed comparison of the proportions between categories. Version 9.0 of the SPSS (Statistical Package for the Social Sciences) software [125] was used to perform the analyses.

6.5.1 Pre-task questionnaire

A total of thirty-four computer-literate adults volunteered as subjects in the study, which permitted seventeen subjects to be assigned to each interface (identified as Groups A and B). All responses to the pre-task questionnaire are summarized, by interface, in Table 6.2. For each question, the frequencies for the different response categories were analysed with the one-sample chi-square, or goodness-of-fit, test. The null hypothesis was that the frequencies for each response category were equal. For all questions except one, the probability (p value) that the frequencies could be observed if the null hypothesis was true was sufficiently low to enable the null hypothesis to be rejected. The results for all subjects are shown in Table 6.1. Groups A and B were each tested using this same procedure, with the null hypothesis that the frequencies for each response category in the group (A or B) were the same as those for

all subjects. The results indicated that the null hypothesis could not be rejected for any question. Note that the size of the groups ($n_A = n_B = 17$) decreases the reliability of the tests in some cases because the expected count per response category becomes too small.

Within the pre-task questionnaires, a significant relationship ($p < 0.01$) was indicated between graphing software experience (q9) and other software experience (q10).

Topic	**Mode†**	**Significance**
gender	male	$p < 0.01$
age	18–25 years	$p < 0.01$
post-secondary education	1–4 years	$p < 0.01$
area of interest	science / applied science	$p < 0.01$
graph-reading frequency	weekly	$p < 0.05$
graph-creating frequency	monthly	$p < 0.05$
preferred graph-creation method	computer	$p < 0.01$
preference for text, graph, or both	both	$p < 0.01$
level of experience with graph-creating software	medium	$p < 0.01$
level of experience with other software which creates graphs as an auxiliary purpose	medium	*not significant*
level of experience with computer graphics	medium	$p < 0.01$

† mode is the response which occurs most frequently

Table 6.1: Analysis of frequencies of response categories for the pre-task questionnaire data for all subjects ($N = 34$).

6.5.2 Post-task questionnaire

The responses to the Likert-scale questions and the open-ended questions are considered and presented separately.

Likert-scale data

The frequencies of response categories for these questions was analysed with a chi-square test (H_0 = all response categories are equally likely) and it was found that H_0 could be rejected for all questions ($p < 0.01$). There was a trend towards positive answers, which can be expected in this type of study [203].

The following significant relationships were found between questions on the post-task questionnaire:

Topic	Response Category	Interface A		Interface B	
q1. Gender	Male		11	15	
	Female		6	2	
q2. Age	18 – 25		10	11	
	26 – 35		5	5	
	36 – 49		2	1	
	50+		0	0	
q3. Years of post-secondary education	0		1	0	
	1 – 4		8	11	
	5 – 6		2	2	
	more than 6		6	4	
q4. Academic area of interest	Fine Arts		0	0	
	Business		1	0	
	Arts/Humanities		0	0	
	(Applied) Science		16	17	
q5. Frequency of reading graphs	Daily		4	6	
	Weekly		7	8	
	Monthly		5	2	
	Less than monthly		1	1	
q6. Frequency of creating graphs	Daily		1	1	
	Weekly		3	3	
	Monthly		7	7	
	Less than monthly		6	6	
q7. Preferred method to create graphs	Pencil & paper		1	2	
	Computer		16	15	
	Other		0	0	
q8. Preferred source from which to intrepret data	Text		0	1	
	Graphs		4	8	
	Text and graphs		13	8	
q9. Level of experience producing graphs with software	Low		5	5	
	Medium		10	9	
	High		2	3	
q10. Level of experience with software having graphics as secondary function	Low		4	7	
	Medium		10	5	
	High		3	5	
q11. Level of familiarity with computer graphics	Low		2	2	
	Medium		9	8	
	High		6	7	

Table 6.2: Pre-task questionnaire responses, grouped by interface.

Topic	Response Category	Interface A		Interface B	
Q1. Helpfulness of past experience	Very unhelpful		1	1	
	Unhelpful		5	5	
	Helpful		9	9	
	Very helpful		2	2	
Q2. Feeling about how task was completed	*No response*		1	0	
	Very unsatisfied		1	3	
	Unsatisfied		10	4	
	Satisfied		5	10	
	Very satisified		0	0	
Q3. Effectiveness of chosen graphs	Very ineffective		0	0	
	Ineffective		1	2	
	Effective		14	12	
	Very effective		2	3	
Q4. Number of alternatives	Very small		0	0	
	Small		0	4	
	Large		12	8	
	Very large		5	5	
Q5. Accessibility of alternative graphs	Very poor		0	0	
	Poor		0	2	
	Good		11	11	
	Very good		6	4	
Q6. Helpfulness of software for accessing alternatives	Very unhelpful		0	0	
	Unhelpful		3	2	
	Helpful		11	8	
	Very helpful		3	7	
Q7. Helpfulness of exploring	Very unhelpful		0	0	
	Unhelpful		0	4	
	Helpful		8	8	
	Very helpful		9	5	
Q8. Helpfulness of software for exploring alternatives	Very unhelpful		0	0	
	Unhelpful		2	2	
	Helpful		11	9	
	Very helpful		4	6	
Q9. Helpfulness of refining alternatives	Very unhelpful		0	0	
	Unhelpful		0	1	
	Helpful		10	5	
	Very helpful		7	11	

Table 6.3: Post-task questionnaire responses for questions 1 through 9, grouped by interface.

Topic	Response Category	Interface A		Interface B
Q10. Helpfulness of software for refining alternatives	Very unhelpful		0	0	
	Unhelpful		3	3	
	Helpful		11	11	
	Very helpful		3	3	
Q11. Usability of software	Very poor		0	0	
	Poor		2	3	
	Good		12	12	
	Very good		3	2	
Q12. Choose a commercial version of software to produce graphs	Very unlikely		2	4	
	Unlikely		8	7	
	Likely		6	4	
	Very likely		3	2	
Q13. Choose a commercial version of software to understand data	Very unlikely		0	1	
	Unlikely		5	8	
	Likely		11	4	
	Very likely		1	4	
Q14. Software compared to others	Much worse		0	0	
	Worse		2	4	
	Better		10	12	
	Much better		5	1	
Q15. Develop understanding	Very unhelpful		0	0	
	Unhelpful		1	4	
	Helpful		12	8	
	Very helpful		4	5	
Q16. Develop questions	Very unhelpful		0	0	
	Unhelpful		4	5	
	Helpful		12	9	
	Very helpful		1	3	
Q17. Involvement	Very unhelpful		0	0	
	Unhelpful		1	2	
	Helpful		9	11	
	Very helpful		7	4	

Table 6.4: Post-task questionnaire responses for questions 10 through 17, grouped by interface.

- satisfaction (Q2) and effectiveness of graphs (Q3) ($p < 0.05$)
- satisfaction (Q2) and helpfulness of software for access (Q6) ($p < 0.05$)
- effectiveness of graphs (Q3) and the helpfulness of the software in exploration (Q8) ($p < 0.05$)
- effectiveness of graphs (Q3) and helpfulness of software in developing understanding (Q15) ($p < 0.01$)
- helpfulness of software for access (Q6) and helpfulness of software for exploration (Q8) ($p < 0.05$)
- helpfulness of software for exploration (Q8) and helpfulness of refining (Q10) ($p < 0.01$)
- helpfulness of software for exploration (Q8) and helpfulness of software in developing understanding (Q15) ($p < 0.01$)
- likelihood of using commercial version for producing graphs (Q12) and understanding graphs (Q13) ($p < 0.01$)
- comparison to other software (Q14) and development of understanding (Q15) ($p < 0.05$)

This data was analysed with the chi-square test to compare the responses from all subjects within groups A and B. Concerning $\mathtt{H_0}$`-interface`, whether one interface was preferred, there was not sufficient evidence to reject it. With respect to $\mathtt{H_0}$`-activity`, whether exploration and involvement are helpful, an artificial dichotomy was created from the responses to Questions 7 (exploration) and 17 (involvement) by assigning "Very unhelpful" and "Unhelpful" to "Negative" and "Helpful" and "Very helpful" to "Positive". For this analysis, a *contingency table* was built (see Table 6.5.2) to analyse the four combinations of responses to the two questions. Because of the small sample size, Fisher's Exact test was used to compute the significance. It indicated that $\mathtt{H_0}$`-activity` could be rejected, $p < 0.05$.

Following this, the subjects were divided into two groups, based on whether they were positive about both exploration and involvement (Group P, $n_P = 29$) or not (Group N, $n_N = 5$). A chi-square test was again performed on the subjects in Group P, dividing them on the basis of the interface they used. For Group P, there was a significant difference

Exploration	Involvement	
	Negative	Positive
Negative	2	2
Positive	1	29

Table 6.5: Contingency table for exploration and involvement: the relationship between exploration and involvement is significant ($p = 0.031$), so $\texttt{H}_0\texttt{-activity}$ can be rejected.

($p < 0.05$) based on the interface used for the issue of whether the software was helpful for accessing alternatives (Q6). The preference was for Interface B. Although $\texttt{H}_0\texttt{-interface}$ can be rejected for Group P, one needs to account for the members of Group N. For example, 4 of the 5 subjects in Group N used Interface B. Comparing groups N and P with a chi-square test, H_0 is that there would be no difference in question responses in the Groups compared to all subjects. H_0 was rejected ($p < 0.05$) for helpfulness of prior experience (Q1), effectiveness of graphs (Q2), and helpfulness of software for developing understanding (Q15) which indicates that these three were good indicators of the group (P or N) to which a subject would belong. One ready hypothesis to explain these results is that the users of Interface B were either very positive or very negative about it. A longer training period may well improve the impression of those in Group N.

Likert-scale questionnaire data can also be analysed [121] by creating a cumulative score for the entire questionnaire by summing the scores of all questions. Table 6.7 shows the distribution of these scores by interface. Two groups, L and H, of equal size ($n_L = n_H = 17$) were formed by sorting on the score and then assigning cases 1 to 17 to the first group, L (score ≤ 51) and cases 18 to 34 to the second group, H (score ≥ 51). Two cases had scores of 51, and based on the ordering done by SPSS; the first of the pair was assigned to group L and the second to group H.

Again, a chi-square test was used to test the frequencies of response categories in the groups with that of all subjects. The null hypothesis was that the proportions would be the same. H_0 could be rejected for Q12 (likelihood of using a commercial version of the software to produce graphs) ($p < 0.05$). H_0 could also be rejected for the higher scoring group, H, on the basis of age ($p < 0.05$).

The overall score was significantly related to the following questions:

- effectiveness of graphs (Q3) ($p < 0.05$)

	Groups		
Condition	Description	Label	Size
Interface used	Interface A	A	17
	Interface B	B	17
Cumulative score	score $\geq$ 51 (high)	H	17
	score $\leq$ 51 (low)	L	17
Rating of involvement & exploration	positive	P	29
	negative	N	5

Table 6.6: Summary of groups used for the analysis of questionnaire data.

- helpfulness of software for access (Q6) ($p < 0.05$)
- helpfulness of software for exploration (Q8) ($p < 0.01$)
- likelihood of using commercial version for understanding (Q13) ($p < 0.05$)
- helpfulness of software in developing understanding (Q15) ($p < 0.05$)

The responses to the post-task questionnaire are summarized in Tables 6.3 and 6.4.

Comments

Subjects were presented with four open-ended questions regarding their use of the software: "which features did you like the most?", "which did you like the least?", "are there features you'd like to see added to the software?", and "do you have any other comments about the software or the task?"

The responses to these questions were very interesting. Their full text is presented in Appendix B, organized by question and interface. Most of the comments dealt with the interface and the interaction style required by the software. As might be expected, the same features (i.e. zoom capability) appeared on both 'most-liked' and 'least-liked' lists. Specific features of the application were singled out as least-liked. Several comments were also made about the application and how it was organized. For example, some subjects commented about the components included in the application and how they were named.

For users of the flat interface (A), the speed of effecting changes and the easy access to a wide variety of options, like the ability to sort by data which was not plotted, were important.

Interface A		Score	Interface B
	0	68	0
	0	67	0
	0	66	0
	0	65	0
	0	64	0
	0	63	0
	0	62	0
	1	61	0
	0	60	0
	0	59	1
	0	58	1
	2	57	1
	1	56	2
	0	55	1
	2	54	1
	0	53	1
	2	52	0
	2	51	0
	0	50	2
	1	49	1
	2	48	1
	2	47	0
	1	46	0
	0	45	1
	0	44	2
	0	43	0
	1	42	0
	0	41	1
	0	40	0
	0	39	0
	0	38	0
	0	37	0
	0	36	0
	0	35	1
	0	34	0

Table 6.7: Distribution of cumulative scores, derived from post-task questionnaire responses. The scores are given in the center column with the count from Group A and B displayed to either side. The lowest value on the graph is 34, which corresponds to an average of 2 ("unsatisfied"), 51 corresponds to an average of 3 ("satisfied"), 68 corresponds to an average of 4 ("very satisfied").

> " ease of changing minor graph parameters (most packages I've used don't have such a capability)."

> "I liked the ability to see all the different representations of the data using the different plots. It is better than being forced to use only one representation to interpret data and answer questions."

> "having the ability to play with the data, and see it in ways that were meaningful to me – and being able to build that meaningful representation."

However, some comments pointed to larger issues not well addressed by the flat interface (A):

> "unable to see several different graphs at the same time"

> "having too much there – would be easier with time I think."

For the hierarchical interface (B), the hypothesized benefits of the approach were echoed in user comments:

> "ability to generate various possibilities that I would not have necessarily have (sic) thought to use."

> "the way many graphs are different options displayed at same time lets it be easy to see which one is best for analysing data."

> "creating different representations/versions from a graph that was almost what I wanted was useful."

Some shortcomings of the hierarchical interface (B) also became apparent in the testing, as non-visual items seemed harder to select. This fact is echoed in the comments, as is the desire for a more direct style of interaction:

> "the fact that it was hard to quickly know which elements were present in a certain cell, you have to examine it closely."

> "I felt constrained by the refine dialogue when I just wanted to select specific elements that were obviously desirable."

6.5.3 Selected graphs

A variety of images were chosen by participants. Figure 6.5 shows four selections made within the *Shapes* application (used for the training task) and Figure 6.6 shows four selections made within the *Graphs* application (used for the main task).

One hundred and fifty-one graphs were marked using the "Task" button: 96 graphs through Interface A and 55 graphs through Interface B. Because use of the "Task" button was not mandatory, the fact that fewer graphs were marked by users of interface B is not necessarily meaningful.

Using SPSS [125], 55 cases were randomly selected from the 96 marked by users of interface A and then a chi-square test was done on the distribution of selected of elements for each component. The null hypothesis that each element has an equal chance of being selected is equivalent to choosing that component first (when no other compatibility conditions are present). In almost all cases, this null hypothesis could be rejected as definite patterns of selection emerged ($p < 0.01$). The layout ($p = 0.02$), colourPalette ($p = 0.601$), and colourSelect ($p = 0.022$) components through Interface B were the only exceptions. The patterns of selection are presented in Figures 6.7 through 6.15. Line charts are used to emphasize the relationships between the data sets for the different interfaces, especially their crossings.

The *Graphs* application made typically unfamiliar representations (such as the trilinear plot [12, 257]) available to the subjects. Although the experiment was not designed to rigourously measure the use of these novel representations, selections from the two interfaces exhibited different patterns (Figure 6.7). In general, it seems that the peaks in the graphs of selected elements (see Figures 6.7 to 6.15) were roughly matched between Interface A and B, and that the choices for Interface B are more evenly distributed. The one notable exception is the distribution of selections for elements in the plotData component.

Finally, the selections in each interface are summarized in Figure 6.16 (for interface A) and Figure 6.17 (for interface B).

6.6 Discussion

Subjects were told that the study session would last approximately one hour. For those who were not finished in the hour, if time permitted and the participant was interested, he or she was allowed to continue. Because the actual tasks were less important than their use of

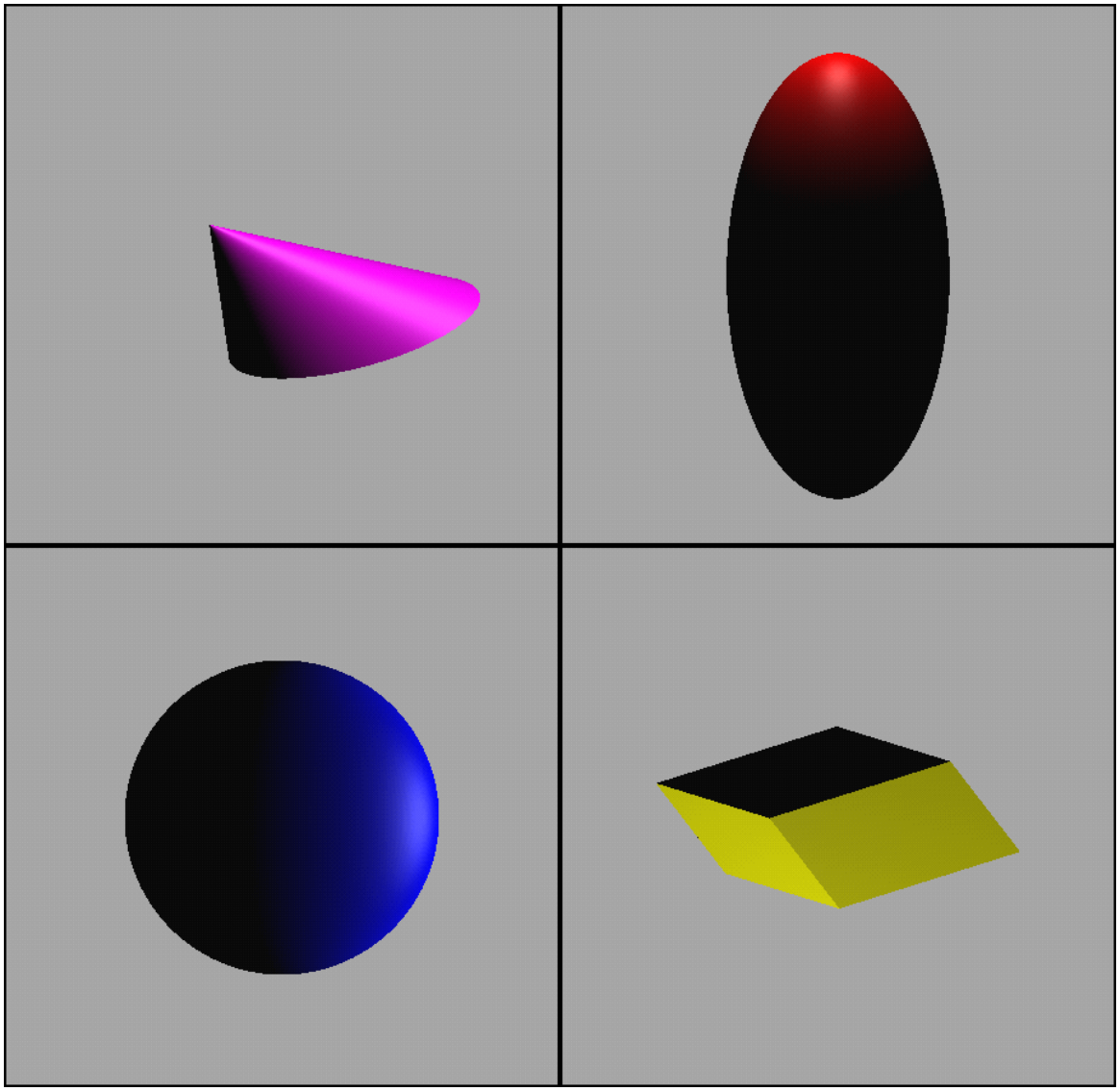


Figure 6.5: Four selected images from the *Shapes* application.

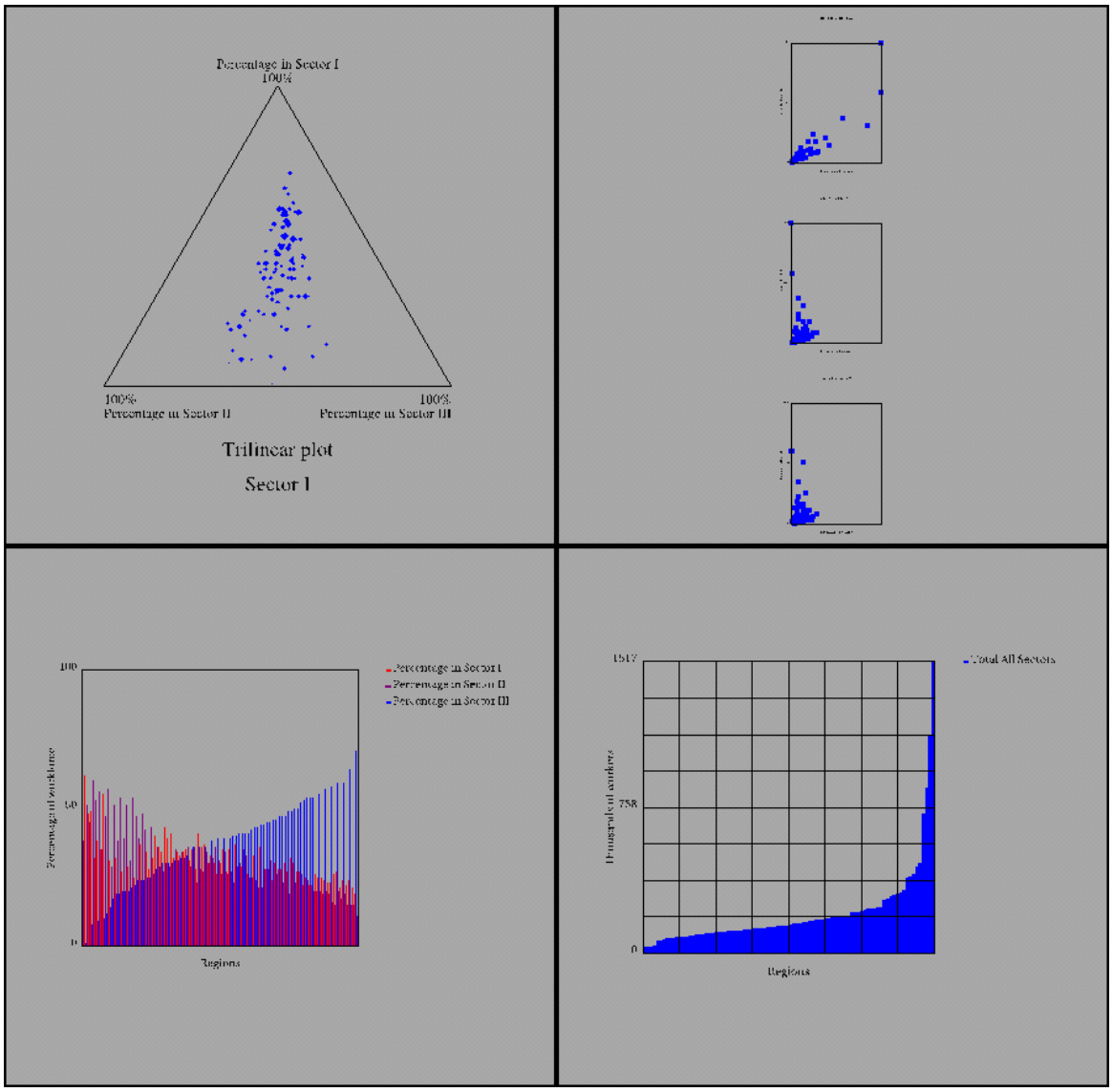


Figure 6.6: Four selected images from the *Graphs* application.

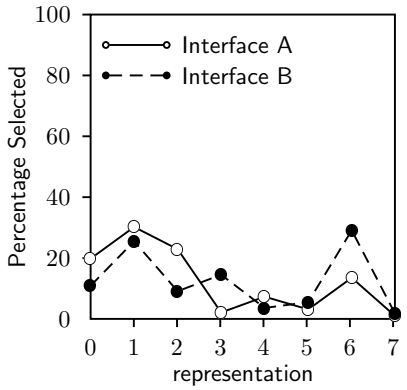


Figure 6.7: Frequencies of representation element choices.

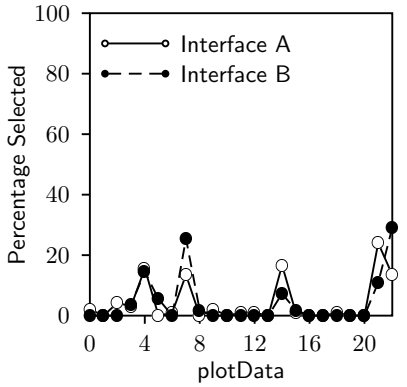


Figure 6.8: Frequencies of plotData element choices.

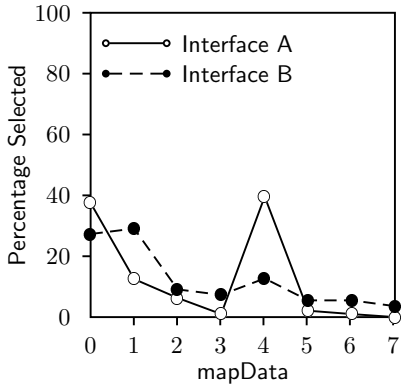


Figure 6.9: Frequencies of mapData element choices.

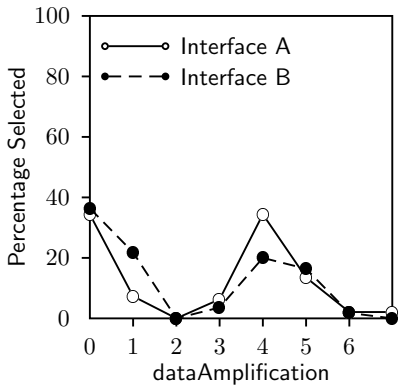


Figure 6.10: Frequencies of dataAmplification element choices.

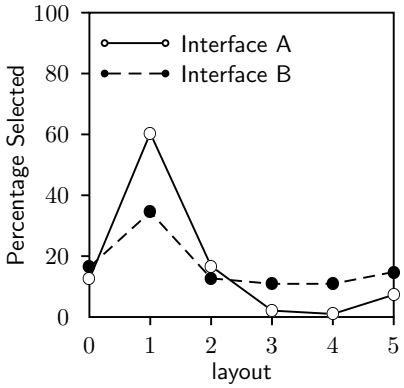


Figure 6.11: Frequencies of layout element choices.

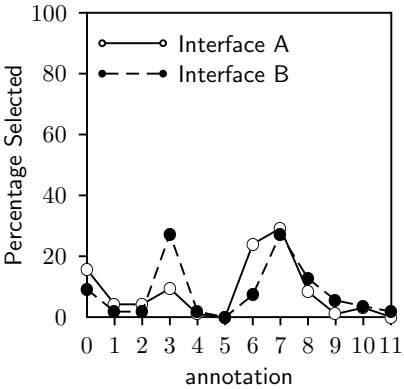


Figure 6.12: Frequencies of annotation element choices.

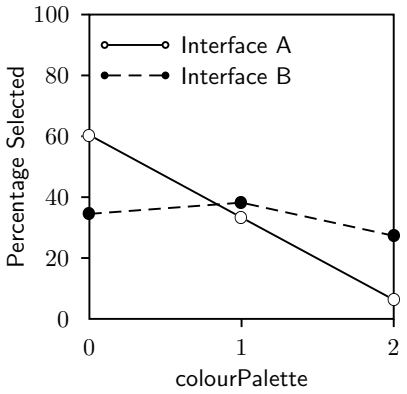


Figure 6.13: Frequencies of colourPalette element choices.

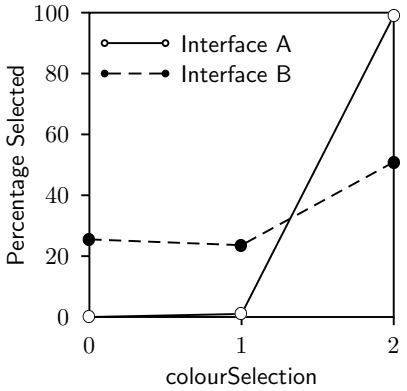


Figure 6.14: Frequencies of colourSelect element choices.

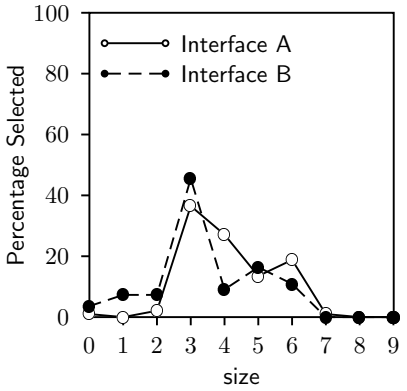


Figure 6.15: Frequencies of size element choices.

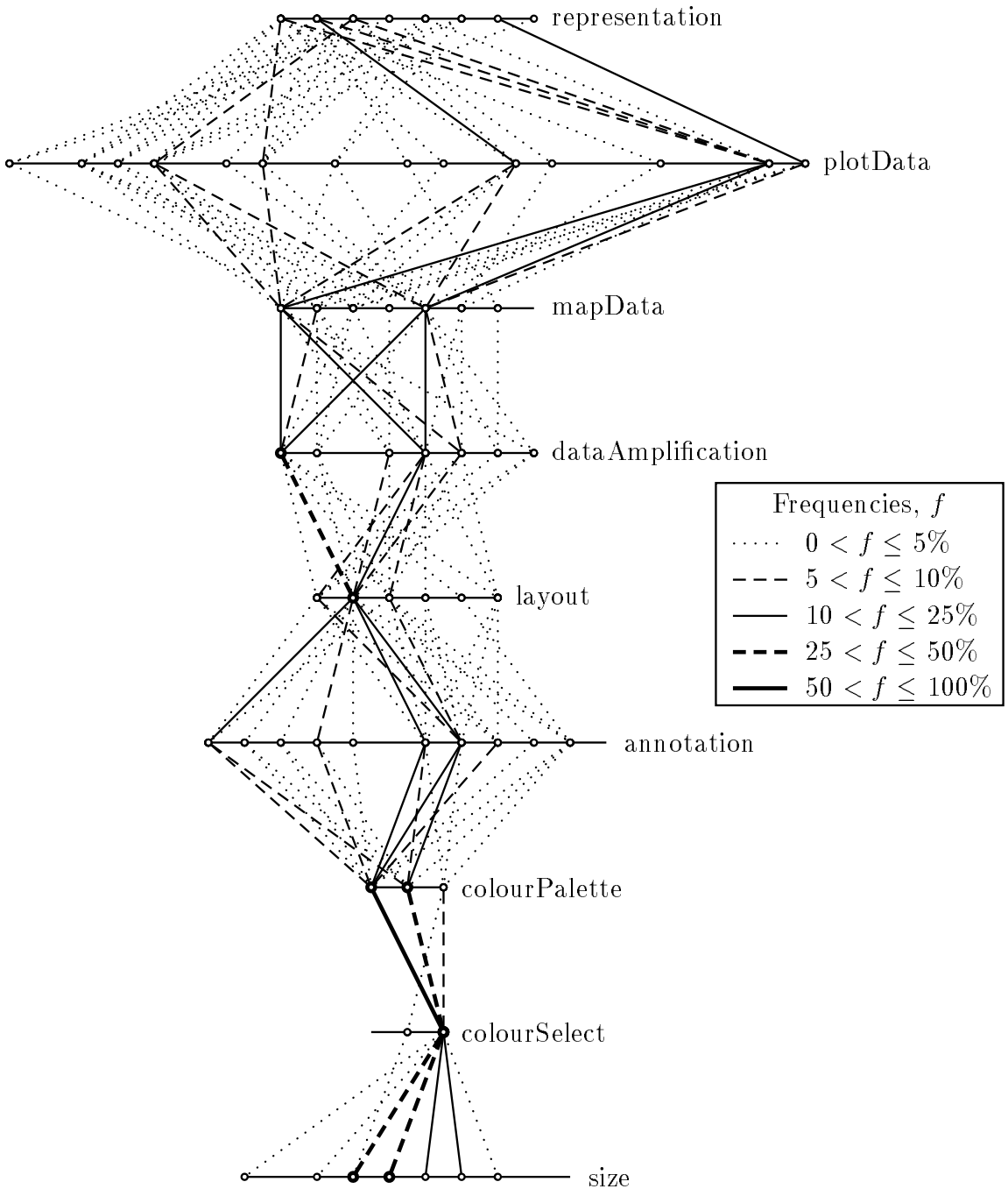


Figure 6.16: Frequency of selections for interface A. The lines connect pairs of elements chosen from adjacent components. The style in which the line is drawn indicates the frequency that the pair was selected.

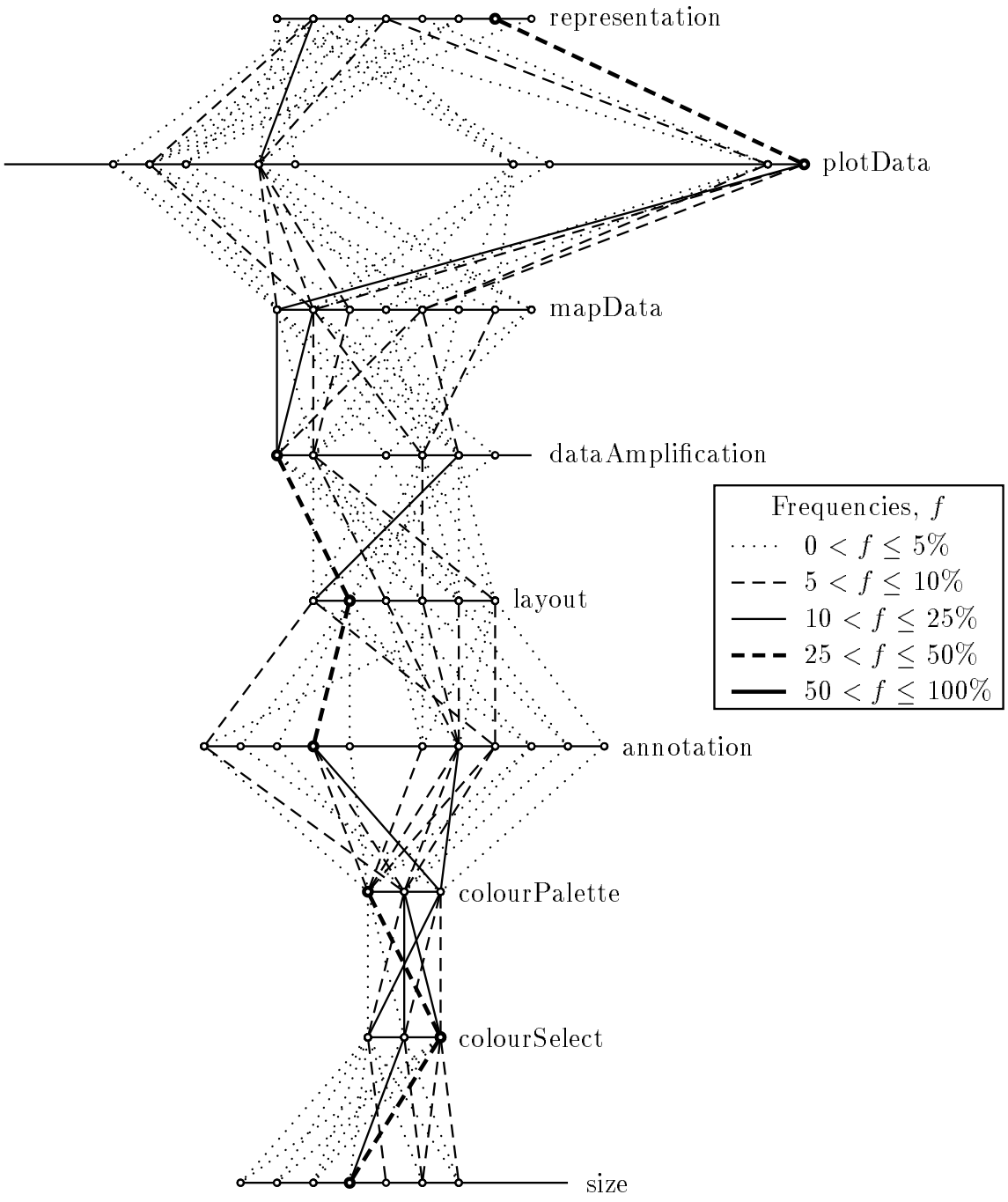


Figure 6.17: Frequency of selections for interface B. The lines connect pairs of elements chosen from adjacent components. The style in which the line is drawn indicates the frequency that the pair was selected.

the software, subjects were allowed to answer the questions in whichever way they saw fit. The actual length of the session was affected by the time needed to complete the training task, which varied between subjects and between groups. Interface B did have more features which the users needed to practice, and it was also more inviting for "play."

Despite similar backgrounds, there was considerable variability in the way that people approached the questions. For some, the task seemed to be very difficult. This underscored the idea that task knowledge is necessary. Both interfaces to the system make alternative visual representations available, however they do not provide the user with any suggestions about which representation to use. It might be a useful feature to begin the exploration at a reasonable position in the search space.

It was thought that the selection of trilinear plot would be encouraged by Interface B. Although the data suggests this, the reasons for subjects making that selection are not always clear. For many subjects, it seemed that this particular representation was chosen if it was already known to them. Many more people considered the trilinear plot but did not select it, because of its unfamiliarity. This may provide a more substantial response to Wainer's concern about the proliferation of "bad graphics" [257], as it would seem that almost no amount of coaxing will make people select a more efficient representation. This study was not designed to explore this question of when people might choose to adopt a representation which is novel to them. As Casner [38] wrote, an efficient representation may come at the price of learning time, and some people are not willing nor able to make this commitment. The continued acceptance of the QWERTY keyboard over the more efficient Dvorac keyboard layout is a case in point [224]. Similarly, there were comments which requested pie charts and other representations, even though the more efficient [1] Nightingale rose was provided.

The distribution of selection frequencies for elements in the components indicates that users of interface B tried more varied alternatives. The exception is the plotData component, which was sampled more thoroughly through Interface A. The reason for this may be that it is more difficult to choose between datasets with no obvious visual cues through Interface B.

Subjects with higher scores tended to be older. This may be because the main task was better suited to more experienced people. Most indicative of a high test score was

[1] the structure of the Nightingale rose can make it easier to compare proportions on the same graph and between different graphs

the likelihood of using a commercial software package to produce graphs (Q12). There was a very strong relationship between that and choosing to use commercial software to understand data. This indicates that these two purposes are seen to be distinct; either interface was always at least as likely to be used for understanding data.

Interface B users were the only ones to rate the number of alternatives as "Small". Likely this is due to the fact that these users only saw as much as of the space that they needed to complete the task, without getting a sense of what was available. One comment suggested that this information be added to the interface. It would be interesting to observe the effect of this change on how people work with the software.

Interface B has the steeper learning curve. Perhaps for this reason, subjects in group N were not able to use all the tools effectively in order to complete the task. It is interesting to note that although the subjects in group N rated the software positively for access to alternatives (Q6), they rated it as unhelpful for exploring (Q8). Both interfaces received generally high marks for their support of exploration, so an explanation for the low marks from group N may be related to the subjects' ideas about exploration. Unfortunately, no specific data was collected which might shed light on this. It would be interesting to probe further.

Satisfaction was surprisingly low: no one rated themselves as "Very Satisfied." Interestingly, all those who rated themselves as "Very Unsatisfied" belonged to Group N. For those in Group H, there was a slight trend for higher satisfaction with Interface B.

In general, the relationships between responses regarding the development of understanding, graphic effectiveness, software helpfulness for exploring, and software help for accessing alternatives are expected. From several user comments, an integration of the two styles of interaction will lead to a more effective tool, giving people a choice of how they would rather work. It would be worthwhile to see if this provision would allow a still-higher success rate using the interface, and if longer or more detailed training would be beneficial. Because there are also many comments which are application-specific, a more detailed investigation aimed at formulating better abstractions for application tasks would be worthwhile. One expects there to be variation between users in which abstraction is most appropriate, and so an investigation about effective ways for a user to tailor the abstraction can also be important.

The assessment test was a success in that it has provided confirmation of the design principles and theories. There are many open issues which will require further work. The

test concentrated on a single user at a single session. However, it is suggested that these ideas are applicable to use by a single user over an extended period and by several users who wish to collaborate. Support for these additional modes of use will be the subject of future implementation work.

Chapter 7

Conclusions

Computer-aided visualization is a new term in the lexicon of computational scientists, though its roots are clear. The term has been used purposefully throughout this dissertation to emphasize that visualization is a creative, human endeavour to which the computer can contribute substantially.

When a design does not balance the roles of human and computer, the resulting software is skewed to one of the extremes of manual, requiring the user to make all the decisions, or automatic, putting all decisions under the control of the software. Although designs from both extremes have been successful, the balance they lack between human and computer means that certain aspects better suited to one must be handled by the other.

Manual systems are designed for the user who is both an expert in programming and in the application task. This user appreciates the control which he or she can exert in creating images. Automatic systems are designed for users with problems which are deemed either too trivial to warrant personal involvement, or too complex to benefit from personal involvement.

There are a number of systems which have balanced the roles of human and computer in different ways. The thesis of this dissertation concerns the criteria for establishing an appropriate balance which allows the user to explore, evaluate, and be involved in image creation without becoming overwhelmed.

The remainder of this chapter presents the specific contributions of this work in relation to the challenges of computer-aided visualization and some directions for future work.

7.1 Specific Contributions

The research described in this dissertation has focused on the development of a new paradigm for the use of computers in support of visualization. From the articulation of a new paradigm to the implementation of the *cogito* system, this work has touched on many theoretical and practical areas, which are indicated below:

a new perspective on computer-aided visualization

The field of scientific visualization has been dominated by a data-centric view [145]. While such work has been very important, visualization also properly has a user-centered aspect which has not received the same attention. Scientific visualization is, first and foremost, a creative human endeavour. The goal that each individual should find effective imagery for his or her problem naturally implies the means to explore, evaluate, and to be involved with the creation of imagery.

a new relationship between human and computer

The design and development of the *cogito* system began with the insistence that the computer can do more than simply produce images. Foley and Ribarsky [81] looked only to intelligent systems for an answer to help scientists deal with the complexity. The alternative envisioned here has made the user responsible for the visual representations chosen and has given the computer the job of supporting the structures required for problem-solving. This reflects "interactive" computer graphics.

support for the individuality of each user

It is one thing to ask a person to make a visual representation of some data. It is a different, more difficult thing to ask a person to make a visual representation of some data and then use it to gain a deeper understanding of the data. That person may not be able to express his or her intentions for creating new visual representations, nor understand what is possible. A system deals with these issues can more fully engage users in the discovery process. This dissertation has argued, in part based on some initial empirical evidence, that support for variability in representations is important. Regardless, such an issue cannot be decided until a system which provides this feature is built and extensively used. If there are situations where a single effective visual representation exists for everyone concerned, the users may still benefit from searching for, and finding, that visual representation for themselves.

design principles for computer-aided visualization

an examination of problem-solving, semiotics (by way of cartography), and human-computer interaction has revealed certain design principles which were used in the implementation of *cogito*.

implementation of a computer-aided visualization tool

an implementation of those design principles has resulted in a system with a general framework for managing the components and elements of particular applications, two very usable interfaces to support local and global searching, and an extensible Application Programming Interface (API) for dealing with new applications. Interestingly, an easy way to develop new applications for *cogito* is to assemble parts from other applications.

empirical study of a computer-aided visualization tool

the results of a preliminary study have confirmed the design principles and their implementation in *cogito*. The study has also contributed to the knowledge about how people perform visualization tasks.

a simplified modular architecture

cogito provides to the user a simpler view of the programming problem than those environments based on a traditional visual programming approach. By hiding from user the specific inputs and outputs of modules, the user can concentrate more fully on dealing with the requirements of the problem.

a new relationship between user and programmer

the *cogito* system places the user in direct contact with visual representations and releases the programmer from the more traditional intermediary between user and computer. The programmer, working with the user, is essential in articulating and enriching the space which the user explores with *cogito*. Over time, the artifact of their work may evolve considerably.

7.2 Future work

However encouraging, the user study described herein is only a preliminary effort. The software does represent an important environment for the testing and development of new

theories.

Short-duration tests, like the one described, will continue to be useful in assessing how well future designs support peoples' efforts to work with images.

Long-duration tests with real problems will be useful to determine how well theories extend as explorations move into more uncharted territory and to see how well the architecture scales with larger problems and larger datasets. One may be able to use this experience to develop more detailed ideas about about how people use which features and when.

Although there are many features to make programming possible, they are not as yet well supported and it would be very interesting to see what sorts of tools might emerge.

This work has focused on visualization of already-collected data. It would be interesting to explore the use of this system as a means of managing numerical experiments by exploring interesting parameter spaces for data computation.

This work has relied on the user to drive the selection of visual representations. However, it might well be appropriate to offer the user some assistance in evaluating particular representations, perhaps through a critic [79, 137].

The implementation discussed in this dissertation represents only the core of the theory which has been outlined. A more complete system would include better support for navigation and collaborative work.

Appendix A

User study materials

This appendix contains all materials involved in the experiment entitled "Evaluation of an Interactive Computer System for Data Visualization" and conducted between May 26 and June 7, 1999. An electronic version of each page of the study materials has been created and is presented in the following order:

- ethics approval
- subject recruitment e-mail message
- information, informed consent, and feedback forms for subjects
- pre-task questionnaire
- introduction to the study and tutorials for each interface
- training task and main task instructions
- data file for the main task
- post-task questionnaire

SIMON FRASER UNIVERSITY

OFFICE OF VICE-PRESIDENT, RESEARCH

BURNABY, BRITISH COLUMBIA
CANADA V5A 1S6
Telephone: (604) 291-4370
FAX: (604) 291-4860

April 19, 1999

Mr. Daryl H. Hepting
Graduate Student
School of Computing Science
Simon Fraser University

Dear Mr. Hepting:

Re: Evaluation of an Interactive Computer System for Data Visualization

I am pleased to inform you that the above referenced Request for Ethical Approval of Research has been approved on behalf of the University Research Ethics Review Committee. This approval is in effect for twenty-four months from the above date. Any changes in the procedures affecting interaction with human subjects should be reported to the University Research Ethics Review Committee. Significant changes will require the submission of a revised Request for Ethical Approval of Research. This approval is in effect only while you are a registered SFU student.

Best wishes for success in this research.

Sincerely,

Dr. Adam O. Horvath, Chair
University Research Ethics Review Committee

c: R.D. Russell, Supervisor

/bjr

Figure A.1: Ethics approval letter

Hello:

I am conducting a user study to evaluate the effectiveness of a prototype software system and its interface. I am seeking computer-literate persons 18 years of age or older to be subjects in this study.
The software being tested is designed to augment a person's ability to work with images in a problem-solving context. Any particular image represents a collection of choices amongst many possibilities. The number of all possible choices may be immense, and the prototype software system seeks to provide effective access to these possibilities. I am interested in determining how well this access is provided.
The study has four parts:

1. you will be asked to complete a short questionnaire which describes your background
2. you will be trained to use a prototype software system with a sample problem
3. using that same software system, you will be given a set of data and asked to graphically examine the data and answer questions about it
4. after you have completed the task, you will be asked to complete another short questionnaire in which you will be asked to evaluate the software system and communicate about your experience using the software

An audio recording of your voice and a video recording of the computer screen only will be made during the entire session. Additionally, your interactions with the software will be logged. There are no extraordinary risks for you if you choose to participate in the study. However, please be aware of the following:

- you will be seated at a desk in front of a computer during the study session
- you will interact with the software system by intermittently positioning and clicking a mouse
- you may at first feel nervous about having your voice recorded, though experience shows that this feeling will not persist throughout the entire study.

You will be free to discontinue your participation at any time. The whole time of your involvement with the study will be approximately one (1) hour. The study will take place in Applied Science Building Room 9836 (part of the Graphics and Multimedia Research Lab).
This research is intended to evaluate a novel approach for the use of the computer in presenting and providing access to alternative visual representations. Your participation in this study will be an important contribution to the furtherance of this research.
If you are a student in the CMPT361 taught by me at SFU during the summer semester, understand that your decision of whether or not to participate will have no bearing whatsoever on any aspect of your work in that course.
If you are interested in participating in this study or have questions before deciding whether to participate, please send e-mail to me at dhepting@cs.sfu.ca.

Best regards,
Daryl Hepting

Table A.1: Full text of e-mail message sent to recruit subjects for the study.

SIMON FRASER UNIVERSITY

INFORMATION SHEET FOR SUBJECTS PARTICIPATING IN EVALUATION OF AN INTERACTIVE COMPUTER SYSTEM FOR DATA VISUALIZATION

The following is a description of the procedures to be followed and a statement of the risks and benefits of this research project.

The software you are evaluating is designed to augment a person's ability to work with images in a problem-solving context. Any particular image represents a collection of choices amongst many possibilities. The number of all possible choices may be immense, and the prototype software system seeks to provide effective access to these possibilities. This study will help to determine how well this access is provided.

The study has four parts:

1) you will be asked to complete a short questionnaire which describes your background.
2) you will be trained to use a prototype software system with a sample problem.
3) using that same software system, you will be given a set of data and asked to graphically examine the data and answer questions about it.
4) after you have completed the task, you will be asked to complete another short questionnaire in which you will be asked to evaluate the software system and communicate about your experience using the software.

An audio recording of your voice and a video recording of the computer screen only will be made during the entire session. Additionally, your interactions with the software will be logged. There are no extraordinary risks for you if you choose to participate in the study. However, please be aware of the following: you will be seated at a desk in front of a computer during the study session; you will interact with the software system by intermittently positioning and clicking a mouse; and you may at first feel nervous about having your voice recorded, though experience shows that this feeling will not persist throughout the entire study period. You will be free to discontinue your participation at any time during the study. The whole period of your involvement with the study will be approximately one (1) hour. The study will take place in Applied Science Building Room 9836.

This research is intended to evaluate a novel approach for the use of the computer in presenting and providing access to alternative visual representations. Your participation in this study will be an important contribution to the furtherance of this research.

Figure A.2: Information for subjects.

SIMON FRASER UNIVERSITY

INFORMED CONSENT BY SUBJECTS TO PARTICIPATE IN EVALUATION OF AN INTERACTIVE COMPUTER SYSTEM FOR DATA VISUALIZATION

The University and those conducting this project subscribe to the ethical conduct of research and to the protection at all times of the interests, comfort, and safety of subjects. This form and the information it contains are given to you for your own protection and full understanding of the procedures. Your signature on this form will signify that you have received a document which describes the procedures, possible risks, and benefits of this research project, that you have received an adequate opportunity to consider the information in the document, and that you voluntarily agree to participate in the project.

Knowledge of your identity is not required. You will not be required to write your name or any other identifying information on the research questionnaires. An audio recording of your voice and a video recording of the computer screen only will be made during the session. The video and audio recordings of the session will be reviewed only by the Principal Investigator. All research materials will be held confidential by the Principal Investigator and kept in a secure location. These research materials will be destroyed after the completion of the study.

Having been asked by Daryl H. Hepting of the School of Computing Science of Simon Fraser University to participate in a research project study, I have read the procedures specified in the accompanying information sheet. I understand the procedures to be used in this study and the personal risks and benefits to me in taking part. I understand that I may withdraw my participation in this study at any time.

I understand that my decision to participate in this study, and my subsequent involvement in it, will have absolutely no bearing on any other dealings I have with Mr. Hepting. This includes, but is not limited to, the case that I am a student in the CMPT 361 course taught by Mr. Hepting, offered at SFU during the 99-2 semester.

I understand that I may register any complaint I might have about the study with the researcher named above or with Dr. Jim Delgrande, Director, School of Computing Science of Simon Fraser University, Burnaby, BC, V5A 1S6, telephone 604-291-4277.

I may obtain copies of the results of this study, upon its completion, by contacting Mr. Daryl Hepting, in care of the School of Computing Science at Simon Fraser University.

I understand that my supervisor or employer may require me to obtain his or her permission prior to my participation in a study such as this.

I agree to participate by completing: a pre-task questionnaire; a training session on the prototype software system; a task with the prototype software system; and a post-task questionnaire. I understand that these activities will require approximately one hour at a time scheduled with Mr. Hepting. I understand that the experiment will be conducted in Room 9836 in the Applied Science Building of Simon Fraser University.

NAME (please type or print legibly): _____

ADDRESS: _____

_____

SIGNATURE: _____

WITNESS: _____

DATE: _____

A COPY OF THIS SIGNED **CONSENT FORM** AND A **SUBJECT FEEDBACK FORM** WILL BE PROVIDED TO YOU AT YOUR EXPERIMENT SESSION.

Figure A.3: Informed consent form.

SIMON FRASER UNIVERSITY
UNIVERSITY RESEARCH ETHICS REVIEW COMMITTEE

SUBJECT FEEDBACK FORM

Completion of this form is **OPTIONAL**, and is not a requirement of participation in the project. However, if you have served as a subject in a project and would care to comment on the procedures involved, you may complete the following form and send it to the Chair, University Research Ethics Review Committee. All information received will be treated in a strictly confidential manner.

Name of Principal Investigator: Daryl H. Hepting

Title of Project: Evaluation of an Interactive Computer System for Data Visualization

Dept./School/Faculty: School of Computing Science

Did you sign an Informed Consent Form before participating in the project? ____

Were there significant deviations from the originally stated procedures? _____

I wish to comment on my involvement in the above project which took place:

_____
(Date) (Place) (Time)

Comments: _____

_____

_____

_____

_____

_____

Completion of this section is optional

Your name: _____

Address: _____

Telephone: (w)_____ (h)_____

This form should be sent to the Chair, University Research Ethics Review Committee, c/o Office of the Vice-President, Research, Simon Fraser University, Burnaby, BC, V5A 1S6.

Figure A.4: Subject feedback form.

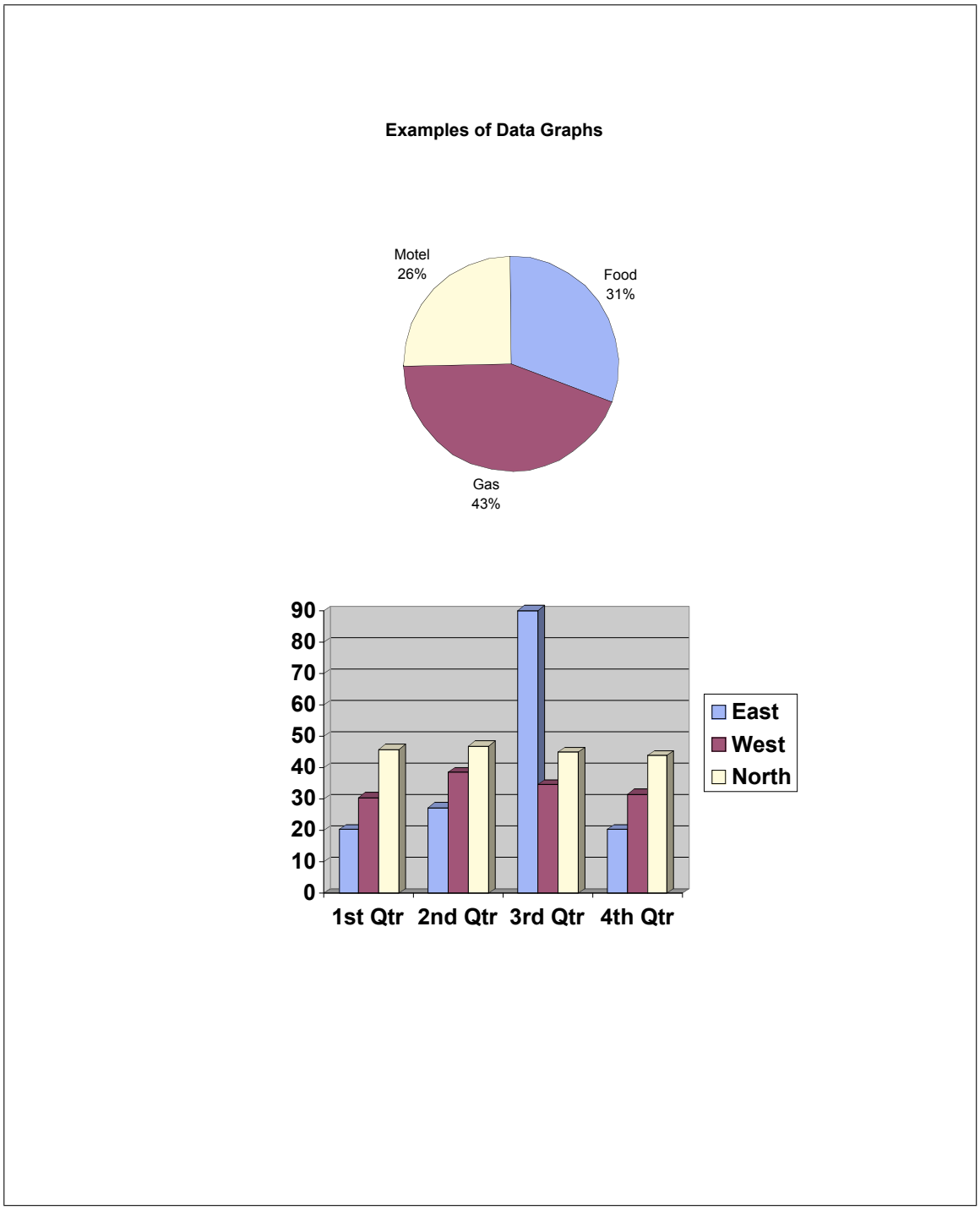


Figure A.5: Sample graphs given for reference while completing pre-task questionnaire.

PRE-TASK QUESTIONNAIRE

The following questions relate to your background and experience using computers and graphs. Please answer each question by circling the most appropriate response immediately following each question. Your answers to these questions will allow for more accurate analysis of the data collected during the study. The task you will perform today relates to your use of graphs which may be like the examples on the previous page. When a question refers to graphs, please consider those examples when responding.

1. What is your gender?

Male **Female**

2. To which age category do you belong?

18 - 25 **26 - 35** **36 - 49** **50+**

3. How many years education have you completed past high school?

0 **1 - 4** **5 - 6** **more than 6**

4. Which of the following general categories best describes your main area of interest?

Fine Arts **Business** **Arts/Humanities** **Science/Applied Science**

5. How often do you read graphs (i.e. look at graphs to gain information from them)?

Daily **Weekly** **Monthly** **Less than monthly**

6. How often do you create graphs?

Daily **Weekly** **Monthly** **Less than monthly**

7. What is your preferred method to create graphs?

Pencil and paper **Computer software** **Other (please specify)** _____

8. When both text and graphs present an interpretation of data, which do you prefer to use?

Text **Graphs** **Both Text and Graphs**

9. What is your level of experience producing graphs with software?

Low **Medium** **High**

10. What is your level of experience using software which produces graphs as part of some other application?

Low **Medium** **High**

11. What is your level of familiarity with computer graphics?

Low **Medium** **High**

Figure A.6: Pre-task questionnaire.

Evaluation of an Interactive Computer System for Data Visualization

Introduction:

Any picture you see is the result of several choices. Sometimes these choices are more explicit and available than others. The following image has parts, which we will label as shape, colour, squeeze, orientation, lightModel, lightSource, lightPosition, and lightDirection. In the software you will be using, these parts are called ***components***. Each component may have several representative pieces, called ***elements***. The components and elements that are presented to you provide a way to think about the problem at hand. Here, cyan is an element of the component colour; cone is an element of the component shape; and so on. A choice of element from each component makes an image. There are some restrictions, because some combinations don't make any sense.

You may be familiar with some computer-graphical applications that present you with a finished image with little or no choice about its composition. The software you will be using today gives you more explicit control of how an image looks, by allowing you to choose and examine the interaction between different elements of different components.

To train you in the use of this prototype software system, you will explore other possible images like this sample cone.

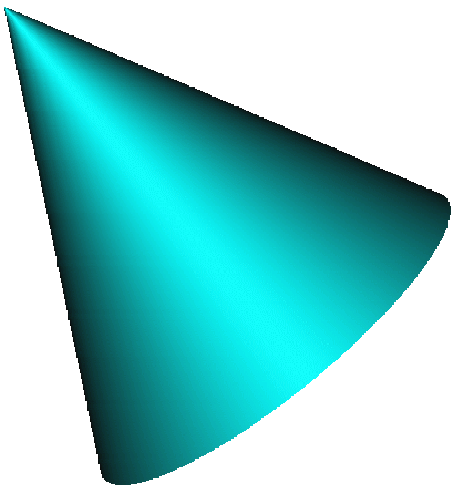
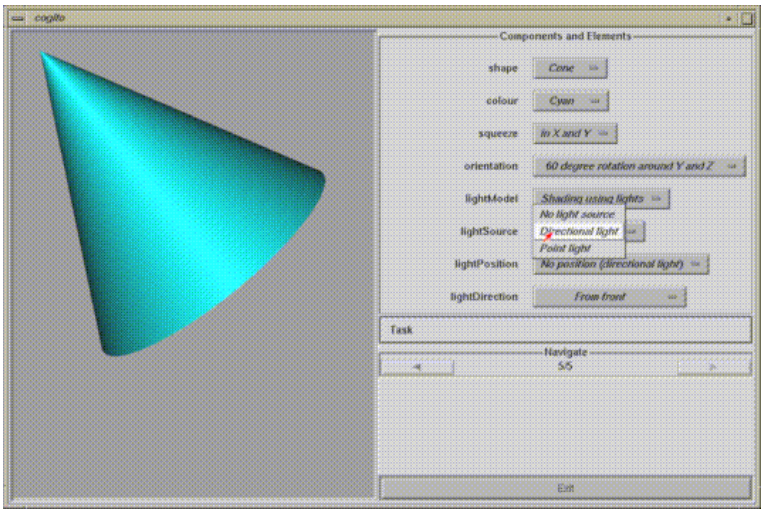


Figure A.7: Introduction to study.

Reference:



On the left, the interface has a viewing area in which the current image is displayed. You may zoom into the image by left-clicking and dragging the mouse. In this state, the normal arrow cursor will change to the zoom cursor, depicted below. Moving the mouse towards the bottom of the screen will magnify the image and moving the mouse towards the top of the screen will demagnify the image. You may pan the image (move the image around inside the window) by middle-clicking and dragging. In this state, the normal arrow cursor will change to the pan cursor, depicted below.


Zoom cursor


Pan cursor

On the right, the interface provides the means for you to control the image displayed on the left. At the top, there are controls to access the available components and elements. At any moment, the controls reflect what is shown in the current image at the left. You can choose a new element for a component by moving your mouse over the button that shows the current element. For example, to change the shape from Cone to Sphere, move the mouse over "Cone" and press and hold the left mouse button to reveal a menu of available shapes. Drag the mouse down the menu until "Sphere" is highlighted and release the mouse. You can alter other components in the same way. Shown above is the menu of available elements for the lightSource component. If an element is greyed-out in the menu, it is not compatible with the other current selections.

Figure A.8: Tutorial for interface A, page 1.

The components are organized from top to bottom. You will always be able to select any element from the first component (shape). However, as you go down the list, items will be greyed-out if they do not fit with your earlier selections. If an element you'd like to select is greyed-out in this way, you must change the element from an earlier (higher in the list) component. There is at least one valid combination of elements for every element in the first component (shape). If any single change of element you make would result in an invalid combination, the system will adjust the elements in the other components to always show a valid combination.

Immediately below the "Components and Elements" area is the task button. Use the task button to indicate when you attempt or answer a question with the current image. Left-click on the button now and you will see a menu with 2 items which correspond to the training questions below. For each image that is used to attempt or answer a question, click and hold the task button to display the menu then drag the mouse down the menu until the appropriate question is highlighted, then release. If you find that you use the same image for more than one question, please check all that apply.

Below the task button is the navigation area, which indicates your position amongst the images you have created (new images are always added to the end of the list) and allows you to move backward and forward amongst those images. If no more movement is possible in a direction, the corresponding arrow will become greyed-out.

At the bottom right of the screen is the exit button, which you can press once you have completed the task.

Tutorial:

For the components, make sure that the following elements are set:

- shape Cone
- colour Cyan
- squeeze In Y
- orientation 45 degree rotation around X, Y and Z
- lightModel Shading using lights
- lightSource Point light
- lightPosition At far top

Notice that the only element available for selection for the lightDirection component is “No direction (point light)”, which indicates that because a point light source has been chosen, no direction is needed for that light source. At this point, the position indicator in the navigation area will read “7/7” [*please note: the actual numbers displayed at this point in your tutorial session may be different, depending on how many components you had to set to get to this point -- the numbers here are only used as an example*]. Click on the left navigation arrow twice (2 times). The position indicator will now read “5/7” to indicate that you are looking at image number 5 out of a total of 7 images.

Figure A.9: Tutorial for interface A, page 2.

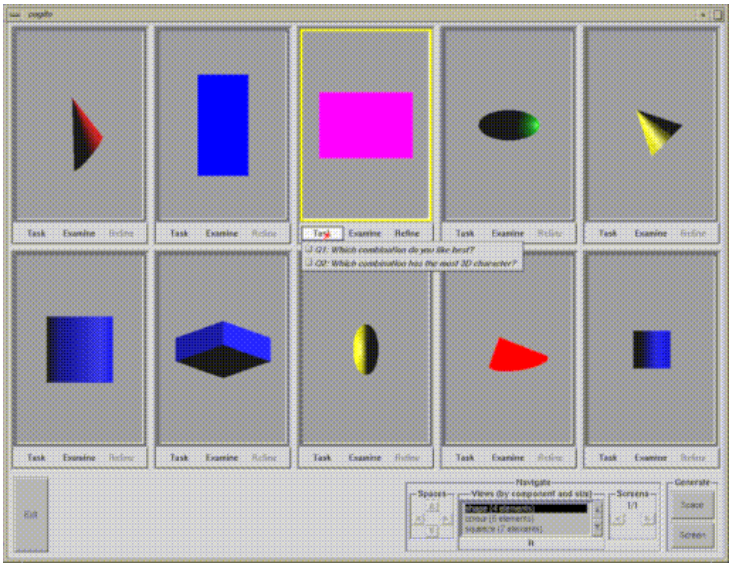
Change the shape from Cone to Cube. Notice that the position indicator now reads “9/9”. The new image has been placed at position 9 of 9 (the end of the list) and immediately before in it in position 8 is a copy of image 5. Since image 5 was used as the basis for the new image, the copy preserves this connection.

Zoom and pan the image so that you have a close-up view of a corner of the cube. Now change the colour to Yellow. Notice that the view operations are not permanent, which can be verified by navigating back one image to the cyan cube.

Click on the task button to show the task menu. Select question 2. You can verify that it has been selected by clicking again on the task button, where you’ll see a red check mark in the box in front of the question.

Figure A.10: Tutorial for interface A, page 3.

Reference:



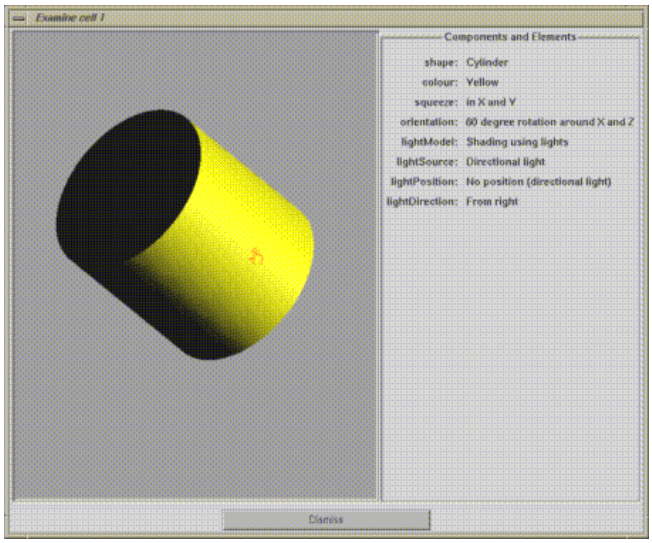
In the upper portion of the screen, you see an array of 10 small images, each of which shows one of the available combinations of elements that form valid images. The system only displays those combinations that make sense. All the available combinations form a ***space*** from which representative images are shown. Alternative images are presented to you through an interactive display, for which you have the power to reorganize (with ***views***) and restrict contents. If you see an image that you think may be useful for your task, you can select it by moving the mouse over the image and clicking the left mouse button. When an image is selected, its border is set to yellow. One or more of these selections can be used to generate a new space whose contents are consistent with the selections. For example, if you select the only the magenta cube (as shown above), you can generate a new space which contains only that cube. If you make more than one selection, the elements from each of the selections will be combined to produce different alternatives, which are similar to and consistent with your selections.

The controls to configure and use the display are presented below the images. At the far right are the “Generate” controls. To see more images in the current space with the current view, press the “Screen” button in the “Generate” area. A new set of images will appear if there are more valid combinations in the current space. Once you have made one or more selections from the current space (possibly from several different views), click on the “Space” button inside this area to create a new space based on your selections. To the left of the “Generate” area are the navigation controls. Images from a space are organized and displayed screen by screen, according to some view. The view is the

Figure A.11: Tutorial for interface B, page 1.

component by which sample images are organized. If the current view is shape, then you will see an example of each shape, one after another. The “Screens” navigation controls allow you to move between already-generated screens from the current view of the current space. Selecting a new view from the “Views” list can change the organization of the displayed space. You can use the arrows at the right of the list to scroll up and down to see all possible choices. If the colour component is chosen for the view, representative examples of each of the six available colours will be shown in sequence. With 10 images in a screen, almost 2 complete cycles could be shown on a single screen. The elements from the other components (like orientation) are selected at random. To the right of this list are the controls for “Space”. You can use these arrows to return to previously explored spaces, from which you can make new selections. Over time, a tree structure may emerge.

Directly below each image are buttons labelled “Task”, “Examine” and “Refine.” Use the task button to indicate when you attempt or answer a question with this image. Left-click on the button now and you will see a menu with 2 items which correspond to the training questions below. For each image you use, click and hold the task button to display the menu then drag the mouse down the menu until the appropriate question is highlighted, then release. If you find that you use the same image for more than one question, please check all that apply.



Since the images on the screen are small, you may click on the “Examine” button to bring up a viewer to get a closer look at the image. To the right, you see a list of the elements used from each component. On the left is a viewing area in which the current image is

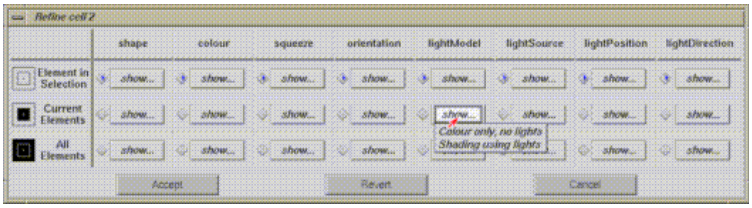
Figure A.12: Tutorial for interface B, page 2.

displayed. Once inside this viewing area, the normal arrow cursor will change to the zoom cursor, depicted below. You may zoom into the image by left-clicking and dragging the mouse. Moving the mouse towards the bottom of the screen will magnify the image and moving the mouse towards the top of the screen will demagnify the image. You may pan the image (move the image around inside the window) by middle-clicking and dragging. In this state, the zoom cursor will change to the pan cursor, depicted below.

Zoom cursor

Pan cursor

The system is configured so that selections will shrink the space. Selecting the magenta cube (above) and generating a space from that selection will lead to a space with only that single combination in it. When an image is selected, the associated “Refine” button becomes active, so that you may refine your selection (in terms of how it contributes to the next space when it is generated). If you get to the point where you have a space containing just a single combination, the “Refine” dialogue box allows you to expand the space in certain components, to see for example how other colours look on the cube with that orientation and lighting.



For each component, you can choose whether to use the single element that is displayed in your selection, use the elements available in the current space, or use all elements (which were available from the start). The default is to use the “Element in Selection” for each component (shown by blue marks in the diamonds). To change a setting, click on the diamond associated with that setting. Only one setting per component is permitted. The icons on the left of the dialogue box provide a cue as to the relative size of the space, which will result from the respective settings. To see which elements are available for each choice of refinement, click on the appropriate “show…” button. The list that pops up is only for reference and you can’t select from it.

At the bottom left of the main screen is the exit button, which you can press once you have completed the task.

Figure A.13: Tutorial for interface B, page 3.

Tutorial:

This interactive display is a bit like a catalogue, but more powerful. You can change the items you see on the screen by a click of the mouse. As the system begins, it starts with the view organized by shape. You see an example of each of the 4 shapes, repeated in sequence. Notice that this line is selected in the "Views" list of the "Navigate" area. The numbers, which follow the names of components in this list, indicate the number of elements in the respective components. Because not all combinations are compatible, you may not see all the elements in combinations in the current space.

Click on "colour" in the "Views" list. Notice that you see samples of Green, Blue, Cyan, Magenta, Yellow, Red, Green, Blue, Cyan and Magenta on the first screen. To see more samples, click on the "Screen" button in the "Generate" area. Click 2 times to generate 2 screens. Notice how the position indicator in the "Screens" portion of the "Navigate" area changes. Instead of looking at screen after screen of samples, this system is designed to allow you to search the space of available images by making selections to narrow (and widen) spaces as appropriate to arrive at the image you need for your task. From within these 3 screens, pick 2 samples that you like, one using the colour red and the other using the colour cyan. Notice that the borders of these images turn yellow to indicate that they have been selected.

Now generate a space based on these selections. In the new (child) space, the selections you made in the old (parent) space are shown first with blue borders and the controls for these images are inactive. The parent selections appear again in this child space, so they are available for further selection. Scroll through the "Views" list and notice that all the components now have either 1 or 2 elements associated with them. Also notice that each combination in this space is a combination of elements from the two parent selections. There are only about 4 screens of samples for this space (depending on which two samples you selected). Go ahead and generate all the screens for this view and space now.

Find a sample that you like. Click on its "Examine" button (you don't need to select the sample to examine it). In the examiner viewing area, you can use the zoom and pan controls to manipulate your view of the image. Any changes you make are transient, in that they won't persist after the current display is changed. You can look at multiple cells in the display at once (so you can have several dialogue boxes open). You must, however, dismiss all of them before you interact with the main window again (otherwise you will hear a beep). Before you make any selections here, notice that the generate space button is greyed-out. There must be at least 1 selection made before a new space can be generated. Select the image you just viewed by clicking within its borders. Notice that the "Refine" button became active when you made the selection. This is because the "Refine" dialogue box provides a means for you to refine your selection. By default, only the elements in your selection will be used to form the next space. If you only have 1 selection and you generate a space, the new space will have only 1 combination in it since all components have only 1 element in them.

Figure A.14: Tutorial for interface B, page 4.

Click on the “Refine” button to reveal the associated dialogue box. Here, the components are listed across the top and the possible refinements along the side. The “show…” button at each cell in this grid will show you the element(s) that each refinement will include. Look at the shape component and in that column; press the three “show…” buttons in turn. If you’ve chosen 2 different shapes, you will have access to 1,2 and 4 elements respectively. If you’ve chosen the same shape in both selections, you will have access to 1, 1 and 4 elements respectively. You can’t select specific elements from the lists which “show…” gives you: you can choose to use all of them or none of them. Select “All Elements” for shape by clicking on the diamond in that grid cell. Notice that only one of the 3 refinements can be selected for any component. Click “Accept” to let your changes take effect. Now generate a new space and see the results: 4 shapes.

Use the “Spaces” navigation controls to move up one space (back to the space where you did the last refinement). Notice that the border of your selection in this space is a lighter grey. This is to indicate that that combination was previously selected. Select that same sample or another one that you like and click on the “Refine” button. The goal is to see this combination of shape, colour, squeeze, and orientation under all available lighting conditions. Now select “All Elements” for lightModel, lightSource, lightPosition, and lightDirection. Notice that because components are interrelated in their compatibilities, the refinement for more than 1 component must be changed in order to see the desired change. Click “Accept” to apply your changes. Notice that the refinements you make to this selection are instructions for how to generate the new space and are applied when you generate a new space. If you deselect the sample before you generate a new space, your refinements to that sample will be lost.

Click on the “Task” button to show the task menu. Select question 2. You can verify that it has been selected by clicking again on the task button, where you’ll see a red check mark in the box in front of the question.

Now generate a new space. Notice that the left sibling button of the space navigation controls is now active.

Figure A.15: Tutorial for interface B, page 5.

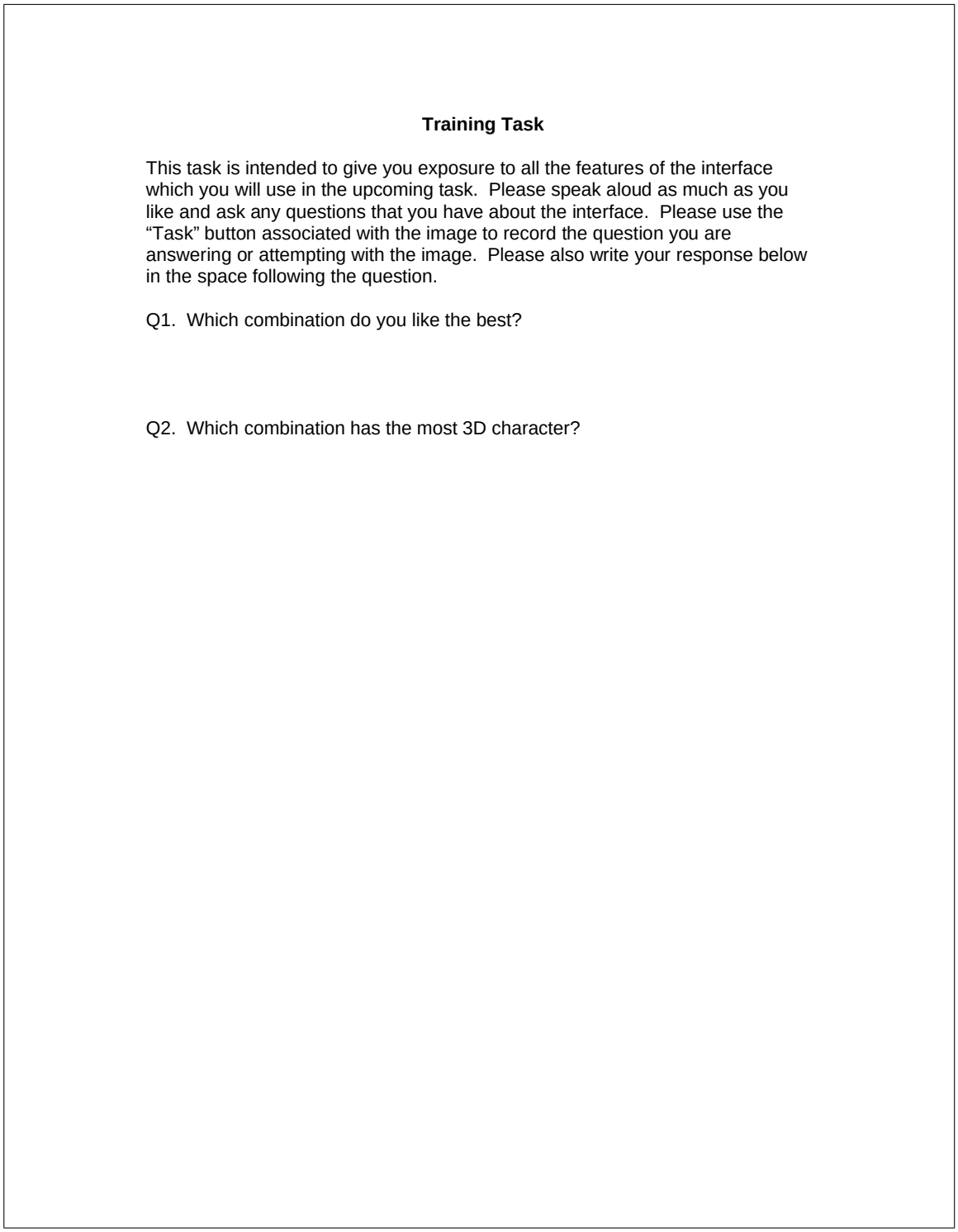


Figure A.16: Training task instructions.

Task

Your task is to answer questions about the following data set with the aid of graphs. There are many ways to graph this data set and some are more effective than others at providing access to the information you need. The software you are using is intended to help you make the choices which will lead to effective graphs for you, to answer the questions. Please write your answers to the questions in the space provided.

Please speak aloud as much as you like and ask any questions that you have about the interface. Please use the "Task" button associated with the image to record the question(s) you are attempting or answering with the graph.

If you see an image which more effectively answers a question than one you've already used, please click on the "Task" button of the new graph and note the question number(s) which it answers.

The data you will be studying is a snapshot of an economy. For each region, there is data for the size of the workforce in each of three sectors of the economy.

Please use only the graphics produced in the software tool you are now using to answer the following questions. You are permitted to answer "No" in response to a question.

Q1. Is there a region whose workforce is evenly distributed amongst the three sectors?

Q2. What is an estimate for the median total workforce of the regions? That is, which workforce has 50% below it and 50% above it?

Q3. Can you characterize the regions in which Sector I predominates?

Q4. Can you describe, in general, the size of Sector I in regions where Sector III predominates?

Q5. Is there a discernable relationship between the 3 sectors?

Q6. Can you find any other relationships in the data?

Figure A.17: Task instructions.

Record	Sector 1	Sector 2	Sector 3	Total	Percent S1	Percent S2	Percent S3
0	67	43	40	150	45	28	27
1	56	71	66	193	29	37	34
2	65	45	57	167	39	27	34
3	15	8	12	35	43	24	33
4	16	8	13	37	44	21	35
5	31	61	122	214	14	29	57
6	48	32	25	105	45	31	24
7	25	53	35	113	22	47	31
8	33	17	14	64	52	26	22
9	28	48	36	112	25	43	32
10	50	20	32	102	49	19	32
11	70	32	29	131	54	24	22
12	42	143	226	412	10	35	55
13	70	55	69	194	36	28	36
14	45	13	20	78	58	16	26
15	65	36	38	140	47	26	27
16	79	39	65	183	43	21	36
17	43	41	36	120	36	34	30
18	64	23	29	116	55	20	25
19	43	41	59	143	30	29	41
20	131	35	62	228	58	15	27
21	58	13	17	88	66	15	19
22	104	34	41	179	58	19	23
23	35	67	39	142	25	47	28
24	46	38	35	119	39	31	30
25	48	52	45	145	33	36	31
26	44	27	38	110	41	25	34
27	164	76	89	329	50	23	27
28	40	51	52	144	28	36	36
29	64	67	84	216	30	31	39

Table A.2: Part 1 of data for the main task

Record	Sector 1	Sector 2	Sector 3	Total	Percent S1	Percent S2	Percent S3
30	63	10	16	89	71	11	18
31	115	107	170	392	30	27	43
32	62	40	71	173	36	23	41
33	137	60	82	279	49	21	30
34	54	30	32	116	46	26	28
35	61	41	55	157	39	26	35
36	68	136	78	282	24	48	28
37	39	34	27	100	39	34	27
38	70	25	28	123	57	20	23
39	51	27	30	108	47	25	28
40	56	160	82	298	19	54	27
41	52	23	32	97	54	24	22
42	101	108	105	314	32	34	34
43	51	51	54	156	32	33	35
44	41	10	16	67	61	15	24
45	70	24	30	124	57	19	24
46	22	5	7	34	64	15	21
47	104	65	65	234	44	28	28
48	116	42	56	214	54	20	26
49	44	57	67	168	26	34	40
50	25	28	28	81	31	35	34
51	74	23	28	125	59	19	22
52	23	127	91	241	9	53	38
53	24	31	27	82	30	38	32
54	132	47	59	238	55	20	25
55	36	173	94	303	12	57	31
56	34	27	33	94	36	29	35
57	81	483	296	860	9	56	35
58	40	69	55	164	24	42	34
59	65	30	35	130	50	23	27

Table A.3: Part 2 of data for the main task

Record	Sector 1	Sector 2	Sector 3	Total	Percent S1	Percent S2	Percent S3
60	94	242	137	473	20	51	29
61	80	79	63	222	36	36	28
62	80	49	62	191	42	25	33
63	37	27	28	92	40	29	31
64	35	20	33	88	40	23	37
65	76	122	114	312	24	39	37
66	40	121	74	235	17	51	32
67	44	215	194	453	10	47	43
68	34	32	23	89	38	36	26
69	94	77	62	233	41	33	26
70	87	45	58	190	46	24	30
71	44	38	35	117	38	32	30
72	52	42	45	139	37	30	33
73	2	575	940	1517	0	38	62
74	8	574	550	1132	1	51	48
75	75	152	174	401	19	38	43
76	37	72	76	185	20	39	41
77	46	328	356	730	6	45	49
78	71	29	33	133	53	22	25
79	57	68	61	186	31	36	33
80	55	47	33	135	41	35	24
81	44	13	18	75	59	17	24
82	33	50	81	164	20	31	49
83	40	30	41	111	36	27	37
84	110	38	40	188	59	20	21
85	60	29	39	128	47	23	30
86	64	47	45	156	41	30	29
87	36	95	43	174	21	54	25
88	41	28	37	106	39	26	35
89	3	25	13	41	8	60	32

Table A.4: Part 3 of data for the main task

POST-TASK QUESTIONNAIRE

The following questions relate to your experience in using the prototype software system to complete the task. The questions which follow are of two types. To answer the first type, circle the most appropriate response immediately following the question. The second type of question is open-ended. Please write whatever you feel is appropriate. The last question is intended as the place for you to make any additional comments not already covered in the questionnaire. Your time and effort in completing the written questions especially is greatly appreciated and all your responses to these questions will be very useful.

1. How do you rate the helpfulness of your experience for performing this task?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

2. How do you describe your feeling about how you completed the task?

Very unsatisfied **Unsatisfied** **Satisfied** **Very satisfied**

3. How do you rate the graphs you chose to complete the task?

Very ineffective **Ineffective** **Effective** **Very effective**

4. How do you rate the number of alternative graphs from which you could choose?

Very small **Small** **Large** **Very large**

5. How do you rate the accessibility of the alternative graphs?

Very poor **Poor** **Good** **Very good**

6. How do you rate the helpfulness of the software for accessing the alternative graphs?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

7. Was exploring alternative graphs helpful to you in completing the task?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

8. How do you rate the helpfulness of the software for exploring the alternative graphs?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

9. Was refining your chosen graphs helpful to you in completing the task?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

10. How do you rate the helpfulness of the software for refining your chosen graphs?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

Figure A.18: Post-task questionnaire, page 1

11. How do you rate the usability of the software?

Very poor **Poor** **Good** **Very good**

12. If a commercial version of this software became available, how likely is it that you would use it to produce your own graphs?

Very unlikely **Unlikely** **Likely** **Very likely**

13. If a commercial version of this software became available, how likely is it that you would use it software to understand your data?

Very unlikely **Unlikely** **Likely** **Very likely**

14. Compared to other means which you've used to create graphs, how do you rate this software?

Much worse **Worse** **Better** **Much better**

15. Was the software helpful to you for developing an understanding of the data?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

16. Was the software helpful to you for developing your own questions about the data?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

17. Was it helpful to be involved in producing the graphs which you used to complete the task?

Very unhelpful **Unhelpful** **Helpful** **Very helpful**

18. Which features did you like the most?

19. Which features did you like the least?

Figure A.19: Post-task questionnaire, page 2

20. Are there features you'd like to see added to the software?

21. Do you have any other comments about the software or the task?

Figure A.20: Post-task questionnaire, page 3

Appendix B

User study comments

User responses to the open-ended questions from the post-task questionnaire are presented by question, separated by interface.

B.1 Which features did you like the most? (Question 18)

B.1.1 Responses from users of Interface A (flat)

- the ability to 'zoom' in on specific parts of a graph.
- rapid change from graph to graph
- colour and organization changeability

- ability to flip between graphs using the arrow keys
- ease of changing minor graph parameters (most packages I've used don't have such a capability).
- center pan button / left zoom

- I liked the different representations and I liked the dataAmplification because I liked to sort the data.

- ease of changing graphs/ ease of selecting elements

- representation, plotData, and layout

- instant feedback, as you changed aspects of the graph, changes were made to the actual graph.

- 3D objects; the light source and shapes create really interesting objects. Many different representations to choose from.

- the size option was quite helpful
- the trilinear graph displayed the info in an easy-to-read and informative way.

- speed of applying changes (so you try a lot of things)
- being able to go back to a previous (maybe thumbnails would be nice) one, so I didn't have to remember the settings.
- sorting by a different thing than I was plotting
- varying representations (scatter vs. column, etc.)

- ability to pan the image [and zoom] in and out;
- parameters were easily accessible and modifications were fast and effective.

- I liked the ability to see all the different representations of the data using the different plots. It is better than being forced to use only one representation to interpret data and answer questions.

- split screen – see results immediately.

- being able to specify colour vs. B&W
- being able to change the graph representations
- sorting

- the ability to sort by percentages versus total amounts
- the ability to choose different graph types and have the result shown on the new graph type.

- zoom
- grid
- different colours for different objects

- having the ability to play with the data, and see it in ways that were meaningful to me – and being able to builld that meaningful representation.

- all the options or types of graphs

B.1.2 Responses from users of Interface B (hierarchical)

- The wide variety of options to refine
- the 'navigate' features, especially 'views' were useful – although I didn't get a chance to explore them fully.
- having [the ability] to choose various 'refine' selections to show graphs
- being able to create several spaces to visualize many graphs in virtual space
- intuitive to use, good for visual people
- point and show
- ability to generate various possibilities that I would not have necessarily have thought to use.
- all the options and being able to limit the possibilities by selecting certain options.
- the ability to navigate between spaces, I think the refine would be better if I had learned to use multiple selections more often.
- hierarchical selection of features of graph
- variety of plot types (some I'd never seen before)
- creating different representations/versions from a graph that was almost what I wanted was useful.
- Also, the 'examine' summary box was useful for giving more information about the underlying data for that particular graph.
- capability to present different perspectives
- flexibility of manipulation of data
- the examine details
- the ability to generate more screens when I didn't really "see" what I wanted exactly.
- great variance in the things one could do with the graph.
- good menu layout.

- it was quite manueverable once you figured out where you were going.

- ability to refine graphs
- ability to return from refinements without losing data.

- I had the power to refine the graphs I wanted

- the way it organises into trees, the hierarchy allows easy access to previous choices.
- the way many graphs are different options displayed at same time lets it be easy to see which one is best for analysing data.

B.2 Which features did you like the least? (Question 19)

B.2.1 Responses from users of Interface A (flat)

- the point and click to zoom and pan graphs was difficult, I often zoomed out when I wanted in.

- unable to see several different graphs at the same time
- slow zoom and pan response

- it took me a while to understand the mapData function, and
- I usually found the large point size just a bit too large.

- hard to see graph
- want to customize elements e.g. select grid size
- only 2D data

- dataAmplification, colourPalette, colourSelect: weren't really used, perhaps they are not important?

- readability, limited customizability of settings for the axes. i.e. more divisions on an axis or grid.

- I don't like how the zoom and pan work. I would prefer it if I could mark a focus point on the graph (a point which remains visible) and then use the left and right mouse buttons to zoom in and out.

- text on the graph is difficult to see sometimes (zooming in/out) - perhaps labels should be displayed (perhaps at the top or bottom of the screen, in separate textboxes) without magnification

- I think that the plotData menu was not laid out in an intuitive fashion (i.e. the placement and order of the options).

- zoom jerky
- call a sort a sort

- graph labels somewhat difficult to read

- I didn't like that when you changed the zoom and pan it didn't stay [between graphs]

- component and element window/buttons need not take up so much space - increase the size of the graph area.

- zooming and panning features
- lack of inline help

- scrolling back and forth is tedious, I preferred to create new graphs rather than scroll back to one already created.
- the large box around the small task button seemed to imply that the whole box was the task button.
- not having the ability to rotate the graph

- auto resize when some factors changed

- having too much there – would be easier with time I think.
- I wasn't sure what mapData was, for instance.

- none

B.2.2 Responses from users of Interface B (hierarchical)

- the inability to reposition and resize the 10 boxes showing the available graphs (i.e. the user needs more flexibility on the interface).
- it was somewhat difficult to view the current attributes of presented graphics. E.g. a quick listing would be helpful, although this is what 'examine' does, perhaps I didn't like the 'examine' interface? Not sure.
- user interface features need to be more user-centered to provide usability.
- moving from one space to another is difficult to picture in mind, i.e. to perceive it.
- begin able to display a list of attributes (elements) of all graphs in a space would be good.
- lack of precise control - let me micro-manage it!
- in generating spaces, I would have liked to be able to EDIT OUT the graphs that I did not like at all, so that going back and forth between the screens and amongst the spaces would help me narrow my search.
- the fact that it was hard to quickly know which elements were present in a certain cell, you have to examine it closely.
- I felt constrained by the refine dialogue when I just wanted to select specific elements that were obviously desirable.
- size of plots and fonts too small to read
- in the refine options, I wnated to be able to select exactly the option to use to generate the next set of graphs, especially when I knew exactly what I wanted to do. Without this option, I may have to generate screen after screen before I get the graph I wanted. Also, I would have liked all the screens to be generated (or at least show me how many pages of screens I have to choose from).
- lots of information

- within the refine the inability to choose exactly what I wanted for a particular attribute. e.g. point size - large
- the zoom feature was too restrictive
- I had a real hard time understanding what it was exactly I had to do. I think more time should have been spent trying to understand what really needed to be accomplished before trying to accomplish it.
- I did not use the feature where I could generate spaces from two or more graphs.
- I found it confusing because I was unfamiliar with the software
- I found it a bit slow when generating the graphs
- it was hard to find the specific graph I wanted from the display of generated graphs. I had to click on each one to get a larger version.
- some features are combined but are quite useless, i.e. bar chart of multiple sectors with single colour - who can distinguish them?

B.3 Are there features you'd like to see added to the software? (Question 20)

B.3.1 Responses from users of Interface A (flat)

- descriptive help options
- I think that a tutorial would be helpful with a number of different applications, to get used to representing many different kinds of data.
- more direct manipulation of graph, i.e. if it was 3D, be able to grab a corner and rotate.
- explanation/help for components/elements.
- more representation types, ex. pie chart and 3D graphical representations.
- For 3D objects, Maple displays a box around the object, which you can manipulate to view the object from anywhere.

- pie charts
- scaling along axes when zooming (i.e. being able to zoom into a graph and still being able to see all the data.)

- more screen real estate for graph (for more zooming)

- increased representation selections (i.e. pie charts, 3D pie charts)
- options to select label scale and choose distance for gridlines.
- larger view frame, perhaps smaller menu for modifying graph components and elements.

- I would like to be able to change the size of the viewing area.

- 3D representations
- more intuitive interface (perhaps a toolbar to make it easier to know how to zoom or pan).
- tool tips
- more room for the graph, less for the buttons and drop-down lists

- ability to add text
- ability to flip graphs/ rotate

- help system or demo
- because of lack of statistic background, hard to understand some charts

- ability to filter out extremes, perhaps. (in data displayed)
- Ability to zoom on a select region of a graph - maybe create a new graph using a subset.

- maybe allow quicker navigation amongst instances of graphs.

B.3.2 Responses from users of Interface B (hierarchical)

- ability to more easily resize 'Examine' view
- to directly change a given attribute to a desired value.

- a navigation screen to show where on the tree I currently am.

- I'd like to be able to specify exactly what properties I would like in the graphs rather than generating spaces and selecting.

- also, I found it really frustrating not being able to see the whole graph with the legend all at once, especially when using 'zoom'.

- also, a good feature to have would be if it told you the number of representations for a given refinement (before you generate the search space).
- a more visual representation of space navigation.

- a more clear labelling of the specific element (being sorted by) in the thumbnail previews

- list main options (for components) in the "screen" display

- better zoom feature

- the ability to view things on a global scale

- perhaps the ability to explicitly set the properties for a graph, rather than just refining with new spaces.

- maybe display some of the elements (like the one that is varied) in the frame around each graph so that you can check the one you want quickly without having to blow up each graph.
- allow user to specify exactly which attribute she wants to change instead of forcing her to search through a generated space of all possible comibinations.

- feature to change specific option, instead of having to generate a space with all the options. It [could] also allow to generate a quick basis, instead of having to locate it then change options to suit the needs.
- is there a better way to organize data than the way now?

B.4 Do you have any other comments about the software or the task? (Question 21)

B.4.1 Responses from users of Interface A (flat)

- the options are unclear, ergo it is hard to manipulate new graphs [marked beside #10]
- note clarity of graph options [marked beside #11]
- how will data entry be accomplished?
- I really like this application as it gives the user many different choices. I think that it would be very useful for research as it would enable the user to pick up on trends that might otherwise be missed.
- task button is unclear – the entire bar looks like a button
- larger font type for axis labels.
- interface is still a bit confusing, what components do what to a graph.
- Different labelling might help to clarify things. Sorting of data, instead of dataAmplification.
- the options window (on the right) is too large. Other than that quite well laid out.
- graphs and colours are easy on the eyes (unlike, say, Excel)
- it might make the scatter plots more attractive if I knew how to read them (was familiar with them) in an intuitive way, but I don't so I didn't use them.
- the task seems well chosen - meaningful without too much detail.
- while esthetics is not vital for a piece of (business) software such as this, I feel that something pleasing to the eye is more desireable to use.
- note: he said he likes the preview features of Excel – immediate changes weren't fast enough!)
- it was most useful at helping visualise the information. I think also that maybe it provided me ways to find what I was expecting - in question 3 I did this, and later found a chart that seemed to suggest the opposite of what I was looking for. I think I was imposing my own expectations on the data by manipulating it to show what I expected.

B.4.2 Responses from users of Interface B (hierarchical)

- Useful, and possibly fun to goof with casually
- I found it difficult to understand the multiple types of attribute quickly enough to use them effectively right away. This was especially true for the graphs, which I have not used much lately (or maybe because it's Friday?)
- very good for visual people/learners but I found it hard to draw a mental picture of the task. Finding the right graph was pretty easy, but drawing the picture in my head was not.
- I didn't want to explore [marked beside #7]
- software not much worse, based on style of interaction [marked beside #14]
- I think my judgements may be unnecessarily harsh on the multiple-choice portion of the this task because of how I originally interpreted the instructions.
- I also think that I have become heavily MS office oriented, in the sense that I wish to be able to generate the graphs of my choosing before starting my exploration. Perhaps if I were offered the choice of my own, then generating spaces from my starting position, it would work better for me.
- looks *very* interesting for taking a deeper look at data, but may be too unwieldy for "quick-and-dirty" visualizations
- interface between this hierarchical viewer and "flat"
- only say unlikely that I'd use commercial verison [marked beside #12 & #13] with reservations
- it appears to be a very useful tool for analysing and interpretation of large data.
- I like the method of exploring representations as opposed to the classic graphs based on setting attributes or even the normal restrictive wizards that are usually available.
- powerful software, easy to use, assuming prior knowledge of graphs and the information needed.
- for exploring data and finding new relationships, etc. this is great. I think that I might be frustrated if I was just trying to build a graph that I had in my mind, however.

- how much processing power is needed to generate the objects quickly to make it practical?
- takes a while to learn all the features

Appendix C

A sample output file

```
#Inventor V2.1 ascii

Separator {
    DEF State_Information Info {
string "
BoundingBox: min (0,0,0), max (1,0.866,0);
DataSize: 40;
DataSources: [3] 1 (newLabel) 2 (newLabel) 0 (newLabel);
GraphAspectRatio: 1.000000;
GraphCount: 1;
LineColour: function linearRamp_RGB length length start (0.5 0.25 0.25) end (1 1 1);
LinePattern: function line_pattern length length;
LineWidth: function line_width length length;
Presentation: overlaid;"
    }
    Complexity {
type SCREEN_SPACE
value 0.5
    }
    DEF camera Group {
Info {
    string "Orthographic camera looking at XY plane"
}
OrthographicCamera {
    position 0.5 0.433 0.66143
    orientation 0 0 1  0
    aspectRatio 1
    nearDistance 0.660768
    farDistance 0.662091
```

```
    focalDistance 0.66143
    height 1.32286
}
    }
    Separator {
DEF lighting Group {
    Info {
string "Base (diffuse) lighting"
    }
    LightModel {
model BASE_COLOR
    }
}
DEF Shape Group {
    Info {
string "Create a trilinear plot"
    }
    Scale {
    }
    Group {
Coordinate3 {
    point [ 0.5 0 0,
      0.25 0.433 0,
      0.75 0.433 0,
      0.67 0.27712 0,
      0.525 0.0433 0,
      0.275 0.3897 0,
      0.7 0.433 0,
      0.44 0.46764 0,
      0.55 0.0866 0,
      0.3 0.3464 0,
      0.65 0.433 0,
      0.53 0.39836 0,
      0.575 0.1299 0,
      0.325 0.3031 0,
      0.6 0.433 0,
      0.68 0.2598 0,
      0.6 0.1732 0,
      0.35 0.2598 0,
      0.55 0.433 0,
      0.96 0.03464 0,
      0.625 0.2165 0,
```

```
      0.375 0.2165 0,
      0.5 0.433 0,
      0.44 0.46764 0,
      0.65 0.2598 0,
      0.4 0.1732 0,
      0.45 0.433 0,
      0.8 0.1732 0,
      0.675 0.3031 0,
      0.425 0.1299 0,
      0.4 0.433 0,
      0.48 0.45032 0,
      0.7 0.3464 0,
      0.45 0.0866 0,
      0.35 0.433 0,
      0.66 0.29444 0,
      0.725 0.3897 0,
      0.475 0.0433 0,
      0.3 0.433 0,
      0.41 0.50228 0 ]
}
    }
    PointSet {
    }
}
    }
    DEF Annotation Separator {
Info {
    string "Annotate trilinear plot"
}
Coordinate3 {
    point [ 0 0 0,
      1 0 0,
      0.5 0.866 0,
      0 0 0 ]
}
LineSet {
    numVertices -1
}
    }
}
```

Bibliography

[1] R. Abraham, P. Broadwell, and B. Beach. Visual math: a fantasy for the future of education. *SIGCUE Bulletin*, 14:2–10, January 1980.

[2] B. L. Allen. *Information Tasks: Toward a User-Centered Approach to Information Systems*. Academic Press, 1996.

[3] D. F. Andrews, E. B. Fowlkes, and P. A. Tukey. Some approaches to interactive statistical graphics. In W. S. Cleveland and M. E. McGill, editors, *Dynamic Graphics for Statistics*. Chapman and Hall, New York, 1988.

[4] V. Anupam, C. Bajaj, and W. Zhang. Scientific problem solving in a distributed and collaborative multimedia environment. *Mathematics and computers in simulation*, 36(4/6), 1994.

[5] W. R. Ashby. Design for an intelligence-amplifier. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 215–233. Princeton University Press, 1956.

[6] G. Avrahami, K. P. Brooks, and M. H. Brown. A two-view approach to constructing user interfaces. *Computer graphics*, 23(3), 1989.

[7] R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg. *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann Publishers, San Francisco, second edition, 1995.

[8] M. P. Baker and C. Bushell. After the storm: Considerations for information visualization. Technical Report NCSA-TR029, NCSA, University of Illinois, January 1995. Also appeared in *IEEE Computer Graphics and Applications*, May 1995.

[9] T. Banchoff and D. P. Cervone. Illustrating beyond the third dimension. *Leonardo*, 25(3), 1992.

[10] T. F. Banchoff. Interactive computer graphics, higher dimensional geometry and electronic publication: From flatland to hypertext. *The Serials Librarian*, 24(3/4), 1994.

[11] J. Bertin. *Graphics and Graphic Information-Processing*. de Gruyter, 1981. Translated by W. J. Berg and P. Scott.

[12] J. Bertin. *Semiology of graphics : diagrams, networks, maps.* University of Wisconsin Press, 1983. Translated by W. J. Berg.

[13] C. Beshers and S. Feiner. AutoVisual: Rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, July 1993.

[14] F. J. Bitz and N. J. Zabusky. David and "visiometrics": Visualizing and quantifying evolving amorphous objects. *Computers in Physics*, November/December 1990.

[15] A. Blackwell and Y. Engelhart. A taxonomy of diagram taxonomies. In *Proceedings of Thinking with Diagrams 98: Is there a science of diagrams?*, pages 60–70, 1998.

[16] A. F. Blackwell. Correction: A picture is worth 84.1 words. In C. Kann, editor, *Proceedings of the First ESP Student Workshop*, 1997.

[17] M. A. Boden. Creativity. In M. A. Boden, editor, *Artificial Intelligence: Handbook of Perception and Cognition*, chapter 9. Academic Press, second edition, 1996.

[18] N. Bowers and K. W. Brodlie. Chameleon: A holistic approach to visualization. In M. Grave and W. T. Hewitt, editors, *Visualization in Scientific Computing*, chapter 18. Springer-Verlag, New York, 1994.

[19] D. W. Brisson, editor. *Hypergraphics: Visualizing Complex Relationships in Art, Science and Technology.* AAAS Selected Symposia Series, No. 24. Westview Press, Boulder, Colorado, 1978.

[20] H. E. Brisson. Visualization in art and science. *Leonardo*, 25(3/4), 1992.

[21] K. Brodlie. Scientific visualization - past, present and future. *Nuclear Instruments and Methods in Physics Research, Section A*, 354(1), 1995.

[22] K. Brodlie, L. Brankin, et al. GRASPARC - a problem solving environment integrating computation and visualization. In G. M. Nielson and D. Bergeron, editors, *Proceedings of Visualization 93*, 1993.

[23] M. Brown, J. Domingue, and J. Stasko. Software visualization. *SIGCHI bulletin*, 26(4), 1994.

[24] M. H. Brown. *Algorithm Animation.* MIT Press, 1987.

[25] M. H. Brown and J. Hershberger. Color and sound in algorithm animation. *Computer*, 25(12):52–63, 1992.

[26] M. H. Brown and R. Sedgewick. A system for algorithm animation. *Computer Graphics*, 18(3), 1984.

[27] J. S. Bruner. *Toward a Theory of Instruction.* Harvard University Press, Cambridge, MA, 1966.

[28] J. S. Bruner. *Acts of Meaning*. Harvard University Press, Cambridge, MA, 1990.

[29] V. Bush. As we may think. *Atlantic Monthly*, pages 101–108, 1945.

[30] M. Callahan, D. Hoffman, and J. Hoffman. Computer graphics tools for the study of minimal surfaces. *Communications of the ACM*, 31(6):641–661, 1988.

[31] G. Cameron. Modular visualization environments: Past, present, and future. *Computer Graphics*, November 1995.

[32] D. T. Campbell. Blind variation and selective retention in scientific discovery. *Pyschological Review*, pages 380–400, 1960.

[33] S. Card. The information visualizer, an information workspace. In *ACM CHI'91 Conference Proceedings*, 1991.

[34] S. K. Card. Human factors and the intelligent interface. In G. Kohl, G. Nassau, and S.S. Nassau, editors, *Proceedings of a Symposium of the Metropolitan Chapter of the Human Factors Society*, November 1984.

[35] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan-Kaufman, 1999.

[36] M. S. T. Carpendale. *A Framework for Elastic Presentation Space*. PhD thesis, Simon Fraser University, 1999.

[37] C. M. Carswell, S. Frankenberger, and D. Bernhard. Graphing in depth: perspectives on the use of three-dimensional graphs to represent lower-dimensional data. *Behaviour and Information Technology*, 10(6):459–474, 1991.

[38] S. M. Casner. A task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics*, 10(2), April 1991.

[39] A. Chabert et al. Java object-sharing in habanero. *Communications of the ACM*, 41(6), 1998.

[40] M. Chapman. *Constructive Evolution: Origins and Development of Piaget's thought*. Cambridge University Press, 1988.

[41] S. G. Charlton. Questionnaire techniques for test and evaluation. In T. G. O'Brien and S. G. Charlton, editors, *Handbook of Human Factors Testing and Evaluation*. Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[42] E. H. Chi. Principles for information visualization spreadsheets. *IEEE Computer Graphics and Applications*, July/August 1998.

[43] N. Clark. Tables and graphs as language. In T. J. Boardman and I. M. Stefanski, editors, *Computer Science and Statistics: Symposium on the Interface*, pages 83–89. American Statistical Association, 1986.

[44] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, NJ, 1993.

[45] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[46] G. W. Cobb. *Introduction to design and analysis of experiments*. Springer, New York, 1998.

[47] B. M. Collins. Data visualization - has it all been seen before? In R. A. Earnshaw and D. Watson, editors, *Animation and Scientific Visualization: Tools and Applications*, pages 3–28. Academic Press, 1993.

[48] D. J. Cowen. GIS versus CAD versus DBMS: What are the differences? *Photogrammetric engineering and remote sensing*, 54(11), 1988.

[49] D. J. Cox. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. *Leonardo*, 21(3), 1988.

[50] M. Csikszentmihalyi and K. Sawyer. Creative insight: The social dimension of a solitary moment. In R. J. Sternberg and J. E. Davidson, editors, *The Nature of Insight*, pages 329–364. MIT Press, Cambridge, MA, 1995.

[51] G. J. Culler. Mathematical laboratories: A new power for the physical sciences. In M. Klerer and J. Reinfelds, editors, *Interactive Systems for Experimental Applied Mathematics*, pages 155–166, New York, 1968. Academic Press.

[52] S. Cunningham, J. R. Brown, and M. McGrath. Visualization in science and engineering education. In *Visualization in Scientific Computing*, pages 48–57. IEEE Computer Society Press, 1990.

[53] W. Davis, H. Porta, and J. Uhl. *Calculus and Mathematica*. Addison-Wesley, 1994.

[54] K. Davlin. Computers and mathematics. *Notices of the AMS*, 40(8), 1993.

[55] R. Dawkins. *The Blind Watchmaker*. W. W. Norton and Company, New York, 1986.

[56] M. O. Dayhoff. A contour-map program from x-ray crystallography. *Communications of the ACM*, 6(10), 1963.

[57] L. Dieci, J. Lorenz, and R. D. Russell. Numerical calculation of invariant tori. *SIAM J. Sci. Stat. Comput.*, 12(3):607–647, 1991.

[58] J. L. Dolby, N. Clark, and W. H. Rogers. The language of data: A general theory of data. In T. J. Boardman and I. M. Stefanski, editors, *Computer Science and Statistics: Symposium on the Interface*, pages 96–103. American Statistical Association, 1986.

[59] J. Domingue, B. A. Price, and M. Eisenstadt. A framework for describing and implementing software visualization systems. In *Graphics Interface 1992 Conference Proceedings*, pages 53–60, 1990.

[60] R. L. Dominowski and P. Dallob. Insight and problem solving. In R. J. Sternberg and J. Davidson, editors, *The Nature of Insight*, pages 33–62. MIT Press, Cambridge, MA, 1995.

[61] D. S. Dyer. A dataflow toolkit for visualization. *IEEE Computer Graphics and Applications*, July 1990.

[62] J. R. Eastman. Cognitive models and cartographic design research. *The Cartographic Journal*, 22(4), 1985.

[63] G. Edwards. Visualization – the second generation. *IEEE transactions on image processing*, 1, 1992.

[64] M. Eisenstadt, J. Domingue, T. Rajan, and E. Motta. Visual knowledge engineering. *IEEE Transactions on Software Engineering*, 16(10):1164–1177, October 1990.

[65] T. E. Elsner. The graphics microcomputer as a visual aid in the classroom. *Coll. Microcomput. (USA)*, 1:19–23, Spring 1983.

[66] D. C. Engelbart. A conceptual framework for the augmentation of man's intellect. In Howerton and Weeks, editors, *Vistas in Information Handling*, volume 1, pages 1–29. Spartan Books, London, 1963.

[67] D. C. Engelbart. The augmented knowledge workshop. In A. Goldberg, editor, *A History of Personal Workstations*, pages 187–236. ACM Press, 1988.

[68] D. Epstein, S. Levy, and R. de la Llave. About this journal. *Journal of Experimental Mathematics*, 1(1), 1992.

[69] M. Fallshore and J. W. Schooler. The verbal vulnerability of perceptual expertise. *Journal of Experimental Pyschology: Learning, Memory, and Cognition*, 21(6), 1995.

[70] S. M. Fangmeier. The scientific visualization process. In *Computer Graphics In Science, SIGGRAPH 88 Course #20 Notes*, pages 26–38. ACM SIGGRAPH, 1988.

[71] S. Feiner. Apex: An experiment in the automated creation of pictorial explanations. *IEEE Computer Graphics and Applications*, 5(11):29–38, 1985.

[72] S. Feiner, B. MacIntyre, and D. D. Seligman. Annotating the real world with knowledge-based graphics on a see-through head-mounted display. In *Proceedings of Graphics Interface '92*, pages 78–85, 1992.

[73] S. Feiner, D. Salesin, and T. Banchoff. Dial: A diagramatic animation language. *IEEE Computer Graphics and Applications*, 2:43–54, September 1982.

[74] R. Finke. *Creative Imagery: Discoveries and Inventions in Visualization*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

[75] R. A. Finke, T. B. Ward, and S. M. Smith. *Creative Cognition: Theory, Research, and Applications*. MIT Press, Cambridge, MA, 1992.

[76] G. Fischer and H.-D. Boecker. The nature of design processes and how computer systems can support them. In P. Degano and E. Sandewall, editors, *Integrated Interactive Computing Systems*, pages 73–86. North-Holland, 1983.

[77] G. Fischer, S. R. Henninger, and D. F. Redmiles. Intertwining query construction and relevance evaluation. In *Human Factors in Computing Systems, CHI '91 Conference Proceedings*, pages 55–62. ACM, 1991.

[78] G. Fischer, R. McCall, and A. Morch. Janus: Integrating hypertext with a knowledge-based design environment. In *Proceedings of Hypertext '89*, pages 105–117. ACM, 1989.

[79] G. Fischer and B. Reeves. Beyond intelligent interfaces: Exploring, analyzing, and creating success models of cooperative problem solving. *Journal of Applied Intelligence*, 1:311–332, 1992.

[80] J. Fiske. *Introduction to communication studies*. Studies in culture and communication. Routledge, New York, second edition, 1990.

[81] J. Foley and W. Ribarsky. Next-generation data visualization tools. In L. Rosenblum et al., editors, *Scientific Visualization*. Academic, 1994.

[82] P. Forcheri, E. Lemut, and M. T. Molfino. The GRAF system: an interactive graphic system for teaching mathematics. *Comput. and Educ.*, 7:177–182, 1983.

[83] I. Fujishiro, Y. Takeshima, Y. Ichikawa, and K. Nakamura. Gadget: Goal-oriented application design guidance for modular visualization environments. In *Proceedings of Visualization '97*, 1997.

[84] N. Fuller and P. Prusinkiewicz. Geometric modeling with euclidean constructions. In N. Magnenat-Thalmann and D. Thalmann, editors, *New Trends in Computer Graphics: Proceedings of CGI '88*, pages 379–391, 1988.

[85] G. W. Furnas. New graphical reasoning models for understanding graphical interfaces. In *ACM CHI'91 Conference Proceedings*, 1991.

[86] G. W. Furnas, L. M. Gomez, T. K. Landauer, and S. M. Dumais. Statistical semantics: how can a computer use what people name things to guess what things people mean when they name things? In *Proceedings of Human Factors in Computer Systems*, New York, 1982. ACM.

[87] P. W. Gaffney et al. Nexus: Towards a problem solving environment (pse) for scientific computing. *ACM SIGNUM Newsletter*, 21:13–24, 1986.

[88] M. Gahegan. Four barriers to the development of effective exploratory visualization tools for the geosciences. Available at http://www.geog.psu.edu/∼mark/visworkshop/visproblems.html, 1998.

[89] D. Gentner and J. Nielsen. The anti-Mac interface. *CACM*, 39(8):70–82, 1996.

[90] N. Gershon. The power and frailty of images: Is a picture worth a thousand words? *Computer Graphics*, November 1995.

[91] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.

[92] J. Goldstein, S. F. Roth, and J. Mattis. A framework for knowledge-based interactive data exploration. *Journal of visual languages and computing*, 5(4), December 1994. p 339.

[93] H. H. Goldstine. *A History of Numerical Analysis from the 16th through the 19th Century*. Springer Verlag, New York, 1977.

[94] K. E. Green and D. H. Schroeder. Psychometric quality of the verbalizer-visualizer questionnaire as a measure of cognitive syle. *Psychological Reports*, 66(3), June 1990.

[95] I. Greif, editor. *Computer-supported cooperative work : a book of readings*. Morgan Kaufmann, San Mateo, CA, 1988.

[96] G. Grinstein and H. Levkowitz, editors. *Perceptual issues in visualization*. Springer Verlag, New York, 1995.

[97] G. Grinstein, R. M. Pickett, and M. G. Williams. EXVIS: An exploratory visualization system. In *Proceedings of Graphics Interface '89*, 1989.

[98] C. Gunn. Remarks on mathematical courseware. In S. Cunningham and R. J. Hubbold, editors, *Interactive Learning Through Visualization*, pages 115–128. Springer-Verlag, 1992.

[99] R. B. Haber, B. Lucas, and N. Collins. A data model for scientific of visualization with provisions for regular and irregular grids. In *Proc. of Visualization'91*, pages 298–305, San Diego, CA, 1991.

[100] R. B. Haber and D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, 1990.

[101] J. Hadamard. *The Pyschology of Invention in the Mathematical Field*. Dover, 1954.

[102] P. E. Haeberli. Conman: A visual programming language for interactive graphics. In *Proceedings of SIGGRAPH 1988*, volume 22, 1988.

[103] R. Hamming. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, 1962.

[104] A. J. Hanson, T. Munzner, and G. Francis. Interactive methods for visualizable geometry. *Computer*, 27(7):73–83, 1994.

[105] R. D. Harding. A structured approach to computer graphics for mathematical uses. *Comput. and Educ.*, 7:1–19, 1983.

[106] J. Hartman, J. Wernecke, and R. Carey. *The VRML 2.0 Handbook: Building Moving Worlds on the Web*. Addison-Wesley, 1996.

[107] J. R. Hayes. Cognitive processes in creativity. In J. A. Glover, R. R. Ronning, and C. R. Reynolds, editors, *Handbook of Creativity*, pages 135–146. Plenum Press, New York, 1989.

[108] D. Heller, P. M. Ferguson, and D. Brennan. *Motif Programming Manual (The Definitive Guides to the X Window System, Volume 6A)*. O'Reilly and Associates, 1994.

[109] D. H. Hepting. Approximation and visualization of sets defined by iterated function systems. Master's thesis, University of Regina, Regina, Saskatchewan, Canada, 1991.

[110] D. H. Hepting. What's a picture really worth? http://www.cs.sfu.ca/∼dhepting/personal/research/words/, March 1999.

[111] D. H. Hepting, G. Derks, K. D. Edoh, and R. D. Russell. Qualitative analysis of invariant tori in a dynamical system. In G. M. Nielson and D. Silver, editors, *Visualization '95 Proceedings*. IEEE Computer Society Press, 1995.

[112] D. H. Hepting, K. D. Edoh, G. Derks, and R. D. Russell. Visualization as a qualitative tool for the computation of invariant tori. Presented at the 3rd SIAM Conference on Applications of Dynamical Systems at Snowbird, Utah, May 1995.

[113] D. H. Hepting, F. D. Fracchia, J. C. Dill, and R. D. Russell. Cogito: a system for computer-aided visualization. Technical Report CMPT TR 96-02, Simon Fraser University, 1996. Unpublished technical sketch presented at SIGGRAPH 96.

[114] D. H. Hepting, F. D. Fracchia, J. C. Dill, and R. D. Russell. A personal paradigm for computer-aided visualization. In D. S. Ebert, editor, *Workshop on New Paradigms in Information Visualization and Manipulation*, pages 46–49, 1996. Held in conjunction with the ACM Conference on Information and Knowledge Management.

[115] D. H. Hepting, W. Huang, and R. D. Russell. A visual tool for moving mesh numerical methods. Technical Report CMPT TR 95-09, Simon Fraser University, 1995.

[116] D. H. Hepting, P. Prusinkiewicz, and D. Saupe. Rendering methods for iterated function systems. In H.-O. Peitgen, J. M. Henriques, and L. F. Penedo, editors, *Fractals in the Fundamental and Applied Sciences*, pages 183–224, New York, 1991.

North-Holland. Reprinted in *Fractals: from Folk Art to Hyperreality, SIGGRAPH 92 Course #12 Notes.*, Edited by P. Prusinkiewicz. ACM SIGGRAPH, New York, 1992, pp. 3-1 to 3-41.

[117] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* Chelsea, 1952.

[118] C. M. Hoffman, E. N. Houstis, and D. G. Taylor. Softlab - a virtual laboratory for computational science. *Mathematics and computers in simulation*, 36(4/6), 1994.

[119] S. R. Holtzman. *Digital Mantras.* MIT Press, 1994.

[120] J. Horgan. The death of proof. *Scientific American*, pages 93–103, October 1993.

[121] S. W. Huck and W. H. Cormier. *Reading Statistics and Research.* HarperCollins College Publishers, New York, second edition, 1996.

[122] I. Illich. *Tools for Conviviality.* Harper and Row, 1973.

[123] E. Imhof. *Cartographic Relief Representation.* Walter de Gruyter, New York, 1982.

[124] Silicon Graphics Incorporated. Topics in IRIX programming. Available at http://techpubs.sgi.com:/library.

[125] SPSS Incorporated. *SPSS Base 9.0 : Applications Guide.* Prentice Hall, 1999.

[126] M. I. Isaak and M. A. Just. Constraints on thinking in insight and invention. In R. J. Sternberg and J. E. Davidson, editors, *The Nature of Insight*, pages 281–325. MIT Press, Cambridge, MA, 1995.

[127] M. E. Jessup. Scientific visualization: Viewpoint on collaborations of art, science, and engineering. *SIGBIO Newsletter*, pages 1–9, February 1992.

[128] E. Kaltofen and S. M. Watt, editors. *Computers and Mathematics.* MIT, 1989.

[129] J. Karat. Evolving the scope of user-centered design. *Communications of the ACM*, pages 33–38, 1997.

[130] G. Kaufmann. Mental imagery in problem solving. In A. A. Sheikh, editor, *International Review of Mental Imagery*, volume 1, pages 23–55. Human Sciences Press, New York, 1984.

[131] A. Kay. User interface: A personal view. In B. Laurel and S. J. Mountford, editors, *The Art of Human-Computer Interface Design.* Addison-Wesley, 1990.

[132] G. W. Kelling. *Language: Mirror, Tool, and Weapon.* Nelson-Hall, 1975.

[133] L. Kjelldahl. Drawing of an unusual kind of diagrams – nomogram drawing. In K. Bo and H. A. Tucker, editors, *Proceedings of Eurographics '84.* Elsevier, 1984.

[134] S. Kochhar. A prototype system for design automation via the browsing paradigm. In *Proceedings of Graphics Interface '90*, 1990.

[135] S. Kochhar. Ccad: A paradigm for human-computer cooperation in design. *IEEE Computer Graphics and Applications*, 14(3), 1994.

[136] S. Kochhar, M. Friedell, and M. LaPolla. Cooperative, computer-aided design of scientific visualizations. In *Proceedings of Visualization '91*, 1991.

[137] S. Kochhar, J. Marks, and M. Friedell. Interaction paradigms for human-computer cooperation in graphical-object modelling. In S. MacKay and E. M. Kidd, editors, *Proceedings of Graphics Interface '91*, 1991.

[138] A. Koestler. *The Act of Creation.* Dell, New York, 1964.

[139] D. A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[140] J. R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems).* MIT Press, 1992.

[141] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–29, 1964.

[142] I. Lakatos. *Proofs and refutations: the logic of mathematical discovery.* Cambridge University Press, New York, 1976.

[143] P. Langley and R. Jones. A computational model of scientific insight. In R. J. Sternberg, editor, *The nature of creativity: contemporary psychological perspectives*, pages 177–201. Cambridge University Press, New York, 1988.

[144] J. H. Larkin and H. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[145] J. P. Lee and G. G. Grinstein, editors. *Data Issues for Data Visualization.* Number 871 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

[146] P. M. Lester. Digital literacy: Visual communication and computer images. *Computer Graphics*, pages 25–27, November 1995.

[147] J. Levine, T. Mason, and D. Brown. *lex & yacc.* O'Reilly and Associates, second edition, 1992.

[148] M. Levoy. Spreadsheets for images. In A. Glassner, editor, *Computer Graphics: SIGGRAPH 94 Proceedings*, 1994.

[149] J. C. R. Licklider. Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics*, pages 4–11, 1960.

[150] M. Lindholm and T. Sarjakoski. Designing a visualization user interface. In A. M. MacEachren and D. R. F. Taylor, editors, *Visualization in Modern Cartography*, volume 2, pages 167–184. Pergamon/Elsevier Science, New York, 1994.

[151] J. Lohse, H. Rueter, K. Biolsi, and N. Walker. Classifying visual knowledge representations: A foundation for visualization research. In *Proceedings of Visualization 90*, pages 131–138, 1990.

[152] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In *Proceedings of SIGGRAPH 87*, 1987.

[153] B. Lucas et al. An architecture for a scientific visualization system. In A. E. Kaufman and G. M. Nielson, editors, *Proceedings of Visualization '92*, 1992.

[154] W. Lytle. The dangers of glitziness and other visualization faux pas. In *Computer Graphics: SIGGRAPH '93 Visual Proceedings*. ACM Press, 1993.

[155] A. M. MacEachren. *How maps work: representation, visualization, and design*. Guilford Press, New York, 1995.

[156] A. M. MacEachren and D. R. F. Taylor, editors. *Visualization in Modern Cartography*, volume 2. Pergamon/Elsevier Science, 1994.

[157] C. Machover. A brief, personal history of computer graphics. *Computer*, November 1978.

[158] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2), 1986.

[159] T. W. Malone, K. R. Grant, K.-Y. Lai, R. Rao, and D. Rosenblitt. Object lens: A "spreadcheet" for cooperative work. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88)*, pages 115–124. ACM, 1988.

[160] B. B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, New York, 1982.

[161] J. Marks et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Computer Graphics: SIGGRAPH '97 Conference Proceedings*, 1997.

[162] R. Marshall, J. Kempf, S. Dyer, and C.-C. Yen. Visualization methods and simulation steering for a 3d turbulence model of lake erie. In R. Riesenfeld and C. Sequin, editors, *Computer Graphics: 1990 Symposium on Interactive 3D Graphics*, volume 24, pages 89–97, March 1990.

[163] B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in scientific computing. *Computer Graphics*, 21(6), November 1987.

[164] G. N. McGregor. Visual illiteracy: Implications in the development of scientific visualization software. *Computer Graphics*, November 1995.

[165] C. McGuinness. Expert/novice use of visualization tools. In A. M. MacEachren and D. R. F. Taylor, editors, *Visualization in Modern tography*, volume 2, pages 185–199. Pergamon/Elsevier Science, 1994.

[166] M. McLuhan. *Understanding Media: the Extensions of Man*. McGraw-Hill, Toronto, 1964.

[167] K.-H. Meine. Cartographic communication links and a cartographic alphabet. In L. Guelke, editor, *Cartographica: The Nature of Cartographic Communication*. University of Toronto Press, 1977. Monograph No. 19.

[168] P. Messaris. *Visual "Literacy": Image, Mind, and Reality*. Westview Press, San Francisco, 1994.

[169] B. A. Meyers. Taxonomies of visual programming and program visualization. *Journal of visual languages and computing*, 1(1):97–123, 1990.

[170] A. I. Miller. *Insights of Genius: Imagery and Creativity in Science and Art*. Copernicus (an imprint of Springer-Verlag), 1996.

[171] P. Mussio, M. Pietrogrande, and M. Protti. Simulation of hepatological models: a study in visual interactive exploration of scientific problems. *Journal of visual languages and computing*, 2(1), 1991.

[172] T. Nadas and A. Fournier. GRAPE: An environment to build display processes. In *Computer Graphics: Proceedings of SIGGRAPH 87*, 1987.

[173] S. Nakamura. *Numerical Analysis and Graphic Visualization with MATLAB*. Prentice-Hall, 1996.

[174] S. D. Neill. The reference process and the philosophy of karl popper. *RQ*, pages 309–319, Spring 1985.

[175] Y. Neumann and E. Finaly. The problem-solving environment and students' problem-solving orientation. *Journal of research and development in education*, 22(2), 1989.

[176] A. Newell and H. Simon. *Human Problem Solving*. Prentice-Hall, 1972.

[177] G. M. Nielson. Visualization in scientific and engineering computation. *Computer*, 1991.

[178] A. M. Noll. A computer technique for displaying n-dimensionae hyperobjects. *Communications of the ACM*, 10(8), August 1967.

[179] D. A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988.

[180] K. L. Norman, L. J. Weldon, and B. Shneiderman. Cognitive layouts of windows and multiple screens for user interfaces. *Int. J. Man-Machine Studies*, 25:229–248, 1986.

[181] J. D. North. The rational behaviour of mechanically extended man. Technical report, Boulton Paul Aircraft Ltd., Wolverhampton, England, September 1954.

[182] A. Nye and T. O'Reilly. *Volume 4M: X Toolkit Intrinsics Programming Manual (Motif Edition)*. O'Reilly and Associates, second edition, 1992.

[183] S. Ohlsson. Restructuring revisited: An information processing theory of restructuring and insight. *Scandinavian Journal of Pyschology*, 25:117–129, 1984.

[184] A. F. Osborn. *Applied imagination*. Scribners, 1963.

[185] R. E. Palmer. *Hermeneutics: Interpretation Theory in Scleiermacher, Dilthey, Heidegger, and Gadamer*. Northwestern University Press, Evanston, IL, 1969.

[186] A. Pang and N. Alper. Mix & Match: A construction kit for visualization. In *Visualization '94 Proceedings*, pages 302–309, 1994.

[187] S. G. Parker, M. Miller, C. D. Hansen, and C. R. Johnson. An integrated problem solving environment: The scirun computational steering system. In *31st Hawaii International Conference on System Sciences (HICSS-31)*, volume VII, January 1998.

[188] S. G. Parker, D. M. Weinstein, and C. R. Johnson. The scirun computational steering software system. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, chapter 1. Birkhauser, 1997.

[189] J. Patten and K.-L. Ma. A graph-based interface for representing volume visualization results. In *Proceedings Graphics Interface '98*, 1998.

[190] J. Pearsall and P. Hanks, editors. *The new Oxford dictionary of English*. Clarendon Press, Oxford, 1998.

[191] D. N. Perkins. Insight in minds and genes. In R. J. Sternberg and J. E. Davidson, editors, *The Nature of Insight*, pages 495–533. MIT Press, Cambridge, MA, 1995.

[192] B. B. Petchenik. Cognition in cartography. In L. Guelke, editor, *Cartographica: The Nature of Cartographic Communication*. University of Toronto Press, 1977. Monograph No. 19.

[193] P. R. Peterson. A genetic engineering approach to texture synthesis. Master's thesis, Simon Fraser University, 1997.

[194] M. Petre, A. F. Blackwell, and T. R. G. Green. Cognitive questions in software visualization. In J. Stasko et al., editors, *Software visualization: programming as a multimedia experience*. MIT Press, Cambridge, MA, 1998.

[195] M. Phillips, S. Levy, and T. Munzner. Geomview: An interactive geometry viewer. *Notices of the American Math Society*, 40(8):985–988, October 1993.

[196] R. L. Phillips. An interpersonal multimedia visualization system. *IEEE Computer Graphics and Applications*, May 1991.

[197] N. Pitcher. Visualization in linear algebra. *International journal of mathematical education*, 22(3):387–394, May 1991.

[198] W. Playfair. *The Commercial and Political Atlas*. Corry, London, 1786.

[199] K. R. Popper. *Objective Knowledge: An Evolutionary Approach*. Oxford, 1972.

[200] B. A. Price, I. S. Small, and R. M. Baecker. A taxonomy of software visualization. In *25th Hawaii International Conference on System Sciences Conference Proceedings*, 1992.

[201] J. Rasure, C. Williams, D. Argiro, and T. Sauer. A visual language and software development environment for image processing. *International Journal of Imaging Systems and Technology*, 2, 1990.

[202] T. Reenskaug. *Working with Objects*. Manning/Prentice Hall, Greenwich, 1995.

[203] B. Reeves and C. Nass. *The media equation : how people treat computers, television, and new media like real people and places*. CSLI Publications, New York, 1996.

[204] L. L. Rezabek and T. J. Ragan. Using computers to facilitate visual thinking: an analogy between visual and verbal processing. *Reading Pyschology: An International Quarterly*, 9:455–467, 1988.

[205] H. Rheingold. *Tools for Thought: the people and ideas behind the next computer revolution*. Simon and Schuster, New York, 1985.

[206] A. Richardson. Verbalizer-visualizer: A cognitive style dimension. *Journal of Mental Imagery*, 1:109–126, 1977.

[207] H. W. J. Rittel. Second-generation design methods. In *Developments in Design Methodology*, pages 317–327. Wiley and Sons, 1984.

[208] P. Robertson. A methodology for choosing data representations. *IEEE Computer Graphics and Applications*, 11(3), 1988.

[209] E. Rogers. A cognitive theory of visual interaction. In B. Chandrasekaran, J. Glasgow, and N. H. Narayanan, editors, *Diagrammatic reasoning : cognitive and computational perspectives*, chapter 14. MIT Press, Cambridge, MA, 1995.

[210] B. E. Rogowitz and L. A. Treinish. Data structures and perceptual structures. *SPIE*, 1913:600–612, 1993.

[211] S. F. Roth, J. Kolojejchick, J. Mattis, and M. Chuah. Sagetools: An intelligent environment for sketching, browsing, and customizing data-graphics. In *Proceedings CHI'95 Human Factors in Computing Systems*. ACM Press, 1995.

[212] J. Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley and Sons, Toronto, 1994.

[213] D. E. Rumelhart and D. A. Norman. Representation in memory. In R. C. Atkinson et al., editors, *Steven's Handbook of Experimental Psychology*, chapter 8. Wiley, New York, second edition, 1988.

[214] M. A. Sass and W. D. Wilkinson, editors. *Symposium on Computer Augmentation of Human Reasoning*. Spartan Books, 1965.

[215] D. L. Schacter and L. A. Cooper. Implicit and explicit memory for novel visual objects: Structure and function. *Journal of Experimental Pyschology: Learn. Mem. Cognit.*, 19:995–1009, 1993.

[216] R. C. Schank and C. Cleary. Making machines creative. In S. M. Smith, T. B. Ward, and R. A. Finke, editors, *The Creative Cognition Approach*, pages 229–247. MIT Press, Cambridge, MA, USA, 1995.

[217] D. Schattschneider. In praise of amateurs. In D. A. Klarner, editor, *The Mathematical Gardner*, pages 140–166. Wadsworth International, Belmont, CA, USA, 1981.

[218] D. Schattschneider. *Visions of Symmetry: Notebooks, Periodic Drawings, and Related Work of M. C. Escher*. W. H. Freeman and Company, New York, 1990.

[219] C. F. Schmid and S. E. Schmid. *Handbook of graphic presentation*. Wiley, New York, second edition, 1979.

[220] J. W. Schooler and T. Y. Engstler-Schooler. Verbal overshadowing of visual memories: Some things are better left unsaid. *Cognitive Pyschology*, 22:36–71, 1990.

[221] J. W. Schooler and J. Melcher. The ineffability of insight. In S. M. Smith, T. B. Ward, and R. A. Finke, editors, *The Creative Cognition Approach*, pages 97–133. MIT Press, Cambridge, MA, USA, 1995.

[222] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *Proceedings of Visualization 96*, 1996.

[223] H. Senay and E. Ignatius. A knowledge-based system for visualization design. *IEEE Computer Graphics and Applications*, 14(6), 1994.

[224] B. Shneiderman. *Designing the user interface: strategies for effective human–computer interaction*. Addison-Wesley, Reading, MA, USA, second edition, 1992.

[225] M. Sicard and J.-A. Marck. The importance of mental perception when creating research pictures. In R. Sutherland and J. Mason, editors, *Exploiting Mental Imagery with Computers in Mathematics Education*, pages 252–262. Springer-Verlag, New York, 1995.

[226] H. Simon. *Models of Discovery*. Reidel, 1977.

[227] H. Simon. *The Sciences of the Artificial*. MIT Press, 1981.

[228] H. Simon. The scientist as problem solver. In D. Klahr and K. Kotovsky, editors, *Complex Information Processing: the impact of Herbert A. Simon*, pages 375–398. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1989.

[229] J. Simonsen and F. Kensing. Using ethnography in contextual design. *Communications of the ACM*, 40(7):82–88, 1997.

[230] J. A. Simpson and E. S. C. Weiner, editors. *The Oxford English dictionary*. Oxford University Press, New York, second edition, 1989.

[231] K. Sims. Artificial evolution in computer graphics. In R. J. Beach, editor, *Computer Graphics: SIGGRAPH '91 Conference Proceedings*, pages 319–328. ACM Press, 1991.

[232] P. Skagestad. Thinking with machines: Intelligence augmentation, evolutionary epistemology and semiotics. *Journal of Social and Evolutionary Systems*, 16(2):157–180, 1993.

[233] S. M. Smith. Fixation, incubation, and insight in memory and creative thinking. In S. M. Smith, T. B. Ward, and R. A. Finke, editors, *The Creative Cognition Approach*. MIT Press, Cambridge, MA, USA, 1995.

[234] R. R. Springmeyer, M. M. Blattner, and N. L. Max. A characterization of the scientific data analysis process. In A. E. Kaufman and G. M. Nielson, editors, *Proceedings of Visualization '92*, 1992.

[235] J. T. Stasko. The path-transition paradigm: A practical methodology for adding animation to program interfaces. *Journal of Visual Languages and Computing*, 1990.

[236] J. T. Stasko. Tango: A framework and system for algorithm animation. *Computer*, 23(9), 1990.

[237] J. T. Stasko. Understanding and characterizing software visualization systems. In *IEEE Visual Languages Workshop 1992*, 1992.

[238] R. J. Sternberg and J. E. Davidson, editors. *The Nature of Insight*. MIT Press, Cambridge, MA, 1995.

[239] B. E. Stevenson, editor. *The MacMillan Book of Proverbs, Maxims and Famous Phrases*. MacMillan, 1948. formerly known as the *The Home Book of Proverbs, Maxims, and Familiar Phrases*.

[240] I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *SJCC*, 1963.

[241] D. R. F. Taylor. Perspectives on visualization and modern cartography. In A. M. MacEachren and D. R. F. Taylor, editors, *Visualization in Modern Cartography*, volume 2, pages 333–341. Pergamon/Elsevier Science, New York, 1994.

[242] M. Theus. Exploratory data analysis with data desk. *Computational Statistics*, 13:101–115, 1998.

[243] D. A. Thomas. Using computer visualization to motivate and support mathematical dialogues. *The journal of computers in mathematics and science education*, 11(3), 1992.

[244] N. J. W. Thrower. *Maps and Man*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[245] S. Todd and W. Latham. *Evolutionary art and computers*. Academic Press, London, 1992.

[246] L. A. Treinish. Unifying principles of data management for scientific visualizaton. In R. A. Earnshaw and D. Watson, editors, *Animation and Scientific Visualization: Tools and Applications*, pages 141–169. Academic Press, 1993.

[247] L. A. Treinish, D. M. Butler, H. Senay, and G. G. Grinstein S. T. Bryson. Grand challenge problems in visualization software (panel position statements). In A. E. Kaufman and G. M. Nielson, editors, *Visualization '92 Proceedings*, pages 366–371. IEEE Computer Society Press, 1992.

[248] E. R. Tufte. *Envisioning information*. Graphics Press, Chesire, CT, 1990.

[249] E. R. Tufte. *Visual explanations : images and quantities, evidence and narrative*. Graphics Press, Chesire, CT, 1997.

[250] J. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[251] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

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

[253] P. F. Velleman and D. S. Moore. Multimedia for teaching statistics: Promises and pitfalls. *The American Statistician*, 50:217–225, 1996.

[254] P. F. Velleman and P. Pratt. A graphical interface for data analysis. *Journal of Statistical Computation and Simulation*, 32:223–228, 1989.

[255] M. Visvalingam and J. D. Whyatt. The Douglas-Peucker algorithm for line simplification: Re-evaluation through visualization. *Computer Graphics Forum*, 9:213–228, 1990.

[256] J. von Neumann. Recent theories of turbulence. In A. Taub, editor, *Collected Works of John von Neumann*, volume 6. MacMillan, New York, 1963.

[257] H. Wainer. *Visual Revelations*. Copernicus (Springer-Verlag), 1997.

[258] J. Walton and M. Dewar. See what i mean? using graphics toolkits to visualise numerical data. Technical Report TR8/96 IETR/6 (NP3068), NAG, 1996.

[259] C. Ware. The foundations of experimental semiotics: a theory of sensory and conventional representation. *Journal of Visual Languages and Computing*, 4:91–100, 1993.

[260] A. Wehrend and C. Lewis. A problem-oriented classification of visualization techniques. In *Proceedings of Visualization '90*, pages 139–143, 1990.

[261] N. Weiner. *Cybernetics: Or Control and Communication in the Animal and the Machine*. MIT Press, 1948. Pages 27–28.

[262] R. W. Weisberg. Prolegomena to theories of insight in problem solving: A taxonomy of problems. In R. J. Sternberg and J. Davidson, editors, *The Nature of Insight*, pages 157–196. MIT Press, Cambridge, MA, 1995.

[263] R. W. Weisberg and J. W. Alba. An examination of the alleged role of "fixation" in the solution of several "insight" problems. *Journal of Experimental Psychology: General*, 110:169–192, 1981.

[264] J. Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.

[265] T. G. West. Visual thinkers, mental models and computer visualization. In S. Cunningham and R. J. Hubbold, editors, *Interactive Learning Through Visualization*, pages 91–102. Springer-Verlag, 1992.

[266] T. G. West. Forward into the past: A revival of old visual talents with computer visualization. *Computer Graphics*, 29(4), November 1995.

[267] C. Williams, J. Rasure, and C. Hansen. The state of the art of visual languages for visualization. In A. E. Kaufman and G. M. Nielson, editors, *Proceedings of Visualization '92*, 1992.

[268] M. D. Williams. What makes rabbit run? *International Journal of Man-Machine Studies*, 21:333–352, 1984.

[269] T. Winograd and C. F. Flores. *Understanding Computers and Cognition.* Ablex, Norwood, New Jersey, USA, 1985.

[270] M. Woo et al. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1.* Addison-Wesley, 1997.

[271] J. Wood, H. Wright, and K. Brodlie. Collaborative visualization. In R. Yagel and H. Hagen, editors, *Proceedings of Visualization 97*, 1997.

[272] D. B. Wright. *Understanding Statistics: An Introduction for the Social Sciences.* SAGE Publications, Thousand Oaks, CA, 1997.

[273] G. Wyvill, C. McPheeters, and B. Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4), 1986.

[274] X. Yang, M. Burnett, and R. Hossli. Toward visual programming languages for steering scientific computations. *IEEE Computational Science and Engineering*, 1(4), 1994.

[275] N. J. Zabusky. Computational synergetics. *Physics Today*, July 1984.

[276] N. J. Zabusky and D. Silver. Case study: Visualizing classical problems in cfd. In A. E. Kaufman and G. M. Nielson, editors, *Proceedings of Visualization '92*, pages 436–440, 1992.

[277] N. J. Zabusky, D. Silver, R. Pelz, et al. Visiometrics, juxtaposition and modeling. *Physics Today*, 46(3):24–31, 1993.