

TYPOGRAPHIC PERFORMANCE  
Continuous Design Solutions as Emergent Behaviors of Active Agents

Suguru Ishizaki

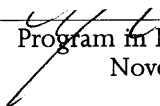
Bachelor of Art and Design, University of Tsukuba, Japan, 1986  
Master of Science in Visual Studies, Massachusetts Institute of Technology, 1989

SUBMITTED TO THE PROGRAM IN MEDIA ARTS AND SCIENCES,  
SCHOOL OF ARCHITECTURE AND PLANNING  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY


February 1996

©Massachusetts Institute of Technology, 1995. All Rights Reserved

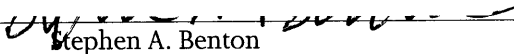
Author

  
Program in Media Arts and Sciences  
November 30, 1995

Certified by

  
William J. Mitchell  
Dean, School of Architecture and Planning  
Massachusetts Institute of Technology

Accepted by

  
Stephen A. Benton  
Chair, Departmental Committee on Graduate Students  
Program in Media Arts and Sciences  
Massachusetts Institute of Technology

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY



FEB 21 1996

LIBRARIES



TYPOGRAPHIC PERFORMANCE  
Continuous Design Solutions as Emergent Behaviors of Active Agents

Suguru Ishizaki

SUBMITTED TO THE PROGRAM IN MEDIA ARTS AND SCIENCES,  
SCHOOL OF ARCHITECTURE AND PLANNING ON NOVEMBER 30, 1995,  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

**Abstract**

This research has been motivated by the lack of models and languages in the graphic design field which address design solutions that continuously change over time. This limitation has become salient in the recent development of computer-based communication media, where design solutions must continuously adapt in response to the dynamic changes both in the information itself and in the goals or intentions of the information recipient.

This dissertation proposes a *Model of Dynamic Design*—a theory of design that provides a model with which the visual designer can think during the course of designing. The proposed model employs a decentralized model of design as a premise, borrows its conceptual model from the performing arts, such as dance and music, and bases its theoretical and technical framework on the field of multiagent systems. In this model, a design solution is considered an emergent behavior of a collection of active design agents, or performers, each of which is responsible for presenting a particular segment of information. The graphical behaviors of design agents are described by their dynamic activities—rather than by the traditional method of fixed attributes—using an abstraction for temporal forms that provides a descriptive scheme for forms that change over time.

This research contributes to the field of communication design in the realm of digital communication: it provides a means of dialogue between a designer and dynamic artifacts, as well as a communication tool between designers; and it suggests an approach for developing computational design solutions.

**Thesis Supervisor: William J. Mitchell**

Dean of the School of Architecture and Planning, Massachusetts Institute of Technology

•

This work was performed at the MIT Media Laboratory. Support for this work was provided in part by ARPA, Alenia Corporation, and NYNEX. The views expressed within do not necessarily reflect the views of the supporting sponsors.



## Doctoral Dissertation Committee

Thesis Advisor

---

William J. Mitchell  
Dean of the School of Architecture and Planning  
Massachusetts Institute of Technology

Thesis Reader

---

Ronald L. MacNeil  
Principal Research Associate  
MIT Media Laboratory

Thesis Reader

---

Mitchel Resnick  
Assistant Professor of Media Arts and Sciences  
Massachusetts Institute of Technology



# Contents

<b>Abstract</b>	3
<b>Doctoral Dissertation Committee</b>	5
<b>Contents</b>	7
<b>Acknowledgements</b>	9
<b>Part I</b>	
<b>1. Introduction</b>	15
1.1. A Scenario	
1.2. Problems and Opportunities	
1.3. A Note on the Title	
1.4. Organization	
<b>2. Approach</b>	23
2.1. Theoretical Perspective	
2.2. Overview of Approach	
2.3. Traditional Graphic Design	
2.4. Summary	
<b>3. A Model of Dynamic Design</b>	37
3.1. Overview	
3.2. Agent's Ability	
3.3. Organization of Agents	
3.4. Life Span of an Agent	
3.5. Other Distributed Models of Design	
3.6. Summary	
<b>4. An Abstraction for Temporal forms</b>	49
4.1. Formal Dimension	
4.2. Phrase	
4.3. Temporal Form	
4.4. Expression	
4.5. Summary	

<b>5. Designing Dynamic Design</b>	<b>55</b>
5.1. Conceptual Framework	
5.2. Methodical Process	
5.3. Computational Description	
5.4. Summary	
<b>Part 2</b>	
<b>6. Case Studies</b>	<b>59</b>
6.1. <i>perForm</i> : Multiagent Design System	
6.2. Dynamic News Display	
6.3. Electronic Mail Display	
6.4. Interactive Poetry	
6.5. Geographic Information Display	
6.6. Expressive Messages	
<b>7. Reflection</b>	<b>97</b>
7.1. Reflections	
7.2. Design Issues	
7.3. Summary	
<b>Part 3</b>	
<b>8. Expressive Design Systems</b>	<b>105</b>
8.1. Design Systems	
8.2. Issues in the Development of Design Systems	
8.3. Review of Previous Design Systems	
8.4. Summary	
<b>9. Conclusion</b>	<b>115</b>
9.1. Contributions	
9.2. Future Directions	
<b>Epilogue</b>	<b>119</b>
<b>Appendix: Agent Description Language</b>	<b>123</b>
<b>References</b>	<b>129</b>



## Acknowledgements

A Number of people have helped me shape the ideas presented in this dissertation.

First, I would like to thank my advisor Bill Mitchell for his advice, inspiration, and encouragement. I am particularly grateful for his aesthetic demands which helped me maintain the balance between intellectual achievement and design excellence in my research. I am also grateful to other members of my dissertation committee. Ron MacNeil has provided me with inspiring arguments and critiques since the beginning of this research. The guidance given by Mitchel Resnick was invaluable in developing my theoretical framework.

I am also indebted to the late Muriel Cooper, my former advisor, who provided me with vision and support, as well as a unique research environment—the *Visible Language Workshop*. It was my privilege to have had the opportunity to study, work, and sometimes argue with her.

I have also been fortunate to have opportunities to interact with other insightful people at MIT's Media Laboratory, including Walter Bender, Stephen Benton, Glorianna Davenport, Henry Lieberman, Tod Machover, Pattie Maes, Nicholas Negroponte, Ken Haase, Whitman Richards, and Patrick Purcell.

I would like to express my special gratitude to Muninder Singh of North Carolina State University, for his unique theory on multiagent systems, as well as for his generous and helpful suggestions.

I have gained much from interactions with fellow graduate students at the Media Laboratory. First, I would like to thank Yin Yin Wong for her relentless arguments, particularly when I risked compromising my ideas, and David Small for his talent for moving things forward. I also thank Louis Weitzman, Robin Kullberg, Ishantha Lokuge, Lisa Strausfeld, Earl Rennison, Xiaoyang Yang, B.C. Krishna, Jeffrey Ventrella, Maia Engeli, Kevin Brooks, Brygg Ullmer, Sara Elo, Janet Cahn, Warren Sack, Sylvain Morgaine, Ming Chen, Laura Robin, Russell Greenlee, each of whom provided me

with useful critiques as well as wonderful friendship. I am also thankful for the undergraduate research assistants at the Visible Language Workshop, including John Grabowski, Jon Levine, and Doug Soo.

Several excellent designers and educators generously offered time for valuable discussions. My particular thanks are due to John Cataldo, Al Gowan, Meg Hicky, of Massachusetts College of Arts; Dan Boyarski of Carnegie Mellon University; Tom Briggs of Theurer Briggs Design; and Yasuyo Iguchi of MIT Press.

I would also like to acknowledge other long-time colleagues and friends who made this long process endurable. John Maeda has always been inspiring in multiple dimensions. Mark Gross taught me how to think critically. Thanks to Nobuyuki Ueda, Tomoyuki Sugiyama, and Takeshi Sunaga for their encouragement and friendship. And, of course, I am thankful to Rié Komuro and my family for their wholehearted support.

I would also like to acknowledge individuals who have supported me during the course of my work: Linda Peterson and Santina Tonelli for their efficient administrative support; Amy Freeman, Nancy Young, and Kristin Hall for their daily assistance; Ben Lowengard and Greg Tucker for their technical support whenever I needed them. My special thanks are due to Kelli Foster for her editorial support in making my writing flow.

I would also like to thank Harlequin Inc. for providing me with Lispworks programming environment; and Ken Appleby of Harlequin for his responsive technical support. Acknowledgements are also due to the sponsors of my research: ARPA, Alenia Corporation, and NYNEX.

**TYPOGRAPHIC PERFORMANCE**  
Continuous Design Solutions as Emergent Behaviors of Active Agents



## Part 1

Part 1 presents the theoretical framework of this research. I first show motivations, and develop the foundation of this research, providing basic concepts and terminology necessary to understand the remainder of this dissertation. I then describe the Model of Dynamic Design, which consists of a multiagent model of design and an abstraction of temporal forms. Finally, I also discuss ways in which the model can be used in designing dynamic design solutions.



## 1. Introduction

*The streams of lights wander around. Rhythmic music echoes and colorful costumes flutter about. Imagine a group of dancers on a stage—maybe a contemporary improvisational performance. Each dancer is moving gracefully, or hopping rapidly—generating a complex body-form over time. Simultaneously, a choreographic harmony is emerging from their coordinating with each other. The experienced and trained dancers are collaboratively achieving the theme of the performance, while dynamically adjusting their movement according to the lighting, music, other dancers, and reactions of the audience. Each performance is unique and cannot be repeated.*

This research proposes a theoretical framework for creating visual design solutions that are as active and dynamic as a dance performance. A design solution, such as a display of on-line news, is considered a performance consisting of a number of active design agents, or performers, each of which is responsible for presenting a particular aspect of information, such as a headline or a news story. The individual design agent is sensitive to changes in its situation—the information itself, the goals of the reader<sup>1</sup>, and the other design agents in the scene. The solution as a whole emerges from the dynamic activities of collaborating agents. This framework is fundamentally different from the traditional view which describes a design solution as a set of fixed declarative statements. The proposed model, a Model of Dynamic Design, suggests a new perspective on the way designers perceive a design solution—as an active entity consisting of a collection of small activities.

1. A reader is a person (or a group of people) who is actively involved in the reading of information presented by the design system. The term “reader” is preferred to the term “user” or “recipient” because of its emphasis on the sense of active communication.

This research has been motivated by the lack of models and languages in the graphic design field which address design solutions that continuously change over time. This limitation has become salient in the recent development of digital communication media, where design solutions must continuously adapt in response to the dynamic changes both in the information itself and in the goals, or intentions, of the reader (hereafter referred to as the reader’s intention). In this section, I begin my

discussion with a brief scenario that illustrates what a dynamic design solution is like. Then I introduce problems and opportunities in the design of digital media that have motivated my research.

### 1.1. A Scenario

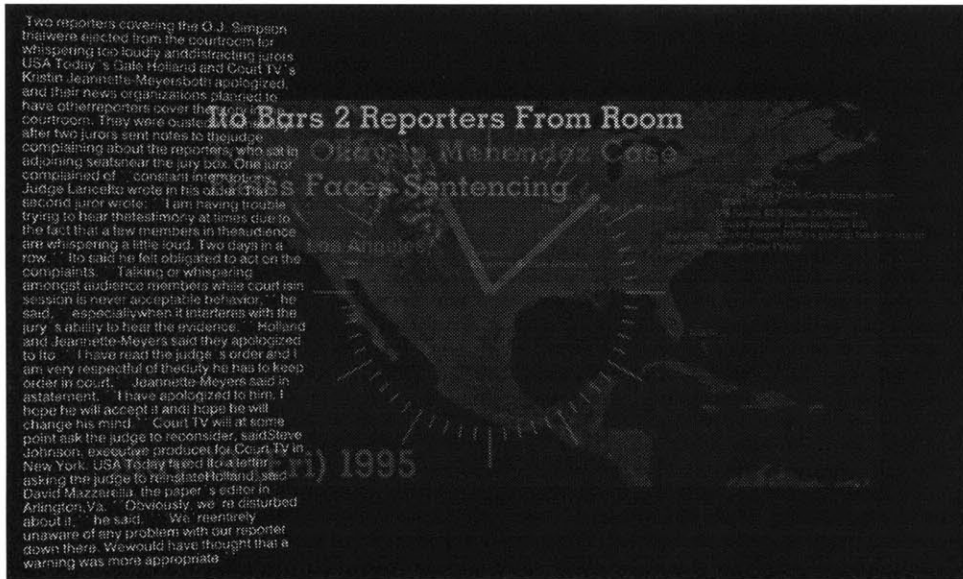
Imagine you are a graphic designer, and just received an assignment from a client. Your task is to design an interface of a news display system that allows readers to access news databases that are constantly updated. News articles arrive at the display system as they are issued, and a reader can view them as they are published. The interests of a user may also vary over time, hence dynamically changing priorities of news articles. For example, a reader may wish to read the top U.S. news, or a reader may be interested in a particular subject matter.

A simple approach would be to use the model of a typical newspaper, or more generally, a page layout. A layout can be updated based on templates whenever some changes occur—*e.g.*, when a new article arrives, or when the user’s interest changes. Or, a static layout can be scrolled up and down within a window. But, is it simply a problem of re-arranging a layout? Can you just flow a page layout in a window? I suggest that there are deeper fundamental differences between the traditional and digital design problems.

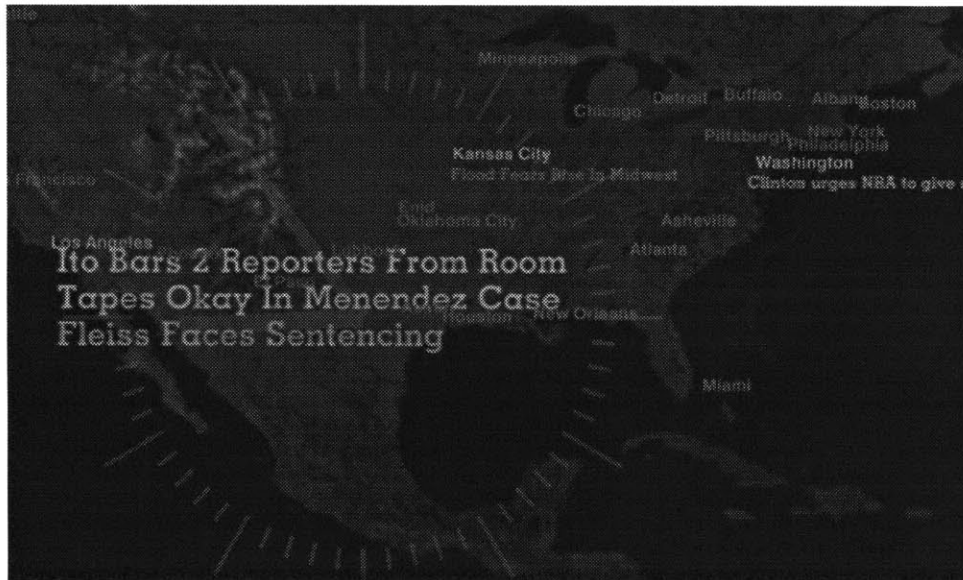
What you wish to create is a visual design solution that can reflect the dynamic changes in both reader’s intention and the databases. But, since information is only available at run-time, and locally to a reader, attributes of design elements, such as typeface and color, cannot be directly adjusted according to a particular set of information. Furthermore, since the changes in context are continuous, even if you were at the reader’s workstation, it is simply impossible for you to solve a design problem manually by hand. In other words, you must design a way of designing, or a design process, in order to dynamically solve this continuous problem over time. On the other hand, since the display is capable of presenting temporal expressions (*e.g.*, animation), you are not limited to the traditional notion of static design solutions; rather, you are given the opportunities to use temporal forms in communication. For example, you may use a repetitive motion (*e.g.*, vibration) as a “label” that signifies an important news stories, in the same way you would use a hue (*e.g.*, red) for the same purpose in the design of a news magazine.

This research provides a designer with a unique scheme for describing this type of dynamic design solution, by viewing it as a continuous and active entity—like a dance performance. An example of a dynamic design solution is shown in **Figure 1.1**. These two scenes are from an experimental on-line news display presenting news articles based on their location. The design solution of this news display is considered an emergent expression created by the dynamic activities of design agents (performers) that are responsible for presenting headlines, news stories, place-names, a clock, a date text, and a map, each of which has its behavior specified by a designer. For example: a headline agent left-aligns its text to its placename when the news story is reported; and when a new headline is issued at the same location it left-aligns its text under the new headline. The Headline Agents also changes the





(a)



(b)

**Figure 1.1.** Screen snapshots of Dynamic News Display. (a) A scene while a user is reading a news story, after a headline was clicked. Notice that other elements become all highly translucent. (b) Close up of the news display. Headline agents at Los Angeles glow their text size bigger, when a user places the cursor over its associated placename.

translucency of text based on its age. The placename Agent changes its text color when there is a headline associated with its location. When a news story is being read by a reader, other design agents make their visual representations highly translucent so that they do not distract the reader's reading (**Figure 1.1.a**).

A designer's task when working with this proposed model is to anticipate potential changes in the context and to specify the expressive forms that design agents should perform according to their immediate situations. In the course of creating a final

dynamic solution, a designer would have to rehearse agents' behaviors iteratively by simulating various dynamic design situations. An analogy to the designer's role can be made with a director of an improvisational dance performance. The director selects dancers with the desired expressive skills (body forms) and coaches their performance (situated acting) through a rigorous rehearsal.

The next section generalizes the nature of design problems in digital communication addressed in this scenario, and provides the motivation for this research.

## 1.2. Problems and Opportunities

Traditional graphic design encapsulates information into fixed media, such as print or film, so that the message can be distributed or stored. In digital communication, design problems become more dynamic as the media become more interactive and include more temporal-varying information. It is therefore important to create a design that continuously adapts in response to the dynamic changes in context—that is, changes in information and the reader's intention. I argue that graphic design as a field limits its own contributions to traditional methodologies, despite increasing efforts from within the field to improve the visual design of electronic media. This lack of models and languages prevents us from exploring design solutions that are unique to digital communication.

This argument is not a criticism of existing design solutions; rather, it is intended to suggest the opportunity to extend the repertoire of visual design with the aid of new communication technologies. One approach to exploiting this opportunity is “to practice.” We all know that the more we experience particular design problems, the better we become at solving them. Nonetheless, although persistent practice is effective in order to improve the design of new media, it can be constrained to the framework and conventions developed in previous experience with the traditional media. Thus a fundamental understanding of the nature of digital communication, and critical analysis of the existing methods are also necessary. This is the purpose of the proposed research: it aims to understand the nature of digital communication, and to develop a new framework that utilizes its unique qualities.

The problem on which I have focused in this research is identified by two unique characteristics of digital communication: dynamic change in context and the capability of temporal presentation (**Figure 1.2**).

First, dynamic changes in context (i.e., information and the reader's intention), raises a new problem in the design of digital media. For example, on-line information systems, such as news databases and traffic information systems, are updated as information changes. The design in such a medium must reflect the dynamic changes in information over time. Interactive media provide readers with personalized access to information. The pace and order of reading and the amount and selection of information are determined by the intention of the individual reader. More recently, digital media have become more personalized, often using artificial intelligence techniques. For example, your news system may learn your interest over time, and generate a special edition for you every day. In such communication media, each

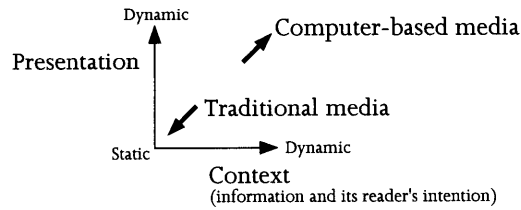


Figure 1.2. Two characteristics unique to computer-based communication raise new problems in design.

individual design problem can be unique and can change continuously over time. Consequently, in digital media, a designer often finds it impossible to design a solution to a particular problem and instead must design **a way of designing**, or a process, in the form of a computer program that can generate design solutions at run-time.

But, what is a way of designing, or a process? What kind of form does this process take? In this dissertation, **a way of designing** is an explicit description of how specific design elements, unknown at the time of designing, change over time according to changes in the immediate context. A goal of this research is to propose both a conceptual model and a descriptive language that designers can use in the course of creating dynamic design solutions.

The second characteristic of digital media, the capability of temporal presentation, introduces new challenges and opportunities in design. Designers are no longer limited to **fixed forms** for communicating information. For example, a gradual color shift can be used to convey complex emotional quality; or a speed of repetitive movement can be used to indicate a quantity. Traditional graphic design has been involved in temporal presentation media such as television and film, and it has developed some conventional tools such as storyboard and notational systems (e.g., [Eisner 1985][Hiebert 1992][Nishimura & Sato 1985]). However, there has been only a few studies in the field that address languages that allow designers to discuss visual forms presented over time (e.g., [Hiebert 1992][Bork 1983]).

What I mean by models and languages can be exemplified by the Munsell color model used in traditional design and design education. The following statement presents the role of such knowledge [Munsell 1929]:

*The vast majority of persons are lacking in adequate color knowledge and must seek for themselves that information relating to its use and appreciation which is the inborn gift of the favored few. But natural taste or aptitude should not be mistaken for organized fundamental knowledge; the naturally gifted cannot accurately communicate their ideas of color without a common language of color, for intelligent discussion of the subject requires mutual understanding of the terms used.*

The Munsell model has provided designers with a three dimensional model of color—hue, value, and chroma—and a vocabulary set that allows them to compose rich color harmonies and analyze complex interactions of color. In this dissertation, I propose such a common language of temporal forms.

Recent developments in digital media have highlighted the lack of models and languages in the design field that address the dynamic change of context in conjunction with temporal presentation. These problems and opportunities have raised various questions, which need to be answered: How can a designer conceptualize and describe a solution that can be interrupted by some immediate changes in information and in the reader's intention? How can a designer describe a design that contains an undetermined number of changing design elements? How can a designer contemplate design problems in digital communication as continuous problems, rather than a series of discrete problems?

Design problems in digital communication demand new types of design solutions, as well as methods and conventions to support the designer. I argue that in order to extend the fixed nature of the traditional design into a more continuous and responsive one, it is valuable to develop a theory of design—a model along with a language—that provides designers with a conceptual framework during the course of solving design problems. I also emphasize that the model must be useful in the development of computer systems that can represent and generate continuous and responsive design solutions, since such a design is only possible with computational support at the present time. However, I must note that the primary purpose of this research is not to investigate the software architecture; rather, it is to develop a new theoretical framework in order to advance the field of graphic design in the realm of digital communication by providing a means of dialogue between designer and dynamic artifact, as well as a communication tool between designers.

Throughout this dissertation, I shall use the term *dynamic design* to define a sub-field of graphic design that is concerned with the type of design described above, where a design problem is considered a continuous stream, a design solution responds to changes in context using temporal forms; and the designer's role is to design a process of designing, rather than designing a particular artifact. In this proposal, the term will also be used to designate design solutions in dynamic design. Consequently, the theoretical model proposed in this dissertation is called *a Model of Dynamic Design*.

### 1.3. A Note on the Title

Typography concerns the visual treatment of language and has been recognized as the most essential component of visual communication. The term “typography” originally referred to printing technology; thus, typography is closely related to the paper medium, which is static and discrete. Although many visual designers have recently explored the use of type in the more dynamic media, such as television and film, there has not been any attempt to create typographic expressions as responsive as those of the performing arts. The challenge of my research has been to liberate typography from the fixed and discrete, to the active and continuous; hence *Typographic Performance*.

The term *typography* in the graphic design field often encompasses in its meaning the design of text in relation to other visual forms, such as photographs and diagrams. Thus, although my research mainly focuses on textual information, it will not be strictly limited to pure typography.

#### **1.4. Organization**

The remainder of Part 1 presents the theoretical framework for the research. In Chapter 2, I outline my approach to the problem by introducing three areas of research—cognitive models of designing, improvisational performance, and multi-agent systems—that constitute this research. The roots of the approach in traditional graphic design are also presented in this chapter. Chapter 3 introduces the multi-agent model of design. Chapter 4 presents an abstraction of temporal forms as a means of describing design agents' formal behavior. Following the theoretical framework, in Chapter 5, I present how I envision the model can be used in practice.

Part 2 presents a series of dynamic design solutions that illustrate the use of the proposed model in concrete contexts and discusses design issues associated with it. Chapter 6 describes five different design projects in the following domains: on-line news, electronic mail, poetry, geographic information, and expressive messages. Chapter 7 reflects upon the design experiments presented in Chapter 6, and discusses issues related to the use of the model in concrete design problems.

Part 3 places the proposed model in the larger scheme of digital communication. Chapter 8 suggests a set of criteria for the development of generative design systems and situates the Model of Dynamic Design within that picture. Research in the generative design systems is also reviewed. Finally, Chapter 9 summarizes the contribution of this dissertation, and introduces the future research directions that build upon this research.



## 2. Approach

This chapter outlines my response to the problems and opportunities in the design of computer-based communication introduced in the previous chapter. I begin by introducing the theoretical perspective of my research. Since this research has been conducted in an interdisciplinary setting, it would be useful to first identify what I mean by theoretical framework in design. Then, I present my approach by introducing the three fundamental constituents of the research that have supported the development of the proposed model: multiplicity of design knowledge, improvisational performance, and multiagent systems. Finally, I present my approach from the perspective of traditional graphic design.

### 2.1. Theoretical Perspective

This section clarifies the theoretical position on which this research stands. The roles and characteristics of the Model of Dynamic Design are presented within the larger field of design research. In Section 2.1.1, theoretical research in the field of design are summarized. Section 2.1.2 clarifies the role and type of the proposed model.

#### 2.1.1. Types of Theories in Design

Research in design theory generally is concerned with the improvement of designed artifacts. Interest in developing a theoretical framework in design began in the 1950s because of the increasing complexity of design problems. Over almost four decades, researchers have examined issues in many areas of design, including architecture, engineering, and product design [Cross 1984].

Kaplan, a philosopher, states: “A theory is a way of making sense of a disturbing situation so as to allow us most effectively to bring to bear our repertoire of habits, and even more importantly to modify or discard them altogether, replacing them by new ones as the situation demands” [Kaplan 1964. p.295]. Although this statement was written in the context of behavioral science, it clearly articulates the role of theory in the design field. Design theories are formulated to provide the designer with a means of analyzing and understanding design problems, predicting future design operations, and logically developing and evaluating design solutions [Lang 1987].

The original motivation in developing design theories was that designers are not perfectly capable of dealing with complex design problems; hence systematic methods to support designers are needed to reduce errors and improve productivity. These systematic methods are usually regarded as a prescriptive model, and they often consist of a set of conceptual devices that help designers organize a complex problem (e.g., the diagramming method) as well as a process that they can follow to gradually reach a solution. In addition, they often suggest how the design solutions are evaluated. The methodical exploration of design solutions proposed by Swiss designers is an example of prescriptive theory (e.g., [Müller-Brockmann 1988][Gerstner 1968]).

Another view of design research focuses on how professional designers design, or think, during the course of designing. The motivation for understanding the cognitive process of designing stems, in part, from failures of early prescriptive models that did not fit the mind set of the practitioners. This cognitive model of design plays an important role in the development of a prescriptive model of design. (e.g., [Schön 1983][Akin 1986][Schön & Wiggins 1992]) Descriptive models are also important in the development of computational tools for designers. The lack of understanding of the nature of a designer's thought process makes it difficult to build a useful tool that can support designers. This type of theory is also called positive, or objective theory [Lang 1987].

Another type of positive theory concerns the nature of design problems. This type of research has been motivated by some of the difficulties in applying early prescriptive models to practical problems. Without understanding the nature of problems that the designer must confront, neither prescriptive nor objective theory can be developed (e.g., [Simon 1973][Rittel & Webber 1984][Buchanan 1992]).

There have always been theoretical efforts to define good design. These theories are called normative theory, as opposed to positive or objective theory. Normative theory describes "what ought to be." Design principles and guidelines are examples of the normative theory. For example, the famous statement "form follows function" is a normative framework. A conventional view of visual design that supports an economy of comprehension (i.e., a good design solution must communicate quickly) can also be seen as a normative position we often blindly accept. Prescriptive theories are often considered a subclass of normative theory, since their purpose is to create good design, and there is no strong foundation in positive science. However, normative and prescriptive theories must be distinguished based on what they offer: normative theories provide evaluative criteria for judging good design, whereas prescriptive theory provide processes (methods) for generating and evaluating solutions.

More recently, a few researchers have begun to propose a generative theory of design, which is a variation of prescriptive theory, it intends to provide designers systematic means of generating better design solutions. However, these theories do not provide an evaluative scheme, but rather a precise language for generating and describing design solutions. For example, shape grammar provides a descriptive framework for creating a formal solutions as a consequence of applying a number



of production rules (i.e., if-then rules) [Stiny 1980][Mitchell 1990][Knight 1989]. Another theoretical framework, postulated by Gross, considers designing a process of exploring constraints [Gross 1986], where constraints are explicitly defined relationships (physical or semantic) among elements of design imposed by a designer. This work proposes both a precise description, as well as a process that designers can follow. This type of research is strongly influenced by computation, and uses computational tools as an integral part of the framework that supports designing.

### **2.1.2. What is and isn't, the Model of Dynamic Design**

The Model of Dynamic Design proposed in this dissertation can be seen as a prescriptive and generative theory. Its purpose is to provide a conceptual framework that imposes a structure on a dynamic design problem and its solution, along with a methodical process that utilizes that framework in the course of solving a design problem. In particular, the form and role of the proposed theory have been influenced by generative theories such as shape grammars and constraint models. Since, it is neither intended to describe the designer's cognitive process, nor to provide a general explanation of dynamic design problems, it is not a positive theory. Nor do I present the Model of Dynamic Design as a normative theory, although it reflects my hypothetical position encouraging dynamic design solutions as a better approach to the design problems in digital communication. That is, it does not provide principles for implementing "good" design solutions. Rather it provides an *analytical device* the designer can use to discuss and discover "good" design solutions.

Much of the research in design theory has addressed a specific field of design, such as mechanical engineering or architecture. However, there has been only a limited amount of research in the field of graphic design. I speculate that this is because most of graphic design problems seem far less complex than those of other fields, and graphic design errors are less obvious and less harmful. As computer-based media have become more dynamic and interactive, however, design problems have become increasingly complex. Furthermore, continuous design solutions in digital communication can not be generated by hand, since design problems are continuously re-defined over time. This research aims to build a theoretical framework to overcome problems in the design of computer-based media, as well as to take advantage of opportunities in creating design solutions not possible in static media.

## **2.2. Overview of Approach**

This section presents the foundation for my approach. First, I introduce cognitive models of design that emphasize the multiplicity of knowledge involved in designing, which has been a motivational basis for the multiagent model. Then, I introduce studies in improvisational performance, which provide the proposed theory with a conceptual framework, as well as a vocabulary set to describe and understand forms that change over time. Finally, I introduce the research in multiagent systems that provides a theoretical and computational framework.

### **2.2.1. Multiplicity of knowledge in Designing**

A dynamic design solution is like a designer's act of designing. It is a continuous process of manipulating and reflecting upon tentative solutions. The only difference

is that a dynamic solution must keep designing, while a traditional design process terminates when an optimal fixed solution is found. Consequently, this research benefits from the study of how designers act during the course of designing. Among a number of cognitive models of design that attempt to understand the process of designing ([Cross 1984][Rowe 1987]), ones that have most influenced the Model of Dynamic Design are those incorporate multiplicity of design knowledge, and multiplicity of seeing and focus of attention.

Schön and Wiggins characterize the process of design as multiple ways of seeing [Schön & Wiggins 1992]. Each way of seeing is responsible for a particular aspect of design, such as structure, color, or finance in their domain of architecture. They postulate that ways of seeing are the basic units of the design process and that they are developed and discovered through practice. Seeing, by their definition, includes judging, evaluating and interpreting, although criteria may or may not be explicit. They also suggest that there may be conflicts among different domains of seeing.

Bucciarelli and Schön also propose an interesting argument which states that cooperative design problem solving among a team of people involved in a design process and multiple ways of seeing in the process of a single designer are strongly similar [Bucciarelli & Schön 1992]. As different experts in a design team may use different languages and representations, different ways of seeing by an individual designer may involve different reasoning processes and representations. In addition, as there are often disagreements among members of a design team, Bucciarelli and Schön suggest that there can be discrepancies among ways of seeing. Reconciling these discrepancies in a design team and reconciling an individual's various ways of seeing both take the form of a dialogue.

Similarly, Mitchell suggests the importance of having many different representation schemes in ideal design support systems [Mitchell 1986]. He suggests a network of multiple design systems, in which each system has its own interpretation mechanism and language, as a way of realizing a computer system that allows a designer to fruitfully explore design solutions. Stiny also suggests that there are various specialist domains in design thinking, each of which has its own algebra and language [Stiny 1991]. He also postulates the possibility of the parallel computation of algebras in multiple domains. Mitchell's multiple representations and Stiny's multiple specialist domains can be seen as formalizations of the idea of multiple ways of seeing proposed by Schön and Wiggins.

While these theories mainly discuss ways of seeing as expertise in design, it is clear that the scale of seeing varies based on the granularity of information to which a designer must attend in the course of designing. For instance, Whitefield and Warren, in their blackboard model of designing, identified three levels of design knowledge—unit, item, and detail—based on their experiments in the domain of engineering design [Whitefield & Warren 1989]. In visual design, roughly four different levels of focus can be identified (Figure 2.1). The most detailed level of focus is the physical attributes, such as location and color (**a**). The second level is the design element, which is often a meaningful unit of information, such as a headline

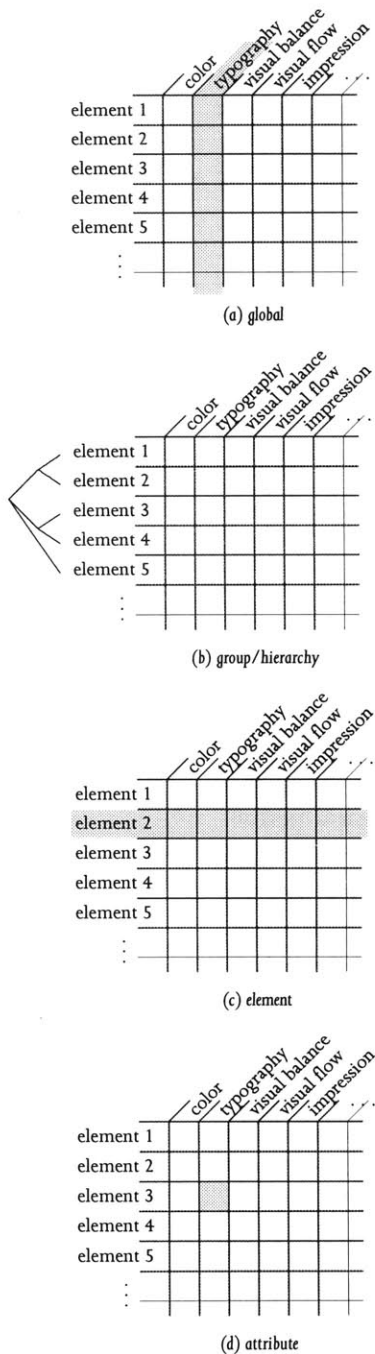


Figure 2.1. A schematic diagram of various focus levels that take place in the act of designing.

or a photograph (c). The third level is a group and a hierarchy of design elements, such as a news article (b). The most coarse level often covers attention to the entire frame of the design. This global level of focus includes color harmony, spatial balance, and rhythm (a). These four levels are equally important in generating design solutions.

In summary, a design process seems to involve complex interactions among a designer's multiple ways of seeing in various types of expertise, as well as multiple focuses of attention in different levels of detail. These findings are useful for developing a descriptive model for dynamic design solutions in two ways. First, since the Model of Dynamic Design is for designers to think with, it is natural to develop a model considering design processes. Second, a distributed model, which is suggested by the multiplicity of design knowledge, is often considered a better approach to describe complex behavior. Bailey states that "there are things one can say in the presentational (parallel) form that simply cannot be said in the discursive (sequential) form: "Too many relations within relations cannot be projected into discursive form" [Bailey 1992]. Complex relations are often easier to represent in a parallel form. I postulate that the complexity of a design solution (e.g., color harmony, visual impression) comes from the lateral interaction among various aspects of the design: hence a design solution can be well-represented in a parallel, or distributed, fashion.

If we accept this premise, the remaining question is to define the components—and their relationships—of this distributed Model of Dynamic Design. In this research, I propose that a design element is a meaningful unit, or component, around which the design solution can be described (Figure 2.2). There are two conditions that support this hypothesis. First, in graphic design, it is generally important to identify meaningful units of information and their roles in communicating an entire message. Then, an appropriate form must be given to each information element. In other words, designing is an act of determining the form of an individual information element, in relation to other elements. Second, when we look at design from a perspective of designing—physical manipulations that take place in

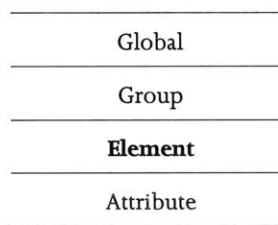


Figure 2.2. The Model of Dynamic Design uses the design element as its basic unit of description.

exploring design solutions—it is clear that the design element is central to the other three levels. It is the level in which we can precisely articulate in terms of physical properties. Rather complex design concepts in the higher levels of seeing, such as balance and color harmony, are hardly articulated, and must be appropriated by a set of physical properties at the level of information element. The lowest level of attribute is too microscopic to be useful for the description of meaningful design solutions.

The above discussion has led to the model of design which views design solutions as an emergent behavior of active design agents. However, as I argued in the introduction, the existing models and languages of traditional design are constrained by the inherently static nature of traditional media. The next section presents a model of improvisational performance that adds a conceptual layer of time to this distributed model of design.

### 2.2.2. Improvisational Performance

In order to describe this multiagent model of Dynamic Design, I have drawn an analogy from the nature of performing arts, such as dance and music, to that of dynamic design. In particular, I learned from the model of improvisational performance, which is inherently responsive. In the proposed model, a design solution is considered a performance consisting of a number of design agents (performers), each of which is responsible for its own role in the design. A design solution as a whole emerges from the collaboration of design agents, just as a musical performance is a harmony generated by a group of performers. The individual design agent is also sensitive to the change in its immediate context (information and the reader's intention) and can respond to it accordingly, just as an improvisational performance involves performers' spontaneous expression in response to the immediate situation, *i.e.*, other players or music. A designer's role is viewed as that of a director instructing design agents' behaviors prior to their final performance—a continuous design solution.

Improvisation often involves performers' spontaneous expressions in response to their immediate situation. The history of improvisation goes back to the figure of the shaman (who existed before the priest, poet, and actors) whose performances were strongly influenced by an immediate situation, *i.e.*, audience and environment [Frost and Yarrow 1989]. Despite the long history of improvisation, the study of improvisation did not begin until recently [Pressing 1984], and few computer-based systems exist (*e.g.*, [Rowe 1993]). In the following paragraphs, I will first look at the nature of improvisation based on the recent studies by Blom and Chaplin in dance [Blom & Chaplin 1988] and by Pressing in music [Pressing 1984]. The influence of the improvisation on the proposed model will be discussed.

In dance improvisation, Blom and Chaplin suggest that performers tend to use “self-contained units of movement,” or phrases, which constitute a movement. Phrases are remembered by a performer in the form of muscle memory through training. The authors also introduce an abstract concept of *form*, which is composed of phrases. During a performance, a form can be either composed in response to a particular situation or retrieved from the remembered set of forms. For instance, in

dance, a simple movement of a hand can be a phrase and can be combined with other phrases, such as arm and body movement, to become an expressive form. Similarly in music, a sequence of changes in pitch can be a phrase and can constitute a melody, or musical form.

Based on an exhaustive survey of various improvisational performances in history, Pressing postulates general characteristics of improvisation from a psychological perspective [Pressing 1984]. One of the unique characteristics he observes is that the improvisational performance depends highly on a performer's cognitive processing capacity; thus improvising is always "the result of combining previously learned gestures, movement patterns, or concepts in a novel relationship or context" [Pressing 1984]. Pressing views improvisation from the perspective of what is called "skilled performance" in psychological research. According to his model, a performer acquires and refines a number of small motor programs, or units of action, through practice. These units of action are similar to phrases in dance described by Blom and Chaplin.

This very nature of improvisation can be characterized as "training specific" performance, as opposed to "composition specific" performance [Pressing 1984]. Since a performer's cognitive processing during a real-time performance is mostly devoted to the global aspect of performance, the performer must mostly depend on skills already learned through training. These skills enable a performer to process other aspects of the performance, such as sensing and understanding his/her immediate surrounding, and thinking about the overall theme of the performance.

When an improvisation is performed by a group of performers, there must be a common agreement about an overall theme of the performance. Usually there is somebody, such as a director or leader, who determines the theme of the performance. This theme and coordination among performers are usually acquired through thorough rehearsing.

The proposed theory borrows two distinct characteristics of improvisational performances. First, improvisation intrinsically addresses performers responses to the spontaneous change of context during a performance while coordinating with other performers. In dynamic design, design agents' prompt responses to the change in the immediate context while maintaining the overall design is also important. Consequently, the analogy is made between the Model of Dynamic Design and improvisational performance in order to explain the framework of the multiagent model. Second, the field of performing arts provides a model and a language that describe the presentation of the active design element, i.e., performers. I have argued that the graphic design field lacks models and languages addressing the temporal use of form. A set of vocabularies and concepts in the performing arts have been adopted and formalized as an abstraction for temporal forms, which will be described in Chapter 4. In addition, an analogy between the role of designers and directors is made. A designer's role in the proposed framework is to create an organization of design agents and "train" them so that they can solve dynamic design problems.

### 2.2.3. Multiagent Systems

While improvisational performance serves as a conceptual framework, research in the field of multiagent systems (MA) provides a theoretical and technical foundation to this research. MA research is a branch of Distributed Artificial Intelligence, and its general goal is to develop theories and computational techniques that are used to represent a group intelligence [Bond & Gasser 1988]. It focuses on how a collection

of intelligent modules (agents or nodes) can be coordinated to achieve one or more goals, and is concerned with the design (capability and representation) of individual agents and their communications<sup>2</sup>. The framework in MA has contributed to the formulation of the proposed model that views a design solution as emergent behavior of a collection of active design agents. MA research also has provided a technical basis for the implementation of the software developed in this research, which is used to represent and generate dynamic design solutions.

2. Although MA, Distributed Problem Solving (DPS), and Parallel AI are closely related, there are subtle distinctions between the three. DPS is also a branch of Distributed AI, which largely shares its research interest with MA systems. However, DPS focuses mainly on how a problem can be distributed to a collection of intelligent problem-solving processes. It is concerned with the decomposition of a problem and the allocation of tasks and resources. Parallel AI is mainly concerned with improving the performance of AI systems and concentrates on parallel algorithms and computer architecture. [Bond & Gasser 1988]

Much of the research in MA borrows theoretical foundations from the principles of various social organizations, such as corporations, markets, and sports teams (e.g., [Fox 1981][Malone 1987]). An extreme type of organization uses only one hierarchy (e.g., corporation). A problem is received by a top-level manager agent who divides the problem into sub-problems and distributes them to lower-level agents. These agents who receive a sub-problem may either solve the problem by themselves, or distribute the problem further to its lower-level agents.

Another extreme kind of organization is a lateral structure where no agents are in control of other agents (e.g., market): a group of agents communicate and cooperate with each other to solve one or more problems. But, in addition to these to extremes, there is also a range of intermediate structures between the centralized and the lateral organizations. One method is to use multiple groups of hierarchical agents (e.g., multiple corporations). Another method is to make a system such that a particular agent may dynamically become a leader agent depending on the context (e.g., sports).

A major disadvantage of having centralized agents in general is that they can easily create a bottleneck. Whenever a higher-level agent breaks down, the part of the system below that agent is lost. In particular, when a centralized agent is a manager that organizes other agents with respect to their global functions (e.g., a manager responsible for color harmony), the entire solution may be damaged. Despite this problem, the centralized approach has some advantages. First, a hierarchical organization with manager agents makes the communication among the agents efficient. Since the organizational structure pre-defines the roles and abilities of agents, wasteful communication is minimized. Second, centralized control is useful if the solutions distributed among multiple agents must be collected and integrated.

Lateral organizations generally do not have the problem of one agent causing a bottleneck. However, proper coordination often requires sensing mechanism and an

iterative exchange of information. Thus, communication and sensing cost is usually high in the lateral organization. Nonetheless, this problem can be partially solved by improving the local capability of individual agents (hence reducing the number of communications), although this method potentially increases each agent's computational cost. The lack of global control may also force agents draw into an infinite loop, or an oscillating state. A possible solution to this problem is a lateral organization that occasionally uses centralized agents—for example, a meta-level agent who observes critical situations. Another approach similar to the use of meta-agents is the use of a few specialized agents. Sycara has proposed a lateral organization and decentralized control with a specialized agent called a “persuader” that helps non-cooperative agents to negotiate a compromise using case-based reasoning [Sycara 1987]. Another possibility is to design an organization such that some agents can temporarily play the role of a manager. In general, the agent with the most valuable information for the situation can become a manager (*e.g.*, [Steeb et al. 1981]). Finally, a lateral organization is suitable when the solution distributed among multiple agents does not need to be explicitly collected to form a global solution.

The proposed model draws an analogy between improvisational performance and the visual design of dynamic media. From the perspective of MA, an improvisational performance can be seen as an emergent behavior of a group of agents (performers) organized in a relatively decentralized fashion, as opposed to a group of deliberate agents with centralized control. Thus, in the proposed model, I view a dynamic design solution as an emergent expression created by a team of design agents, each of which has highly responsive “skills” enabling them to respond to the changing context. In other words, a multiagent design solution does not explicitly collect individual efforts to build a solution. Like a dance performance, the independent and decentralized activities of design agents as a whole directly become a solution. Since there is no meta-agent that manages other design agents, and responsibilities are distributed among individual agents, there is a low risk of having a large failure caused by a meta-agent containing a descriptive mistake. On the other hand, the use of a decentralized system can become problematic because it lacks centralized control to avoid oscillations. I postulate, however, that it is possible to avoid oscillating situations by carefully designing agents. The issue of careful design is further discussed in Chapter 3.

The Model of Dynamic Design suggests the occasional use of a manager agent—like a spontaneous lead dancer in an improvisation. A leader agent in a design solution enables the explicit coordination among agents. In the dynamic news display, for example, The Headline Agent is responsible for instructing its associated news story agent to appear or disappear; the Placename Agent informs its associated headlines about how a reader is interacting with its text on the display. A more complex example of explicit coordination is illustrated in the case study for an email display system.

Among the various possible agent representations, such as rule-based or game theoretic approaches ([Bond & Gasser 1989]), the proposed model adopts reactive agents as a basic descriptive scheme in order to satisfy the relatively simple and

concrete description of design solutions, as well as to describe the responsive nature of dynamic solutions. There are two arguments for using reactive agents as constituents for the model. First, the relatively simple design for reactive agents provides visual designers with a plain method of describing their design specifications. This method also allows designers to easily alternate specifications, verify solutions, and experiment with multiple ideas. Second, it is often the case that designers can not articulate their rules, since many decisions are made with perceptual feedback such as balance and rhythm. I hypothesize that using the reactive approach, this type of design knowledge can be embedded into a set of reactive patterns.

In particular, I have found Hickman and Shiels' model of cooperative/situated agents [Hickman & Shiels 1991] and Singh's model of group ability [Singh 1991, 1994] to be the most appropriate for my project. Hickman and Shiels experimented with the cooperative/situated agents as an extension of single situated agents proposed by other researchers (e.g., [Agre & Chapman 1990][Brooks 1991][Steels 1990]). They built a system where agents perform the task of TV assembly. The activities of agents are simply described as a set of situation-action rules (e.g., If situation  $S_i$  is observed, then perform action  $A_j$ ). "Situation" here includes the activities of other agents. Hickman and Shiels' experimentation has shown that relatively simple reactive agents can cooperate to achieve a common goal using the mutual intelligibility of actions. Their result has also shown that cooperative/situated agents can be flexible, robust and fault-tolerant. However, their system involves only two agents that have the same ability: they are not allowed to communicate, they simply observe the other agent. It is easy to imagine that increasing the number of agents and having heterogeneous agents makes coordination between agents more difficult.

Singh has proposed a model of a group of situated heterogeneous agents that can achieve a common goal. An analogy is drawn from a team of football players. A group of agents as a team share common strategies (e.g., offense formations). However, there is no explicit representation of global strategies; rather, the common strategies are represented as a collection of individual agents' strategies. According to Singh's theory, given a particular goal, a group is said to have a common strategy if a set of individual agents' strategies can achieve the goal. In this way, the group knows how to achieve a goal. Singh's representation of agents is essentially same as Hickman and Shiels'. However, it provides a higher level abstraction for implementing more complex and larger scale multiagent systems. This idea of strategy, or know-how, is a useful conceptual scheme for the designer of agents; hence it is adopted for the Model of Dynamic Design. One of the known problem of reactive agents is their ineptitude for long term planning. In other words, agents' activities tend to be short-sighted. This problem raises a question of how a complex visual presentation can be created if agents are purely reactive and short-sighted. The abstraction of group strategy enables the creation of solutions that are not purely reactive. A sequence of coordinated presentations can be generated by a team of agents that are carefully described—or "trained."



### 2.3. Traditional Graphic Design

This research is fundamentally rooted in the field of traditional graphic design. This section outlines how the proposed framework relates to the traditional design, as well as how it can extend its repertoire in the domain of dynamic design.

#### 2.3.1. Systematic Approaches

In traditional design, the designer usually solves particular problems, whereas in dynamic design, I have described that the designer must find a process of design that can be encoded in the form of a computer program. This design process can be called *meta-design* in that it is a process of generalizing and identifying how the designer would design if doing so directly.

The idea of meta-design is not unique to dynamic design. There have been a number of attempts in graphic design to use a systematic approach to solve design problems. The earliest attempt at systematic graphic design was in the Hochschule für Gestaltung ULM in Germany [Lindinger 1991] during the mid-1950s. The school postulated scientific methods and critical theories for the design of visual communication media, including print, film, and television (e.g., [Bonsiepe 1968] and Figure 2.3). However, since HfG ULM emphasized the application of theories developed in traditional science, which are often difficult to understand yet too simplistic for the rich quality of visual design, their approach has not been widely accepted by practitioners. Subsequently, Swiss designers, such as Gerstner and Müller-Brockman, developed a systematic approach which was more suitable to design practice (e.g., [Gerstner 1968][Müller-Brockman 1988]). They developed a practical method that uses visual structures (such as grids) and rules as guides for solving design problems (e.g., Figure 2.4). While HfG ULM emphasized semantics of visual communication (in response to the Bauhaus approach which focused on syntactic aspects), the Swiss designers focused more on the issues of form. Works by Swiss designers have strongly influenced the graphic design field. Not only was the systematic approach taken by the designer to guide the design process, it also became an object of design that is often called a design system. The design system

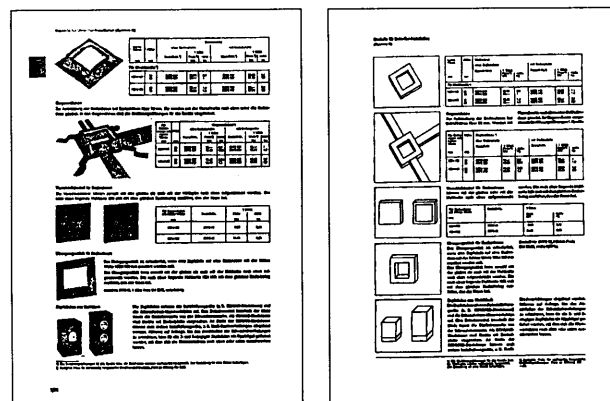


Figure 2.3. Bonsiepe applied Shannon's complexity formula to mathematically measure the "order" of a layout [Bonsiepe 1968]. The layout on the left is the original and the one on the right is the layout with a higher order.



Figure 2.4. A series of layouts using a 20 by 20 grid structure [Müller-Brockmann 1988].

is a collection of rules and guidelines for creating a particular series of design solutions that must have the same style. Corporate visual identity manuals and publication design formats are examples of this design system.

The nature of the systematic approach in traditional graphic design is similar to that of the proposed framework. Both require designers to be more conscious of their decision making, and to describe a general scheme of design instead of particulars. However, there is an essential difference in the way final visual solutions are generated. In a traditional systematic approach, the designer's role is to define structures and rules for a particular domain of design problems and to follow them in order to generate final solutions. In the proposed framework, designers must describe solutions in a more precise manner, in such a way that they can be "followed" by a computer program. For example, in the news display scenario, a designer must carefully and precisely describe how headlines, articles, and photographs should be displayed over time. Then, the computer program performs the task of designing at run time based on the specification, or design strategy, given by the designer. Consequently, it is clear that the quality of visual communication depends upon the expressive capability of the model and the language.

Another essential difference between traditional and dynamic design is that the design solutions for dynamic design are likely to be continuous, responding to dynamic changes of context. In traditional systematic design, design rules are typically applied to a variety of problems discretely defined; while in dynamic design, there is usually a continual design problem that must be continuously addressed. For example, an electronic news system may receive a new set of news articles anytime a reader is interacting with the system. Also, even without any new "event," time should be reflected in the behavior of a design solution. In such a design problem, a solution involves continuous visual forms expressed in response to the dynamic changes of context—a *new concept in graphic design*.

### 2.3.2. Visual Languages Study

In traditional visual communication design, there have been a number of studies that tried to identify the basic elements which constitute visual design (e.g., point, line, and color). For example, Bauhaus masters such as Klee and Kandinsky provide a rich set of graphical abstraction and languages for visual expression from a

creator's perspective [Klee 1925][Kandinsky 1926]. On the other hand, some others influenced by theories of Gestalt psychology, such as Kepes and Dondis, carefully examine the nature of visual expression, which has helped designers to shape their expressions [Kepes 1961][Dondis 1986]. Such conceptual models and languages developed over the years have helped designers to explore their design solutions and have made visual communication rich and communicative.

There are two important perspectives their research: syntax and semantics. The research in syntax tries to identify essential constituents in visual communication and understand how they relate each other. On the other hand, research in semantics tries to understand communicative qualities of visual languages. Although their research provides a rich understanding in visual communication and helps designers formulate their design solutions, it is limited to the presentations of static media, such as paper. In other words, there is only a little known about the visual language expressed over time, and thus, it is of my interest in this investigation. However, my research is neither a study of syntax nor semantics of temporal visual language; rather, it tries to provide a theoretical framework by which the syntax and semantics of temporal visual language are discussed, and analyzed.

More recently, few researchers have also explored the temporal use of visual expression, particularly concerning television graphics and interactive presentation. Owen provides a matrix of visual techniques for three dimensional and dynamic diagrams [Sivasankaran 1991]. Bork provides a set of vocabulary for characterizing the temporal use of text on computer screens [Bork 1983]. Hiebert employs a systematic approach for the design of temporal presentation, and proposes a scoring method as a means of sketching out interaction of forms expressed over time [Hiebert 1992]. Hiebert also proposes a language to characterize visual presentation. These approaches are similar to that of this research, in that it proposes a general descriptive scheme for analyzing and constructing temporal presentation. Nonetheless, there are three unique points in this research that distinguish it from the previous work.

First, the abstraction of temporal forms proposed here emphasizes the development of an *analytical description*—structural and hierarchical—which enables the representation of complex temporal forms. It also encourages considering temporal form as a *conscious unit*—manipulable object—in which a designer can compose, or layout, in time and space. The abstraction is also purely structural, in that it does not intend to provide any principles or categorization scheme regarding particular forms and communication; instead, the proposed model is intended to afford such analytical discussions.

The second unique aspect of my research is to clearly distinguish *form* from *content*, and to consider the temporal form as a situated realization of the content expressed over time. For example, an email message can be expressed in typography or voice, depending on its immediate situation, depending on whether a reader is engaged in a visual task (e.g., drawing a diagram) or an auditory task (e.g., talking on a phone). In this example, the email message is said to be represented by an appropriate form based on the context in which it is situated.

I have just treated voice as if it is a temporal form without a prior explanation. In fact, in my work, voice or any other physical forms of presentation (e.g., lighting, infra-red etc.) are considered a form. This is the third point. The notion of form is no longer limited to a single physical realization. A temporal form can be composed of visual, auditory, and any other components—similar to the way a piece of text is considered as having various formal component in static design, such as color, typeface, size, and so forth.

This research is rooted in traditional graphic design and tries to appropriate it in the realm of computer-based communication by adding new dimensions to the field of study. It emphasizes the precise, structural understanding of a design solution and its associated decisions, so as to encourage reflective design thinking and to make responsive design solutions computable. The notion of form in traditional design has been naturally extended from visual form to rather general communicative expression.

#### 2.4. Summary

In this chapter, I have outlined my approach from three view points. First, the type of the proposed theory—prescriptive and generative—is identified within a larger context of design theories. Second, both conceptual and theoretical components of the Model of Dynamic Design are described. The model employs a decentralized model as a natural approach to support design thinking, influenced by cognitive models of designing, and suggests a multiplicity of knowledge and focus of attention involved in designing, borrows its conceptual framework from the improvisation, bases its theoretical and technical framework on the theories of reactive multiagent systems. Finally, I have also discussed my approach from the perspective of research and practice in traditional graphic design and argued that the field of design lacks models and languages that are able to address dynamic and responsive design solutions.

Having presented foundations of my approach, the next step is to present the core theoretical framework of the Model of Dynamic Design. Chapter 3 introduces a multiagent model of design which considers a design solution as an emergent entity generated by a collection of active design agents. Chapter 4 enriches the multiagent model by adding a precise descriptive structure for an agent's formal actions—or temporal form.

## 3. A Model of Dynamic Design

This Chapter presents a *Model of Dynamic Design* in which a design solution is considered an emergent behavior of a collection of active design agents. This chapter focuses on the multiagent aspect of the model. First, I begin my discussion with an overview of the model. Then, the characteristics of the multiagent model are presented in detail.

### 3.1. Overview

The motivation to adopt the multiagent model has been inspired by the findings in the cognitive models of designing which characterize a process of design as multiple ways of seeing. Recall that, in Section 2.2.1, I have postulated that a dynamic design solution can be described as centered around meaningful units of information, or design elements. Consequently, the proposed framework models a design solution as a system<sup>1</sup> consisting of a collection of smaller design systems, called design agents, each of which system corresponds to a segment of information. Specifically, the smaller system, or design agent, is responsible for presenting a particular segment of information. A design agent is a system that can modify its expressive behavior as the context changes and can cooperate with other design agents. In traditional graphic design, a design element is usually described by a set of fixed attributes, while in dynamic design, a design element is described by a set of *dynamic activities*.

1. The term "system" used in this chapter should not be confused with "computational system," software, or hardware. "System" here simply refers to a conceptual entity that is dynamic.

The term *context* here is used from the perspective of an agent, and it is comprised of information, the reader's intention, and the immediate presentation environment (e.g., surrounding visual elements). Any of these three constituents are potentially dynamic.

In a dynamic design solution, evaluation criteria described by a designer are considered constant unless the designer alters them. This assumption can be said to be *constancy of appreciation* [Schön 1983][Rowe 1987]. Even though the appreciation

scheme may vary from one context to another, the design heuristic is constant as a whole. This assumption is important particularly when the solution is represented in a computer system. It guarantees that a design solution maintains its value system unless a designer explicitly alters it.

In the Model of Dynamic Design, agents are generally assumed to be cooperative—they must try to collaborate as opposed to fight. While the game of karate is a highly improvisational activity, it is not a cooperative performance. In other words, the use of improvisational agents alone does not guarantee a collaborative design solution. It is a designer's task to describe the behaviors of agents in such a way that they can coordinate their behaviors to solve a design problem.

It has often been suggested that discrepancies among agents are useful in the design of distributed systems [Schön & Wiggins 1992][Papazian 1993][Bucciarelli & Schön 1992]. In particular, discrepancies are effective when a system involves multiple global modules that are able to negotiate to optimize a solution similar to a team of experts working together. However, I have decided to assume that agents do not have discrepancies for two reasons. First, the notion of negotiation can become an obstacle to thinking about continuous solutions. Second, since the proposed model views a design solution as a team of improvisational agents, it is more natural to consider collaborating agents.

The following sections more look closely at the characteristics of the design agent and how its behavior can be described.

### 3.2. Agent's ability

The behavior of the agent is viewed as its ability. If a design agent is capable of communicating a particular aspect of information at any given situation in such a manner that a designer desires, the agent's ability is said to satisfy the designer's intention.

I have described that an agent acts, or presents information, according to its given context. But, how do agents achieve that, or how does a designer describe their ability? The proposed multiagent model of design adopts reactive agents as constituents. As described in Section 2.2.2, design agents are like performers in an improvisational performance, in that they perform reactively using their skills, without deliberately planning their actions. Performers seem to simply act on immediate situations, but this does not imply their performance is not intelligent. Rather, deliberation is considered embedded in their reactive pattern, or skill, through previous training. Furthermore, an emergent expression created by a team of reactive performers is also a highly intelligent group-activity. Consequently, in this multiagent model, the ability of the design agent is described as a set of situation-action patterns.

Agent's ability is specified either for a particular agent or for a particular class of agent. When it is specified for a class of agent, agents that belong to the same class share their behavior. For example, there may be a set of design agents belonging to one class that are all responsible for presenting a headline in a dynamic electronic

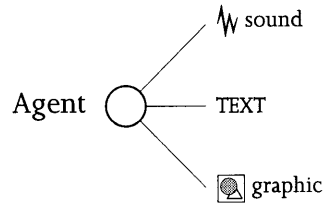
news system. In dynamic design, it is often the case that a designer focuses on a behavior for a class of design agents, rather than a particular design agent, since particular information is usually unknown at the time of designing. However, there is nothing to prevent designers from creating a single unique agent. A clock agent in Dynamic News System is an example of such a unique agent.

The ability of the agent is described by the following properties:

- Physical Realization
- State
- Sensor
- Action
- Strategy

### 3.2.1. Physical Realization

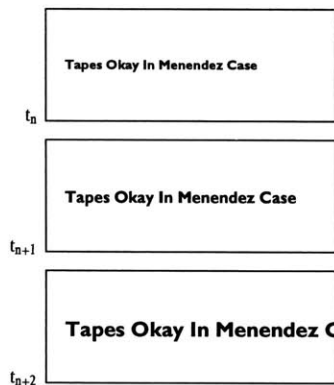
Physical realization is the agent's presentation perceivable to a reader. It is important to recognize that the agent is not bound to, but is independent of, a particular type of expression (**Figure 3.1**). For example, an agent which is responsible for presenting a headline in Dynamic News Display may express itself using text, or alternately, voice. Since the agent is an active entity, it chooses an appropriate expressive means according to a given context. It is assumed that appropriate physical realization for the agent is determined based on the available technology for a particular design problem.



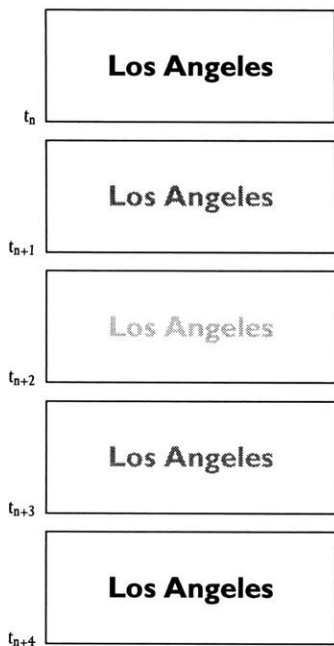
**Figure 3.1.** An agent can have multiple physical realizations.

### 3.2.2. State

State is a unit of what the agent “knows.” It includes the agent's current physical properties and the external information. Physical properties are information about the agent's physical realization. For example, physical properties of a text include color and typographic attributes. For agents with auditory representation, loudness and pitch can be their physical property. External information includes other related agents and current context. For example, a placename agent knows about headline agents that were issued there, so that it can inform associated headline agents when it is clicked. External information also includes contextual information about reader's intention and changes in information. In Dynamic News Display, external information is provided by the application program.



**Figure 3.2.** A headline agent's action that increases text size over a short period of time.



**Figure 3.3.** A placename's action that flashes a text.

### 3.2.3. Sensor

The design agent is assumed capable of sensing information from the external world, including information to be presented (e.g., a headline from a database), the reader's intention, and other agents activities. A set of sensors are defined for an agent by a designer. A content observed by a sensor is a special type of a state.

### 3.2.4. Action

Action is the abstraction for the agent's basic ability—just as body forms and phrases are the abstraction for describing performer's skill in dance. There are three basic categories of actions that the agents can perform: formal, communicative, and external. Formal action is an expressive action that influences the agent's form, such as typography and color. The representation of formal actions is described in depth in Chapter 4. For example, a gradual change of size as shown in **Figure 3.2** is a formal action. Blinking as shown in **Figure 3.3** that may be used to attract a reader's attention is also a formal action. Communicative action is the act of sending a message to other agents, which is used when coordination is necessary. An example of communicative action is a message sending action taken by a placename agent to inform its selection to associated headline agents. Finally, external action is a type of action used to influence outside of the multiagent system, such as an application program. For example, there can be a quit button agent which terminates the application program.

Each formal action—temporal forms which an agent can perform—must be carefully described by a designer for a particular design problem. An action can be performed instantaneously (e.g., change font), or it may take a certain duration (e.g., gradually increase size). This part of the model does not prescribe how precisely actions are described. However, an abstraction of temporal form introduced in the next chapter will provide a descriptive language which allows careful and analytical understanding of forms expressed over time.

Any meaningful formal expression can be considered an action. However, what can be considered meaningful action is not obvious. In general, a unit of agent's action is solely determined by how a designer views its role in communication. For example, consider the blinking action shown in **Figure 3.3**. A designer may consider it as a meaningful action to attract a reader's attention. Alternatively, the first half of the same action, which changes the color of text from dark to bright, can be considered an action in some other case. The blinking action can be considered a composite action, or a single action, depending on a designer's view. The notion of action simply provides an abstraction to structure temporal design as a collection



of meaningful temporal forms; rather than a collection or interpolation of static states.

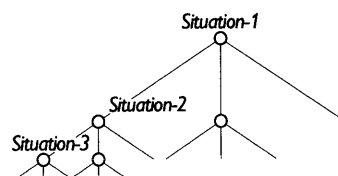
Action can either be persistent or flexible. The persistent action is a type of action that can not be terminated during the middle of its performance. For example, suppose the action shown in **Figure 3.3** is a persistent action and currently repeatedly performed by a placename because it is important. Even if the agent becomes no longer important at  $t_{i+2}$  (the time local to the action), the agent does not terminate this action and selects an action after completing this action. The flexible action is a type of action which can be terminated at any time during its performance.

### 3.2.5. Strategy

Strategy is the layer of abstraction over actions, which describes the agent's ability to achieve a particular goal. Strategy is like performer's knowledge to express certain themes in a musical or dance improvisation; that is, the theme is achieved by selecting an appropriate action at an appropriate time. A set of strategies determines the ability of a design agent.

In order to select an appropriate action as a response to its immediate situation, the agent must be able to recognize various situations by which it can be surrounded. In other words, the designer of a dynamic design solution must identify possible situations for individual agents. A situation can be an agent's own state, another agent's state, information it's presenting, a reader's intention, or a combination of these. For example, situations that the placename agent in the news display scenario can recognize include when a cursor is above its text, the text is clicked, there are headline agents associated with its place, and there is any news story being presented.

Situations are often structured in a hierarchical manner. For the headline agent in the news display, suppose there is the following situation: a reader is interested in reading news articles based on locations (*Situation-1*). Then if a reader selects a placename with which the headline agent is associated, it generates a sub-strategy for the headline: a reader is potentially interested in reading its story (*Situation-2*). Notice that, *Situation-2* is situated within a larger situation *Situation-1*. You can also find another substitution within *Situation-2* when a reader selects the agent: a reader is interested in reading its story (*Situation-3*). In other words, when the agent is situated in one situation, it does not have to watch out for all the situations it can recognize; rather, it only has to consider sub-situations which potentially happen within the immediate high level situation. **Figure 3.4** presents a schematic diagram of situations that are hierarchically defined.



**Figure 3.4.** A schematic diagram showing a hierarchy of situations.

Given the agent's abilities to recognize necessary situations, a strategy is specified as a simple procedure and is composed of a set of actions and a set of reactive rules, based on the reactive agent model proposed by Singh. The simplest strategy consists of a sequence of one or more design actions. For example, a headline's strategy to attract a viewer's attention may consist of the actions of becoming red and increasing its size. From a perspective of a headline agent, this strategy can be written as:

S1 **attract-viewer's-attention-strategy:**  
*perform flash-action*

More complex strategies involve some decision making to determine what action to take according to the immediate situation. An example of this type of complex strategy, for the headline agent, is to attract a viewer when the news item is important. This strategy may be achieved by adding a conditional statement to the above example:

S2 **attract-viewer's-attention-strategy:**  
*If my news article is important for a reader*  
*perform **flash-action***  
*otherwise*  
***do-nothing***

A strategy can also consist of a set of other strategies. For instance, a simple strategy for a headline agent to use **attract-viewer's-attention-strategy** until it is deleted by a reader can be defined as follows:

S3 **basic-strategy:**  
*while I am not deleted,*  
*use **attract-viewer's-attention-strategy***

Notice that these two strategies, **basic-strategy** and **attract-viewer's-attention-strategy**, involve three situations. First, a higher level situation is when the headline agent is not deleted. Then, there are two sub-situations: one when the agent's news article is important, another when it is not.

For example, if a headline agent is not deleted, and its news story is important for a reader, the agent selects **flash-action** to perform. It usually takes a certain duration to perform an action. While **flash-action** is being performed by the agent, it also checks whether or not the situation holds. If the situation changes, e.g., the agent's news story becomes no longer important, the agent stops performing **flash-action**, and starts doing nothing according to **attract-viewer's-attention-strategy** (S2). Figure 3.5 presents a schematic diagram of situation changes.



Figure 3.5. A schematic diagram that shows how a headline agent selects an action at two different situations at  $t_i$  and  $t_j$ .

Note that the strategy *s4* defined below is equivalent to *s3*. *s4* simply integrates **attract-viewer's-attention-strategy** (*s2*) into *s3*.

*s4* **basic-strategy**  
 while I am not deleted,  
   If my news article is important for a reader  
     perform **flash-action**  
 otherwise  
   **do-nothing**

An ability of an agent expressed by *s4* is also equivalent to that of *s3*, along with **attract-viewer's-attention-strategy** (*s2*). The only difference it makes is that if we did not define **attract-viewer's-attention-strategy** (*s2*), *s2* is no longer accessible as a strategy from other strategies. In other words, *s4* is treated as a meaningful unit of knowledge. The same strategy can be defined in different ways; however, it must be designed to represent a meaningful unit of ability that an agent can use to achieve its communication goals. This decision on the structure of strategies is an important part of designing.

I have shown simple examples of strategies to illustrate the fundamental concept. Strategy is a useful abstraction which a designer can use in the course of exploring design solutions. It provides designers with a framework for identifying changes in context and determining appropriate dynamic design solutions that respond to them. To describe a rather realistic design solution, more situations and strategies need to be described. Examples of more complex design solutions are presented in Part 2.

### 3.2.6. External view of Agent's Ability

Figure 3.6 summarizes the agent's ability. The design agent acts on the immediate situation using a strategy which is designed to achieve its current goal. The information observed from the external world and messages sent from other agents are used by the strategy to determine what action to take.

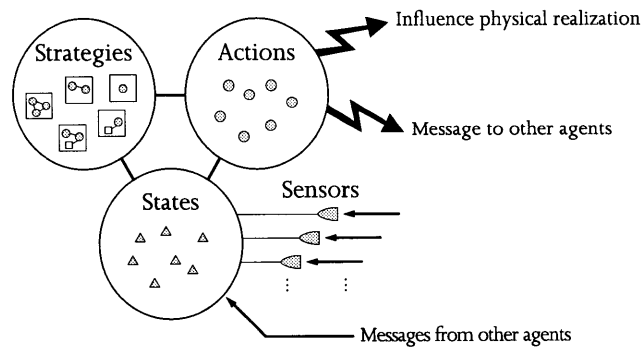


Figure 3.6. Schematic diagram of the ability of the agent.

It may seem complex to understand how an agent behaves given a set of strategies and actions. One method which helps one understand an agent's behavior is to consider its history. Figure 3.7 shows a history of a headline agent. This diagram shows that the agent repeated performing **flash-action** through  $t_i$ , since its news story

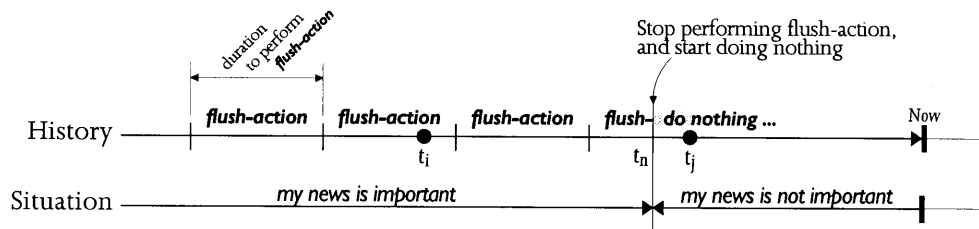


Figure 3.7. A historical view of a headline agent.

is important. Then, its situation changed at  $t_n$ , where the news story became no longer important (a new situation). At  $t_n$  the agent stopped performing **flush-action** and started doing nothing and kept doing nothing from  $t_n$  until now.

Furthermore, an agent's future can be understood in terms of history. Given a particular point in time and an agent, there are many possible histories. As time progresses, an agent can be said to select one (and only one) history. Figure 3.8 is a schematic diagram showing the concept of possible histories found in the future of an agent. The notion of history is an external description of an agent's behavior, and it is not remembered by the agent. History is an analytical device that can be used to understand the agent's behaviors.

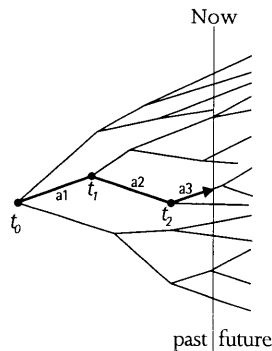
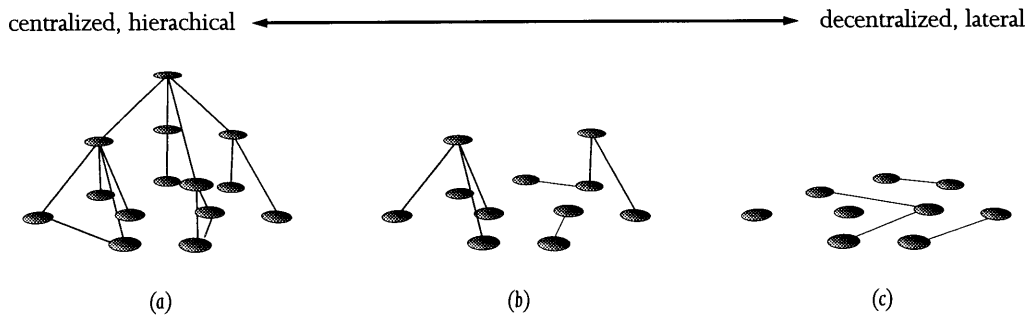


Figure 3.8. A history of an agent is a result of selecting appropriate actions over time.

### 3.3. Organization of Agents

The model emphasizes a decentralized and lateral interaction among design agents, as opposed to a purely centralized and hierarchical one (Figure 3.9). Like dancers or players on a football team, who do not have strict hierarchical control, the design agents collaboratively act in order to achieve a global goal. However, an agent can become a local leader with some authority over other agents, like the lead dancer or the quarterback. For example, a headline agent in the dynamic news design can be a leader which oversees a story agent and a photograph agent. Leaders can also be chosen dynamically. Between the extremes of lateral and hierarchical organization, there are continuous levels of different types of organization. Thus, the model intentionally does not define a specific and fixed type of organization. Decisions about the dynamic changes in agents' organizational structure (e.g., creating temporal leaders and their roles) are left to designers.



**Figure 3.9.** Simplified diagrams of organization structures: (a) centralized and hierarchical, (b) mostly decentralized, but with partial leaders, and (c) decentralized and completely lateral.

As a consequence of using a decentralized model, the Model of Dynamic Design potentially inherits some of the disadvantages of decentralized systems. One well-known disadvantage of decentralized systems is a lack of global control, which has a potential to create an oscillating situation. However, there are several reasons why I decided not to use meta-level agents to oversee design agents. First, since the individual activities of agents do not need to be explicitly collected, oscillating activities do not necessarily become a bottleneck in terms of a design solution. Also, changes in context can simply release an oscillating activity. Furthermore, it is conceivable that a designer may intentionally utilize oscillating activities as a part of a design solution. Second, the behaviors of meta-agents are difficult to articulate. For example, finding a method to discover undesirable oscillating situations is not a trivial task, particularly when the chain of effects is long. Furthermore, such description is likely to be remote from the design solution itself, hence making it difficult for designers to deal with.

One might also argue that it is difficult to have exact control over an overall design solution (e.g., color harmony and visual impression) using a decentralized model without global management. As postulated in Section 2.2, global aspects of design solutions are difficult to articulate. Since it is dangerous to use a descriptive method that a designer can not be articulate, rather concrete and situated descriptions are preferred. I assert that it is a designer's responsibility to describe the behaviors of individual design agents in such a way that the solution is coherently coordinated. Similar to the role of a director in improvisational performance, the designer must carefully instruct and rehearse design agents in such a way that they generate a quality performance.

I have described that a design solution as a whole is an emergent behavior of a team of design agents. This emergent design solution is said to be generated by agents' **group strategies**. Or, a partial set of design agents may use a group strategy to collaboratively present information. However, there is no explicit description about group strategies. Group strategy is an abstract concept which consists of a collection of strategies used by a set of agents. Since each agent is capable of identifying its immediate situation, including what others are doing, each one participates in solving a larger problem by taking its own role. Also, a collection of situations for a group, or entire set, of agents can be called a **composite situation**. Group strategy and composite

situation are useful concepts when considering a design solution as a whole. In addition, it is possible to use the notion of history to understand the entire design solution, or a group of design agents. History is also useful for analyzing the design solution a whole.

#### **3.4. Life Span of an Agent**

When considering a design solution, a designer must consider the birth and death of an agent. An agent can be created when information is generated. For example, in the on-line news scenario, a headline agent can be created when a news article is issued. An agent can also be created by another existing agent. The decision on the agent's termination is less obvious than its creation. An agent may terminate itself when the information it is representing is no longer accessible, or it can wait until the information becomes available again. Alternatively, an agent may terminate itself when its associated information is no longer important for a recipient. It is also possible to consider that the agent never terminates, or it is terminated by some other agent. It is a designer's role to determine how agents are born and when and how they are terminated. The decisions on the life span of the agents depend on the nature of the design problem and how the designer conceives a solution.

#### **3.5. Other Distributed Models of Design**

Although there has been much research in the past that implies a parallel, or decentralized, model of designing, there have only been a few researchers who proposed such a theory. Papazian proposes an approach to the development of design systems based on the multiplicity of semantics and the dynamic shift of focus of attention in the design process [Papazian 1983]. He has developed a model of design generation based on an arbitrary number of metaphor modules in which each module provides a particular way of seeing. His experimental computer system can produce an opportunistic, robust, and complex design by providing multiple interpretation mechanisms that are allowed to conflict. However, his model does not provide a strong mechanism for coordinating these multiple modules.

The blackboard model of design can also be seen as a distributed model of designing [Coyne et al. 1989][Whitefield & Warren 1989]. The blackboard model involves many small, distinct, intelligent processes called knowledge sources (KS's) that indirectly communicate information with other KS's through a common space called a blackboard. Although the blackboard model is essentially a decentralized model of intelligence, the granularity of a unit process is significantly smaller than that of the multiple ways of seeing discussed above.

Xiang postulates a decentralized model of decision making in the context of environmental planning [Xiang 1993]. He provides a computational model of coordinating a network of self-interested (non-hierarchical) problem solvers for the development of organizational decision support systems. Although his model is intended to involve human decision makers as part of a system, it shares the basic approach with the model proposed here.

Although these models share a common ground with the model postulated in this proposal, they are different in three ways. First, the existing distributed models of design are concerned with creating fixed solutions, whereas this research focuses on creating active solutions that continuously adapt to the changes in their immediate context. Second, most of them use various global experts domain as their component modules, while the Model of Dynamic Design uses local and concrete design elements. Finally, all of other distributed models are not in the domain of visual design. At the present time, the Model of Dynamic Design is the only work in the domain of graphic design to use distributed model to represent design solutions.

### 3.6. Summary

This chapter has presented the core framework of the Model of Dynamic Design—the *multiagent model of design*. This multiagent model provides a view that considers a design problem a continuous stream, and describes design solutions as a collection of active design agents. This model enables a fundamentally different approach towards solving design problems in dynamic and continuous communication environments, compared to a traditional approach.

In this chapter, the design agent's formal action is simply described as a meaningful unit of temporal change in agent's form. The next chapter further develops the Model of Dynamic Design by adding a descriptive language for constructing and analyzing agent's formal actions that are expressed over time.





## 4. An Abstraction of Temporal Forms

This chapter further develops the theoretical framework of dynamic design by introducing **an abstraction of temporal forms**. This abstraction scheme is intended to provide a means to describe precise formal behaviors of the design agent.

### 4.1. Formal Dimension

Formal dimension is roughly equivalent to the formal attributes of the design element in static design, such as typeface and color. Formal dimension is different from static attribute, only in that its value is assumed to be dynamic. The agent's formal state, described in Section 2.4.1. is a particular value assigned to a formal dimension at a particular time. For example, hue, brightness, typographic weight, and position are the formal dimensions; and red, dark, bold, or ( $x=10.0$ ,  $y=10.0$ ), are the states of these formal dimensions respectively.

In the process of traditional static design, i.e., page layout, once a value for a particular attribute is decided for a design element, the value is considered *fixed*, without involving a notion of time. Even when an attribute value is altered for some reason, the concept of time does not exist in the description of forms. In the proposed model, each formal dimension of a design element must be considered a fluid entity. For example, a text element can be described as *being* 24 point, or *flashing* between black and white every two seconds.

As a consequence, it is feasible to consider a design element which is continuously changing without any breaks. However, since design solutions usually demand meaningful and intentional communication, we can break an agent's expression in a set of meaningful units. This meaningful form was described as **Action** in Chapter 2. This research proposes an abstraction in which the action is described by **phrases** and **temporal forms** (or simply **forms**). Phrase and form are the terms that have been adopted from the abstraction of dance performance [Blom & Chaplin 1988] and used as the abstraction for describing the agent's formal activities.

#### 4.2. Phrase

Phrase is the primitive unit of the agent's formal action and is described by the following properties:

- A method
- A set of formal dimensions it changes
- A set of formal dimensions it uses
- A set of external information it uses
- Duration

Method is a function of time that is used to realize the temporal changes of one or more formal dimensions. A method may or may not use particular values of formal dimensions and external information, in order to change its formal dimensions. Duration describes how long it takes a phrase to occur.

Figure 4.1. shows a simple example of a phrase. Suppose we have a headline agent,  $H_1$ , which is responsible for presenting text information: "Text."  $H_1$  belongs to the class, headline.  $phr_1$  is a phrase which is shared by design agents that are instances of the headline class and changes the brightness attribute,  $b$ , of an agent according to a method shown in the graph.

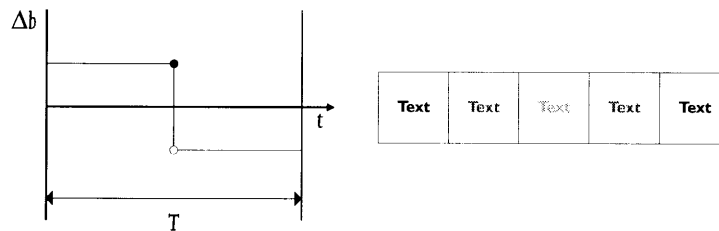


Figure 4.1. An example of a phrase which changes brightness.

Figure 4.2. shows a slightly more complex example of a phrase. This phrase changes the position of a text from its current place ( $t_n$ ) to a given destination ( $t_{n+4}$ ). For example, in Dynamic News Display, a similar phrase is used by headline agents when they are born. A headline moves from an initial position, which is very close to a view point in  $z$  dimension, to the place its news was issued, which is farther in  $z$ .

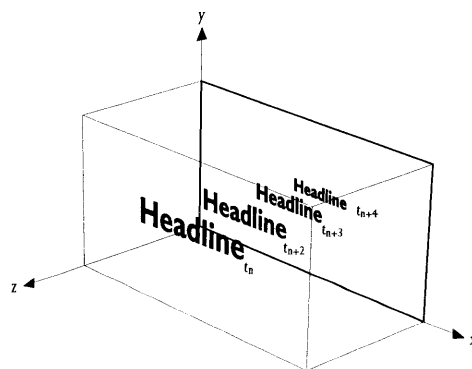


Figure 4.2. An example of a phrase which gradually moves a text from a current place to a destination.

Note that even though I visually demonstrate a phrase in **Figure 4.1** and **4.2** in order to illustrate the concept, a phrase itself is not considered an action. It is a primitive construct which comprises a **temporal form** described in the next section, with other phrases.

In creating concrete design solutions, three basic methods are possible for generating phrases, which are adopted from Rowe's musical composition methods [Rowe 1993]:

- **Sequence**
- **Algorithmic generation**
- **Transformation of above**

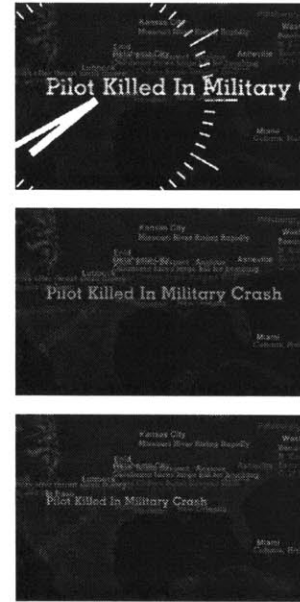
Sequence is a fixed series of changes that can be recorded. A phrase generated as a sequence always produces the same result. A phrase can also be generated by an algorithm. Also, a phrase created by sequence or algorithm can be a transformation to another phrase. Phrases are units of agents' behaviors and they must be carefully crafted in order to achieve goals of a particular design problem.

### 4.3. Temporal Form

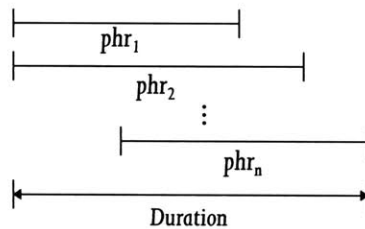
Temporal form, or simply form, is an abstract concept that is used to describe individual actions of the design agent. One or more phrases constitute a form, or an action—i.e., a unit of the formal behavior of the agent. In general, the design agent's expressive behavior, which is determined by its strategies, is realized as a series of temporal forms performed over time. A form is described by the following properties:

- A set of phrases
- A set of temporal relationships over its phrases
- Duration

**Figure 4.4** shows a schematic diagram of a temporal form. A form can consist of any number of temporally overlapping phrases; thus, the resulting expression can be fairly complex. A phrase starts and ends at anytime within a duration of form. In other words, the boundary and duration of a form is determined by its first and last



**Figure 4.3.** A phrase shown in **Figure 4.2** is used in **Dynamic News Display**, when a new headline is introduced.



**Figure 4.4.** Schematic diagram of a form.

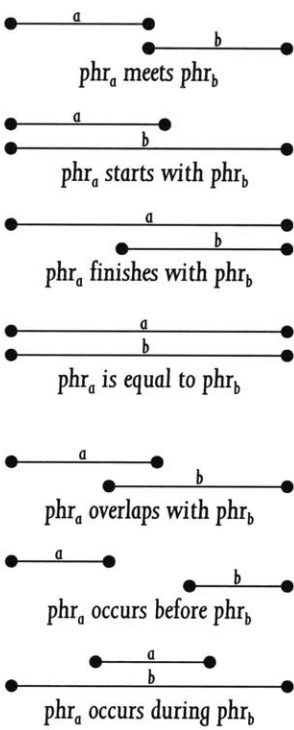


Figure 4-5. Schematic diagrams of temporal relationships between two phrases [Allen 1983].

phrases. A set of temporal relationships among phrases specifies when each component phrase should be performed. Figure 4-5 present the basic set of temporal relations between two phrases [Allen 1983].

Figure 4-6 presents a simple example of a form for a headline agent, which includes the phrase example,  $phr_1$ , shown in Figure 4.3. Suppose headline, a class of agent discussed earlier, comprises two other phrases:  $phr_2$ , which influences an agent’s font size attributes, and  $phr_3$ , which rotates a graphical object. We can now compose a form,  $f_1$ , which is a complex action consisting of three simple phrases.

As shown in Figure 4.6, phrases within a form often temporally overlap. A problem may occur when two or more phrases in a form simultaneously affect the same formal dimension. Temporally overlapping phrases which can alternate values of the same formal dimension may create unpredictable results. One solution to this problem is to impose a restriction rule: if there are two or more phrases that affect the same attribute in a single form, they must not overlap. However, the proposed model does not impose any of these rules for the purpose of generality.

In this document, temporal forms and phrases are represented by simple illustrations, similar to the ones used in Figure 4-1 and 4-2. Since there are many different ways in which a phrase can be created, this research does not provide a particular visualization method (e.g., scoring). Instead, a representation will be appropriated based on the nature of a phrase. For relatively complex composite forms, a schematic diagram like Figure 4.4 or 4-6 will be used to represent layers of phrases.

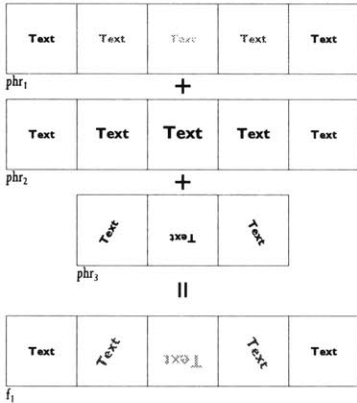


Figure 4.6. Example of a form that changes brightness, font size, and rotation.

#### 4.4. Expression

While phrase and form are the structural entities that describe the agent's action, expression is a general term used to describe a meaningful set of temporal forms, or actions, which represent a message in a particular fashion over time. For example, when a headline agent introduces its text, it first places text at an initial position and flashes white. Then the agent gradually moves the text so that it aligns to its associated placename. This introductory expression can be seen as a sequence of two actions: to appear with a flash, and to move.

In dynamic design, an expression is generated as a result of performing strategies. On one hand, if we look at a history of actions that a particular design agent performed, it is simply a consecutive list of actions. On the other, if we look at a particular agent's future, there are many sets of possible histories.

#### 4.5. Summary

I have presented an abstraction of temporal form, which provides a precise language for the description of the design agent's basic actions. In order to illustrate the idea of phrase, temporal form and expression, I have used examples in which phrases and forms are explicitly defined (*i.e.*, algorithmic). And it is often necessary to provide precise descriptions when they are implemented as a part of computer program. However, in the process of designing, phrases and forms can be implicitly identified, and whether or not a designer uses these specific terms—phrase, form, and expression—is irrelevant to the role of the abstraction. The essence of the abstraction is to recognize a temporal change in form as an *object* that can be manipulated and used purposefully. Also, it is to understand how temporal forms can be further analytically observed and described as a composition of phrases.



## 5. Designing Dynamic Design

**T**he Model of Dynamic Design—a multiagent model along with an abstraction for temporal forms—is intended to help designers analyze, describe and create dynamic design solutions. In this section, I outline how I envision the proposed model to be used in actual problem solving situations. The theoretical framework described in this dissertation does not impose a specific way in which the model is used. However, based on the framework, I suggest three types of situations in which the proposed model can be used.

### 5.1. Conceptual Framework

Although the model provides precise structural descriptions for representing dynamic design solutions, precise descriptions are not always necessary. A designer can use fundamental concepts of dynamic design, with an analogy of improvisation, as a framework for solving a design problem. The Model of Dynamic Design provides a perspective from which one perceives and understands design problems and solutions.

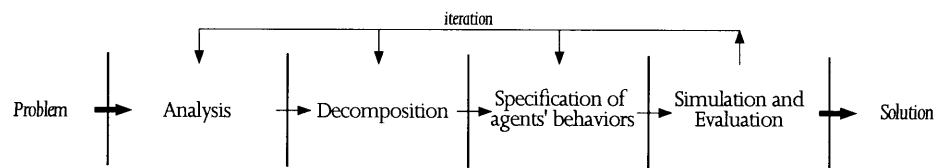
At this level, it is important to recognize that a design problem is not a collection of small discrete problems addressed over time; rather, it must be seen as a stream of dynamic problems. For example, consider the problem of visualizing news display. It should not be perceived as solving a new design problem as a new news story arrives at a workstation. Instead, the model suggests viewing the problem as one continuous flow, and describing that design solution as a dynamic entity—or process—that responds to the continuously changing context. It is also important to understand that this dynamic design solution is composed of a collection of collaborative design agents, each of which performs various actions according to its immediate context. Here, an action is understood as a meaningful communicative form expressed over time.

This conceptual level of dynamic design provides designers with critical languages when designing computer-based communication media. Analysis of the agent's role will help designers understand the nature of a design problem. Identification of the

agent's formal actions will provide a constructive discussion about communication. Identification of agent's organization can provide deeper understanding of the dynamic design problems.

## 5.2. Methodical Process

In addition to the conceptual model of Dynamic Design, this research proposes a process oriented method which a designer can follow to solve a dynamic design problem. **Figure 5.1** shows a general flow of a design method which uses the Model of Dynamic Design described in the preceding chapters. This method fundamentally resembles a general prescriptive design process paradigm—analysis, synthesis, and evaluation, proposed in other domains of design [Cross 1984].



**Figure 5.1.** A schematic diagram of the design process using the proposed theory.

The first phase is the analysis of the design problem. The designer must understand the nature of information, the goal of communication, and types of intended readers. The second phase is the decomposition of the problem. A design problem is decomposed and described as a set of multiple design agents. In this phase, a style of organization, types of design agents, and their roles and abilities are determined. The third phase is the specification of the agents' behaviors. Here, a designer designs—or describes—the behaviors of an individual class of design agents in the form of actions and strategies. Next, the solution, or partial solution, created in the third phase as a collection of design agents is evaluated by simulating their behavior in context.

Although this design method is presented in a linear fashion, the process of designing a dynamic solution is not a simple four-stage process; rather it is an iterative process of exploring and examining each agent's behavior and various relations among them. Regardless of the phases, the role of a designer is like that of a director in the performing arts, or a coach in a football team, in that the design process consists of a course of “dialogue” between a designer and the design agents. A designer must carefully determine the behaviors of agents in such a way that each agent can act its role according to its immediate context and can contribute to a design solution as an emergent whole.

## 5.3. A Computational Description

At the present time, in order to realize a dynamic design solution, it is inevitable to rely on computational support. The Model of Dynamic Design provides a basis for developing a **computer language** which designers can use in order to describe precise behaviors of design agents. A computer language, which I refer to as **agent description**



*language* in this dissertation, can be implemented in various ways, depending on hardware and software environments and characteristics of design problems. Nonetheless, a language must provide designers with a means of specifying different types of agents and their ability described in Chapter 2 and 3.

In general, an agent description language is developed along with a computational engine which interprets the language and generates dynamic design solution. On one hand, one may develop a general language which can be used in multiple design problems. On the other hand, a specific *agent description language* can be built for a particular application software.

#### **5.4. Summary**

I have presented three types of situations which the Model of Dynamic Design presented in Chapter 3 and 4 can be used in solving design problems. The fundamental conceptual framework of dynamic design provides designers with a critical language to create and analyze dynamic design solutions. A process oriented method guides designers through the course of developing and refining a design solution. A computer language provides designers with a means to realize dynamic design solutions.

This chapter concludes the description of the theoretical framework of this research. In Part 2, the Model of Dynamic Design is illustrated with case studies using concrete design problems.



## Part 2

The Model of Dynamic Design presented in Part 1 is applied in solving five different domains of dynamic design problems: dynamic news display, electronic mail reader, interactive poetry, geographic information display, and expressive messages. Issues raised through the creation of experimental design solutions are discussed.



## 6. Case Studies

Five experimental design solutions developed with the Model of Dynamic Design are presented. These design examples illustrate the use of the model in the description of dynamic design solutions, as well as in the process of generating these solutions.

The first three experimental design solutions were implemented using a system called *perForm*, software specially designed for this project based on the proposed model. I begin this chapter with an introduction to *perForm* in order to present how the following case studies are realized in a computational environment. Then, three case studies, Dynamic Display of News, Electronic Mail Display, and Interactive Poetry, will be presented. Thereafter, two examples in the domains of geographic information display and expressive typography, which partially use the Model of Dynamic Design will be presented.

This chapter focuses on illustrating how the Model of Dynamic Design can be used in solving concrete design problems. Reflective discussions on issues raised through the course of developing these solutions are presented in Chapter 7.

### 6.1. *perForm*: A Multiagent Design System

*perForm* is intended as an experimental apparatus which would provide a means to specify design agents' behavior based on the Model of Dynamic Design and to visually simulate their dynamic interaction over time.

*perForm* provides an agent description language called *persona*, along with a multiagent design simulation engine. This engine simulates parallel activities of design agents, whose dynamic behaviors are specified by *persona*. *persona* is implemented in LISP and provides a set of macros that are used to define the agents' ability, which includes actions, strategies, sensors, and messages. The syntax of the language is described in Appendix. However, in this chapter, descriptions of agents' behaviors will be presented in English to avoid introducing the details of *persona*'s syntax.

*perForm* uses a special three dimensional graphics software library for the visual realization of the design agent (Figure 6.1). This library enables the use of high quality typography, along with images and other graphical objects, in order to examine the proposed model with design solutions that are not too simplistic.

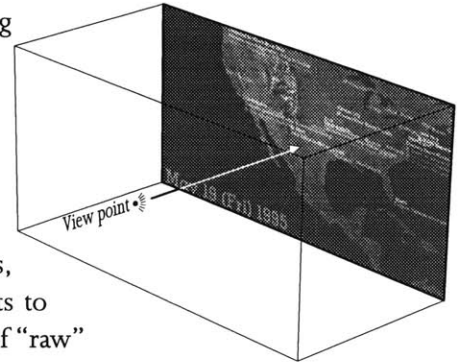


Figure 6.1. A diagrammatic view of three dimensional space used in the design of Dynamic News Display.

*perForm* is a software module which generates dynamic solutions, and it is intended to be situated in application software, such as a news display system. In other words, *perForm* is not intended to stand by itself. *perForm* expects to receive a semantically structured data set as input, instead of “raw” data. That is, it assumes that semantic structures are provided by an application program. It also expects that the reader’s intention is provided by application software. For example, in the news display scenario, data elements, such as headlines and news stories, as they come into *perForm*, must be tagged and their relationship must be explicitly represented.<sup>1</sup>

1. I do not imply that a dynamic design solution only operates on forms. In fact, there must be many semantic interpretations of information and reader interaction are made in terms of agents’ response to their situations. However, when it involves knowledge in other domains, such as language analysis and reader learning, interpretation of information is often performed by other parts of the system

In the following case studies, such application programs are partially implemented in order to provide each design solution with structured information and information about reader’s interaction over time. Figure 6.2 shows a schematic flow of information in a News Display System.

In addition, recall that behaviors of the design agent can be specified for a class of information, or a particular piece of information (described in Section 2.2). *persona* only allows a description for the former, in that agents’ abilities are described for a class of design agent. Thus, a single unique agent must be created as a class with a single instance.

In the following sections, each case study is generally presented in the following format. First, a basic design problem is introduced, followed by my design intention. Next, the decomposition of a design problem, or how information is represented as a collection of design agents, is presented. Then, dynamic behavior, or *ability*, of each agent and the design solution as a whole are presented. This last part starts with an overview of the agents’ roles and behaviors. Then, a description of the

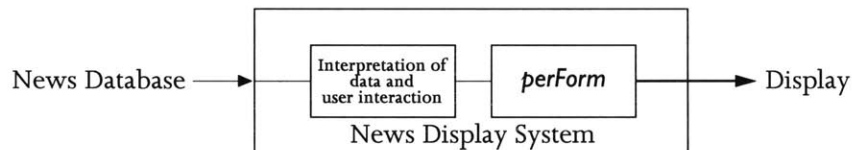


Figure 6.2. A schematic diagram of Dynamic News Display. *perForm* receives a structured and interpreted set of data and uses interpreted information about a reader in order to produce a dynamic design solution.

agents' strategies are presented in depth. As described in Chapter 5, the design processes took place in an iterative manner, although design solutions are presented in a linear fashion.

## 6.2. Dynamic News Display

### 6.2.1. Design Problem

The problem is to design a visual interface for a computer system that provides users with access to an on-line news database where news articles are produced at any time. Assume that this system, Dynamic News Display, collects news articles whenever a new article in a specified category is entered into the database. This system should be situated at an office or a home, where a user can occasionally browse through the overview of what is happening, and if there is any article that attracts the user, he or she can read the article in detail. This system should also be usable for reading news stories that are issued within an particular period of time. For example, a reader may use the system to quickly review news stories of that day at night.

### 6.2.2. Intention

I intended to create a visual interface that can always provide users with an overview of entire news articles issued within a certain time period (e.g., the past 12 hours). I also wanted to provide users with multiple levels of reading between individual news stories to the entire set of news articles. In addition, I decided to provide news articles based on the city from where they are issued. I also decided that its overall look should be relatively conservative and functional. I intended to provide a visual impression of the world as an active and busy place where various things are happening simultaneously.

### 6.2.3. Decomposition

The above database information is decomposed into the following design agents (Figure 6.3): Headline Agent, Story Agent, Map Agent, Placename agent, Clock agent, and Date agent.

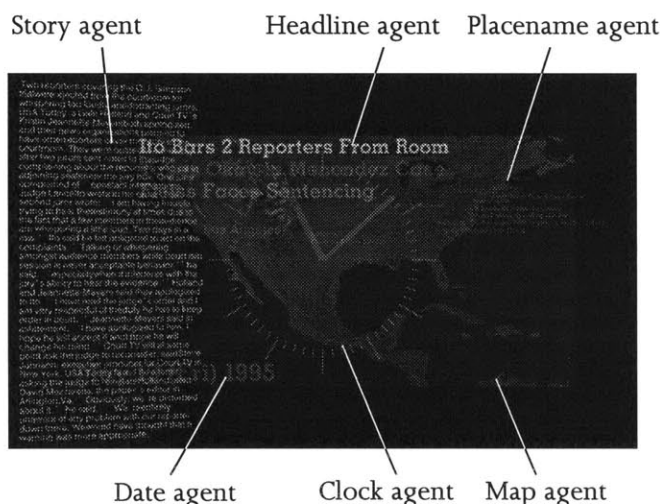


Figure 6.3. Realizations of six design agents are indicated on a screenshot of the News Display.

Notice that a news article is decomposed into a headline agent and a story agent. The role of the Headline Agent is to quickly inform the user about the content, as well as to present the age of its associated news article. The role of the Story Agent is simply to provide its message content. The reason to decompose a news article into two different design agents (headline and story) is to create a visually less overwhelming interface, as well as to provide multiple readings of the news database. It is also conceivable to create a class of design agent which is responsible for presenting the entire news article without decomposing it into two agents. .

Behaviors of the other agents are determined straightforwardly. The role of the Map Agent is to provide a visual background where other design agents are situated based on their spatial associations. The role of the Placename Agent is obviously a label to a city. In addition, it functions as an interface element in which a user can point and click. The Placename Agent 'knows' about headline agents that are currently representing news articles issued there. The roles of the Clock Agent and date agent are to provide time and date respectively.

#### 6.2.4. Agents' behaviors

The following paragraphs present scenes from the Dynamic News Display. This particular example used a set of news articles categorized as top U.S. news from a Clarinet database, issued on May 19, 1995.

Figure 6.4 shows the display just before 9 o'clock in the morning. There are three headlines at Los Angeles, one at Kansas City, and one at Washington D.C. These placename agents responsible for presenting Los Angeles, Kansas City, and Washington D.C. have recognized that there are news stories associated with them and changed their color from light gray to bright orange. Also, notice that three headlines at Los Angeles have larger type size than the others. This scene is taken right after a reader

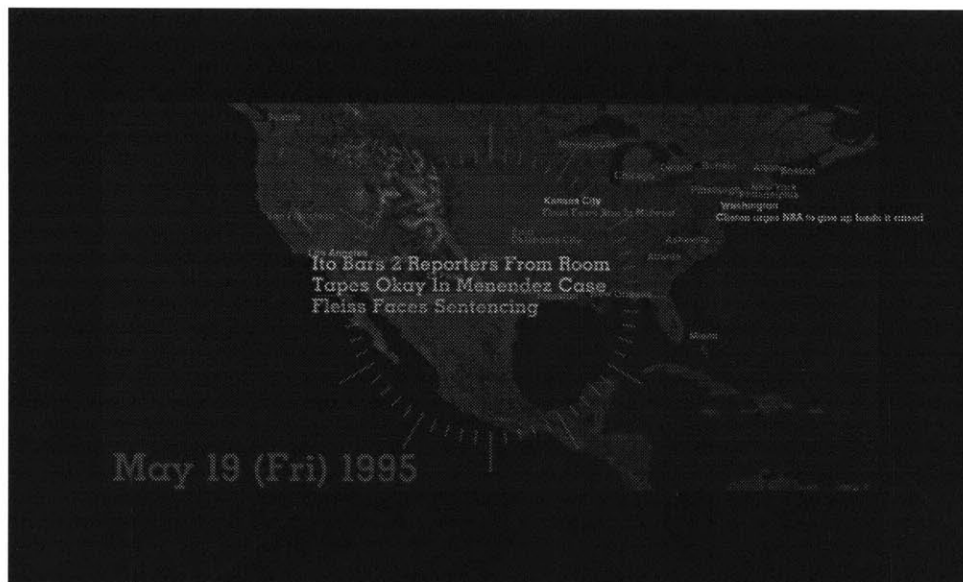
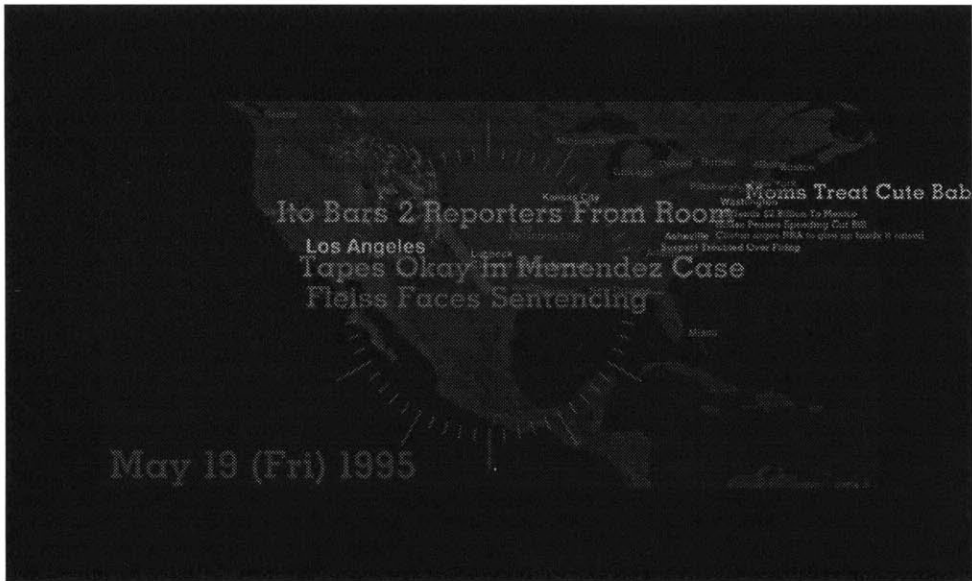
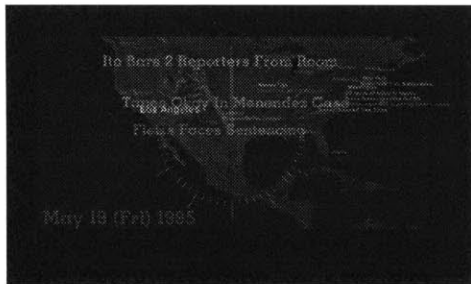


Figure 6.4. At 8:54 in the morning of May, 19, 1995, there are news stories at Los Angeles, Kansas City, and Washington D.C. Three headline agents at Los Angeles has grown size of their realization (text) in reaction to a reader's placing a cursor above their placename.

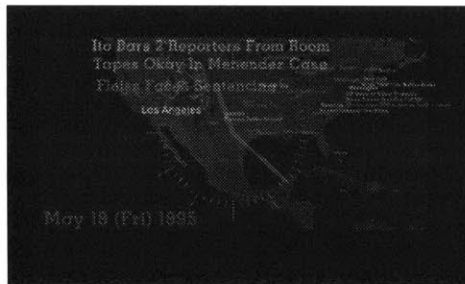




(a)



(b)



(c)

**Figure 6.5.** A sequence of scenes after Los Angeles is selected by a reader's clicking. Headline texts presented by headline agents gradually reach their destination (upper left), where their role becomes to wait for a reader's selection.

moved a cursor over to the text “Los Angeles.” First, the Los Angeles agent notices that a cursor is on top of it—a new situation, and then it inform headlines associated with it that it is being focused on. Having been informed by the placename, those headline agents now “know” that their placename is being focused on. According to my instructions for this situation, these headline agents gradually increased their type size. They then keep that size while the cursor is on the associated placename, in order to maintain the readability of their text. Each headline agent left-aligns its text to the headline that is issued next to it with a certain leading. The most recent headline aligns its text to its placename. In general, the Headline Agent tries to maintain this alignment whether or not its associated placename is focused, using the same strategy. The Headline Agent also looks at the age of its news article and changes the translucency of its text proportional to its age. For example, the headline presented at Kansas City shows that its story is older than other news stories.

**Figure 6.5** presents a sequence of three scenes after Los Angeles is selected (by clicking its placename text). Three headline agents gradually move their realization (text) to the upper left part of the display. In this design, selecting a placename means further examining the articles at that location (i.e., to read the news stories).

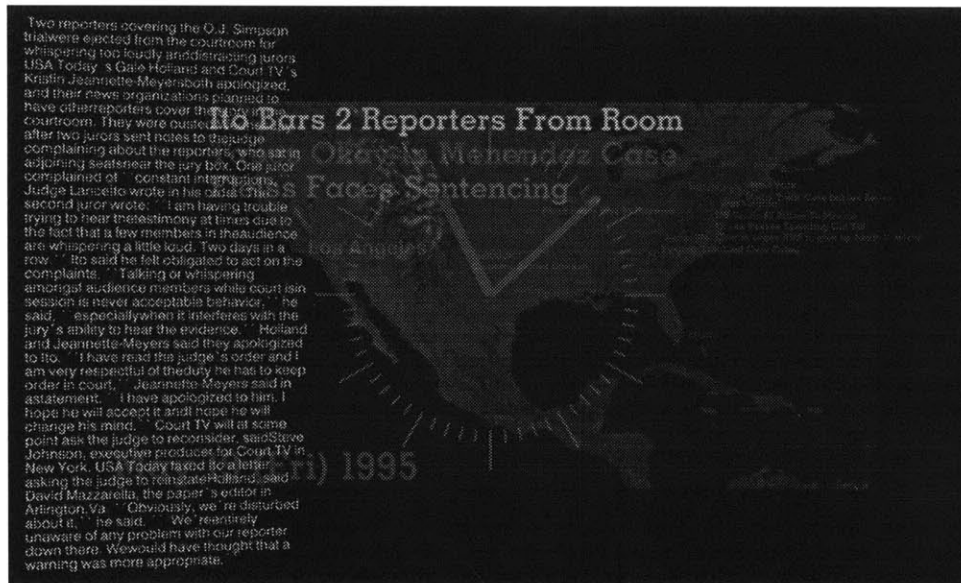


Figure 6.6. A scene after a headline “Two Bars 2 Reporters From Room” is selected. The headline informed its story agent (which was invisible), and in turn, the story agent presented itself. While the story is presented, the headline keeps flashing (between orange and green).

In addition, notice that some new news stories have come into other cities, while the reader has been interacting with the news articles in Los Angeles. For example, there is a headline agent just arriving in New York.

When a reader selects a headline by clicking its text, it informs its news story agent that it has been selected. As a consequence of this change in its immediate situation, the news story agent, which has been using a strategy of hiding, makes it visible using another strategy (Figure 6.6).

Behaviors of other agents are relatively simple. The Clock Agent finds *current time* based on whether the system is used in quick review mode or real-time mode, and displays it in the form of clock. It also changes the color of its clock every hour based on the previously chosen set of twenty four colors around a hue circle. The Date Agent keeps checking current date and displays it in the form of a text. The Map Agent simply keeps displaying a map. In an earlier design, the Map Agent changed the brightness of the map based on the time of a day; however, I decided not to use that strategy since the color coding scheme of other agents became difficult to maintain.

The agents in the Dynamic News Display use various strategies in order to accomplish their performance presented above. In order to see how their behaviors are described, let us closely look at the specification for the Placename Agent and the Headline Agent.

The most general strategy for the Placename Agent is *pn-top-strategy*<sup>2</sup> and it is described as follows;

2. *pn* and *hl* are prefixes that stand for placename and headline, and are used to identify the owner of each strategy and action in the document.

NS1 **pn-top-strategy:**  
*while a system is running*  
*if there is any news stories,*  
*use **pn-with-headline-strategy***  
*otherwise,*  
*use **pn-no-headline-strategy***

This strategy is used by the Placename Agent as long as the system is running, and it provides the agent with the ability to chose either **pn-with-headline-strategy** or **pn-no-headline-strategy** depending on whether or not there are any news stories associated with it. **pn-no-headline-strategy** and **pn-with-headline-strategy** are defined as follows:

NS2 **pn-no-headline-strategy:**  
*if I am using a set of right formal dimensions for no-headline situation,*  
***do nothing***  
*otherwise,*  
*use **pn-change-to-normal-form-action***

NS3 **pn-with-headline-strategy:**  
*if I am selected (clicked once),*  
*use **pn-selected-strategy***  
*otherwise,*  
*use **pn-not-selected-strategy***

If there is no news article associated with the Placename Agent, it chooses **pn-no-headline-strategy**, which simply makes the agent see if it is using the right form for the normal situation where there is no associated headline. If it is using the right form, it decides to do nothing, and otherwise it uses **pn-change-to-normal-form-action**, to change its color and typography to its normal form. For example, the Placename Agent uses this action after all the associated headline agents terminates themselves.

On the other hand, when there is one or more associated news articles, the Placename Agent uses **pn-with-headline-strategy**. This strategy makes the agent check if it is selected by a reader. If it is selected, the agent uses its sub-strategy, **pn-selected-strategy** and otherwise, it uses another sub-strategy, **pn-not-selected-strategy**, as shown below:

NS4 **pn-selected-strategy:**  
*if I am using a set of right formal dimensions for selected situation,*  
*use **pn-inform-headlines-my-status-action***  
*otherwise,*  
*perform **pn-change-to-selected-form-action***

NS5 **pn-not-selected-strategy:**  
*If I am using a set of right formal dimensions for with-headline situation,*  
*use **pn-inform-headlines-my-status-action***  
*otherwise,*  
*perform **pn-change-to-form-with-headlines-action***

**pn-selected-strategy** and **pn-not-selected-strategy** are similar strategies, in that both of these strategies make the agent check if it is using an appropriate form for its immediate situation. And if it is not using the right form, the agent takes a strategy to change its form appropriately. Otherwise, if the agent is using an appropriate

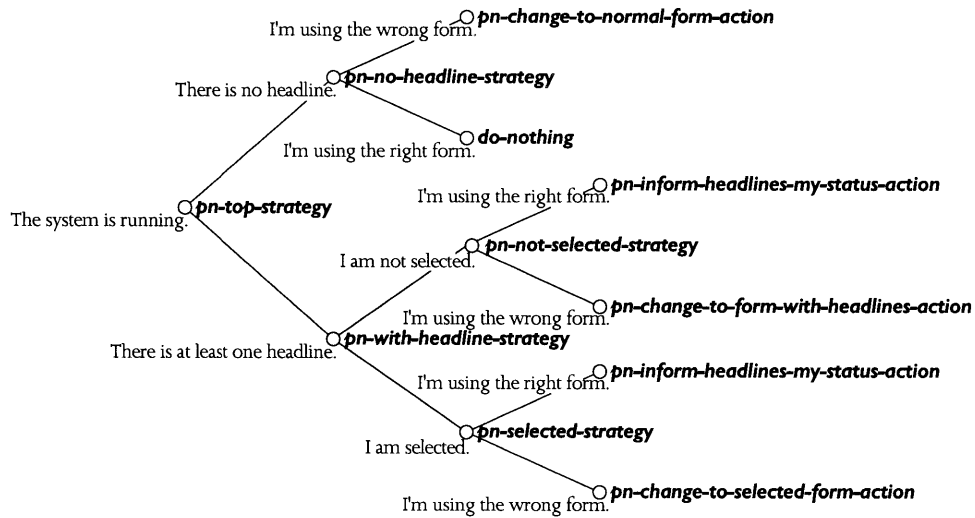


Figure 6.7. Hierarchy of situations for the placename agent.

form, it informs its associated headlines whether a cursor is on its text (placename label), selected (clicked once), or neither of these two. The placename for Los Angeles has made its text larger and bright yellow, so as to indicate its selection Figure 6.5. As described above, **pn-change-to-selected-from-action** (NS4), was used to achieve this action. Figure 6.8 presents a simplified view of this action.

The creation of these strategies and actions took an iterative process. The Placename Agent's strategy started from a relatively simple one and grew as various situations were identified in a process of simulating dynamic design solutions. Figure 6.7 summarizes the hierarchy of situations and associated strategies for the Placename Agent.

Having seen how the behavior of the Placename Agent is described, let us now change our focus of attention to the Headline Agent. Here is the top level strategy for the Headline Agent:

NS6 **hl-top-strategy:**  
 use **hl-introduce-strategy**, then  
 use **hl-while-it-is-alive-strategy**, then  
 use **terminate-myself-strategy**

**hl-top-strategy** consists of three strategies that are used in sequence. The first strategy used by the Headline Agent right after its creation is **hl-introduce-strategy**. Using **hl-introduce-strategy**, the agent presents its text close to a viewpoint in z dimension (Figure 6.9.a) using white. Then, right after that, it gradually aligns its text to its associated placename (Figure 6.9.b-c). If its placename is already selected, it gradually moves its text to the upper left position.

NS7 **hl-introduce-strategy:**  
 If my placename is selected now,  
 use **hl-introduce-strategy-when-placename-selected**  
 otherwise,  
 use **hl-introduce-strategy-normal**

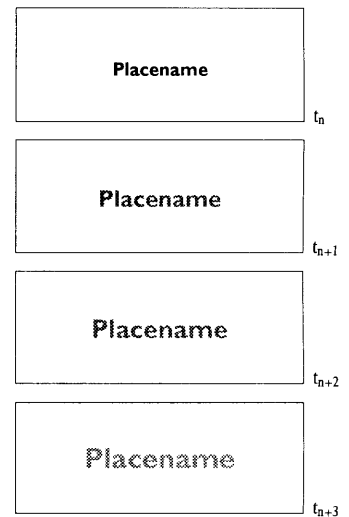


Figure 6.8. A simplified view of **pn-change-to-selected-form-action**.



interacting with its associated placename. The information about this reader's interaction with its placename is informed by the placename agent using ***pn-inform-headlines-my-status-action*** (NS4,5). ***hl-while-it-is-alive-strategy*** is described as the following:

NS8 ***hl-while-it-is-alive-strategy***:  
*While my age is not older than the maximum age the user specified,  
 if my placename agent is focused (a cursor is over the placename),  
 use ***hl-placename-focused-strategy***  
 else, if my placename agent is selected,  
 use ***hl-placename-selected-strategy***  
 otherwise,  
 use ***hl-placename-not-focused-strategy****

Using this strategy, the Headline Agent determines which one of ***hl-placename-focused-strategy***, ***hl-placename-selected-strategy***, or ***hl-placename-not-focused-strategy*** to use, depending on whether its placename is being focused (i.e., The cursor is on top of its placename text.), selected (i.e., It is clicked, but not deselected by the second click.), or neither of these. ***hl-not-focused-strategy*** simply makes the headline agent able to maintain the alignment and distance to its next headline or to its placename. ***hl-focused-strategy*** provides the headline agent with the ability to gradually increase its text size, and to maintain the alignment proportionally the same as ***hl-not-focused-strategy***, while a reader is holding a cursor on top of its placename. Notice that in the strategy definitions below, ***hl-placename-normal-strategy*** (NS11) and ***hl-update-form-by-age-strategy*** are shared by ***hl-placename-not-focused-strategy*** and ***hl-placename-focused-strategy***. ***hl-placename-normal-strategy*** takes care of the alignment. ***hl-update-form-by-age-strategy*** is a simple strategy that merely changes the translucency of a headline text based on the age of the news article associated with the Headline Agent.

NS9 ***hl-placename-not-focused-strategy***:  
*if I my formal dimensions are right for normal-situation,  
 use ***hl-update-form-by-age-strategy*** and  
***hl-placename-normal-strategy*** in sequence  
 otherwise,  
 use ***hl-change-to-normal-form-strategy****

NS10 ***hl-placename-focused-strategy***:  
*if I my formal dimensions are right for focused-situation,  
 use ***hl-update-form-by-age-strategy*** and  
***hl-placename-normal-strategy*** in sequence  
 otherwise,  
 use ***hl-change-to-focused-form-strategy****

NS11 ***hl-placename-normal-strategy***:  
*if I am at the right position,  
 if I my news story is important,  
 use ***hl-when-important-strategy***  
 otherwise,  
***do-nothing***  
 otherwise, (if I am not at the right position)  
 if there is a newer article than myself at the same place,  
 use ***hl-align-to-next-headline-strategy***  
 otherwise,  
 use ***hl-align-to-placename-strategy****

***hl-placename-selected-strategy***, used by ***hl-while-it-is-alive-strategy*** (NS8), provides the Headline Agent with ability to increase its text size (larger than the focused size),

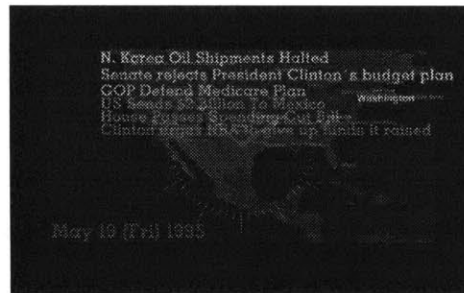
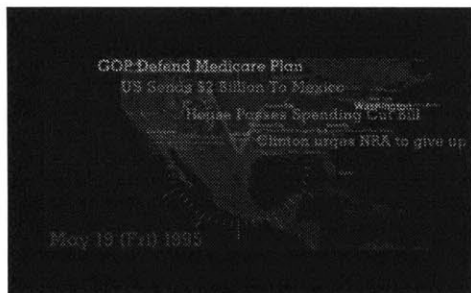
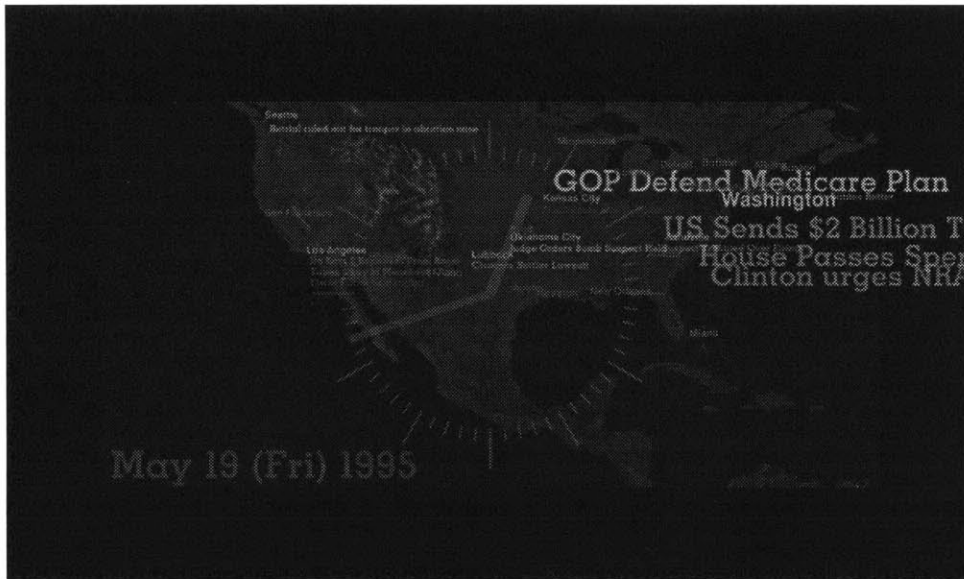


Figure 6.11. A sequence of screen images showing how headlines migrate to their 'reading' position. Simple instructions to follow the next headline creates an wave-like shape as an emergent form.

and move its text to an upper left position as shown in Figure 6.11. **hl-placename-selected-strategy** is described as the following:

- NS12 **hl-placename-selected-strategy:**  
*if my formal dimensions are right for this situation (placename is selected)*  
*if I am selected (clicked once),*  
 use **hl-present-story-strategy**  
*otherwise, (I am not selected)*  
 use **hl-update-form-by-age-strategy** and  
**hl-placename-normal-strategy** in sequence  
*otherwise,*  
 use **hl-change-to-selected-form-strategy**
- NS13 **hl-present-story-strategy:**  
*if I am just clicked,*  
 use **hl-newly-clicked-strategy**  
*otherwise,*  
 use **hl-presenting-story-strategy**

In this situation (i.e., when the Headline Agent's associated placename is selected), the Headline Agent first tries to change its size and location using **hl-change-to-selected-form-strategy**. If a Headline Agent is the newest in the group, it simply moves to the upper left position. If a Headline Agent is not the newest one, it simply

takes an action to align itself to its next headline, which is the same action used in the previous strategies. The gradual movement of the newest headline creates an interesting emergent wave-like form shown in **Figure 6.11**.

After the Headline Agent places itself at the right position, it is now sensitive to a reader's mouse click. When the Headline Agent is selected, it uses **hl-present-story-strategy**, to inform its news story agent about its selection using **hl-newly-clicked-strategy**. Simultaneously, it starts flashing its color between green and orange using **hl-presenting-story-strategy**, indicating it is the headline of a story being presented, while observing if there is a second click or not. If the Headline Agent recognizes a second click while its story is being presented by its news story agent, it informs the story agent of its de-selection and stops its flashing.

In addition, notice that in **Figure 6.6** texts representing placenames and headlines, other than the selected headline and its news story, are highly translucent. When the Headline Agent and the Placename Agent notice a situation (using their sensors) where a news story is being presented (other than its own in the case of the Headline Agent), they use a strategy which tries not to visually distract the reader's attention. For the sake of clarity, this strategy was not included in the strategies introduced in the previous paragraphs. **pn-top-strategy** (NS1) can be redefined as:

NS14 **pn-top-strategy:**  
*while a system is running*  
*if there is any news story being presented,*  
*if I am not using high translucency*  
*use pn-try-not-to-distract-strategy*  
*otherwise,*  
*use pn-normal-strategy*  
*otherwise,*  
*if I am using normal translucency*  
*use pn-normal-strategy*  
*otherwise,*  
*use pn-change-to-normal-translucency-strategy*

NS15 **pn-normal-strategy:**  
*if there is any news stories,*  
*use pn-with-headline-strategy*  
*otherwise,*  
*use pn-no-headline-strategy*

**hl-placename-normal-strategy** (NS11) can be redefined as:

NS16 **hl-placename-normal-strategy:**  
*if I am at the right position,*  
*if my news story is important,*  
*use hl-when-important-strategy*  
*otherwise,*  
*use hl-hide-or-normal-translucency-strategy*  
*otherwise, (if I am not at the right position)*  
*if there is a newer article than myself at the same place,*  
*use hl-align-to-next-headline-strategy*  
*otherwise,*  
*use hl-align-to-placename-strategy*





Figure 6.12. A scene right after a reader asked for news articles related to terrorism. Notice that the headlines at Oklahoma City and Washington D.C. are highlighted using bright color.

NS17 **hl-hide-or-normal-translucency-strategy:**  
*If there is a news story other than my story is being presented,*  
*use **hl-hide-strategy***  
*else if I am not using a normal translucency,*  
*use **hl-change-to-normal-color-strategy***  
*otherwise,*  
***do nothing***

Recall that the news system can use a user specified keyword to identify the importance of individual news stories. Figure 6.12 shows a scene after a reader asked for news articles related to “terrorism.” Those headline agents whose news stories with high importance values assigned by the system use a strategy that provides those agents with an ability to indicate their importance by using the color of their text. Notice that one article in Kansas City, two in Oklahoma City, and another in Washington D.C. are indicated by a slightly brighter color. **hl-placename-normal-strategy** (NS11) provides an ability to achieve this performance, using **hl-when-important-strategy**.

### 6.2.5. Summary

The dynamic nature of news databases has provided an interesting design problem, and I have demonstrated the use of the dynamic model of design in solving it. The solution is described as a collection of design agents, each of which are responsible for presenting its information. Their abilities are specified by using the multiagent model—i.e., strategies, actions, sensors, states, and physical realization. Actions are described based on the abstraction for temporal forms.

Now, imagine the design of typical a newspaper. The model demonstrated here clearly presents a different method for a designer’s approach to a design problem, when compared to a rather traditional model of designing. The Model of Dynamic Design has provided a natural means to describe a dynamic design solution in a decentralized manner. Descriptions of each design agent are relatively simple;

however, their parallel, but collaborative, activities have generated meaningful responses to the changes in the information and the reader's intention. The strategy to avoid visual clutter used by the Placename and Headline Agents while a news story agent is presenting its text is an example of such collaboration. The strategy for headline agents to maintain their alignments is another example of simple strategy that generates an interesting group movement representing their relationship.

### 6.3. Electronic Mail Display

#### 6.3.1. Design Problem

The problem in this case study is to design a visual interface for an electronic mail system, eMail-Display. The system provides users with information that typical email systems support (e.g., sender, subject, and time). However, this design experiment limited the functionality of the interface only to reading, and not writing. The interface must represent an arrival of a new message, reply-replied relationships among messages, whether or not a message is read by the user, and the number of mail messages.

#### 6.3.2. Design Intention

Communicative goals for this project are similar to that of typical email systems. I intended to create a visual interface that can present roughly ten to fifteen messages at once, each of which is showing its sender's name and its subject. In addition, I intended to make the interface playful, avoiding conservative structured layout, by using interesting temporal forms and vivid colors.

#### 6.3.3. Decomposition

The information is decomposed into the following design agents (Figure 6.13): Sender Agent, Subject Agent, Message Agent, Clock Agent, Number of Messages Agent (NumMsg Agent), Reading Mode Switch Agent (Switch Agent), and Date Agent

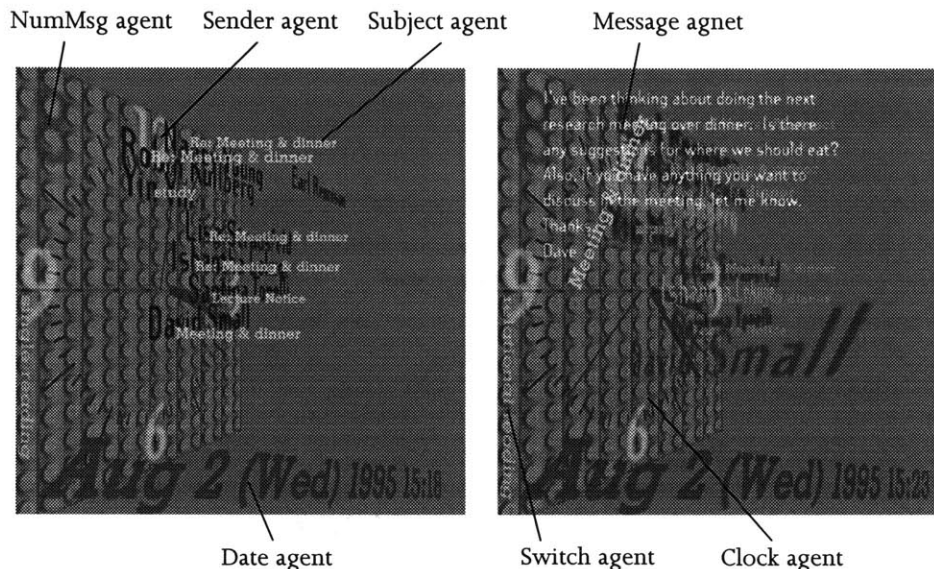


Figure 6.13. Seven design agents are indicated in a screenshot of the Email Display.

Agent. Similar to the decomposition of the news article in the Dynamic News Display, a mail message is decomposed into three design agents: a Sender Agent, a Subject Agent, and a Message Agent. In addition to their obvious roles that can be implied by their names, these three agents have additional communicative roles. First, I decided to make the Sender Agent responsible for representing whether or not its message is read, the temporal relationship to other messages, and reply-replied relationships. The Sender Agent is also responsible for informing its associated message agent when it is clicked by a reader. The Subject Agent visually maintains its relationship to its sender name. The Message Agent presents its text when its sender is clicked.

The roles of the Clock Agent, the Date Agent, and the NumMessage Agent should be implied by their names. The roles of the Switch Agent are to indicate the current *reading mode* of the system and to switch the reading mode between single and relational (discussed later) when it is clicked.

#### 6.3.4. Agents' behaviors

Similar to the Dynamic News Display, the design agents are realized using three dimensional space. However, unlike the News Display, the view point is rotated 45 degrees clockwise around the y axis, creating a sense of perspective (Figure 6.14). A clock agent serves as a background, in addition to its basic function of showing time. It also uses a pattern of yellow LEGO blocks, to create a playful atmosphere. Unlike other graphical elements, which use single realization element such as text, the clock agent uses multiple parts—numbers, the LEGO background—in addition to the clock itself.

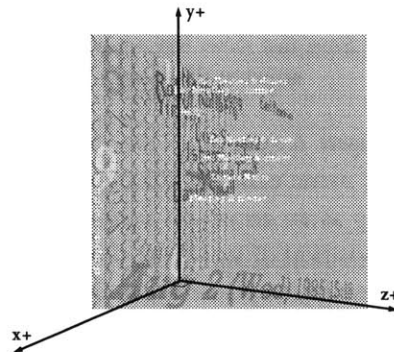
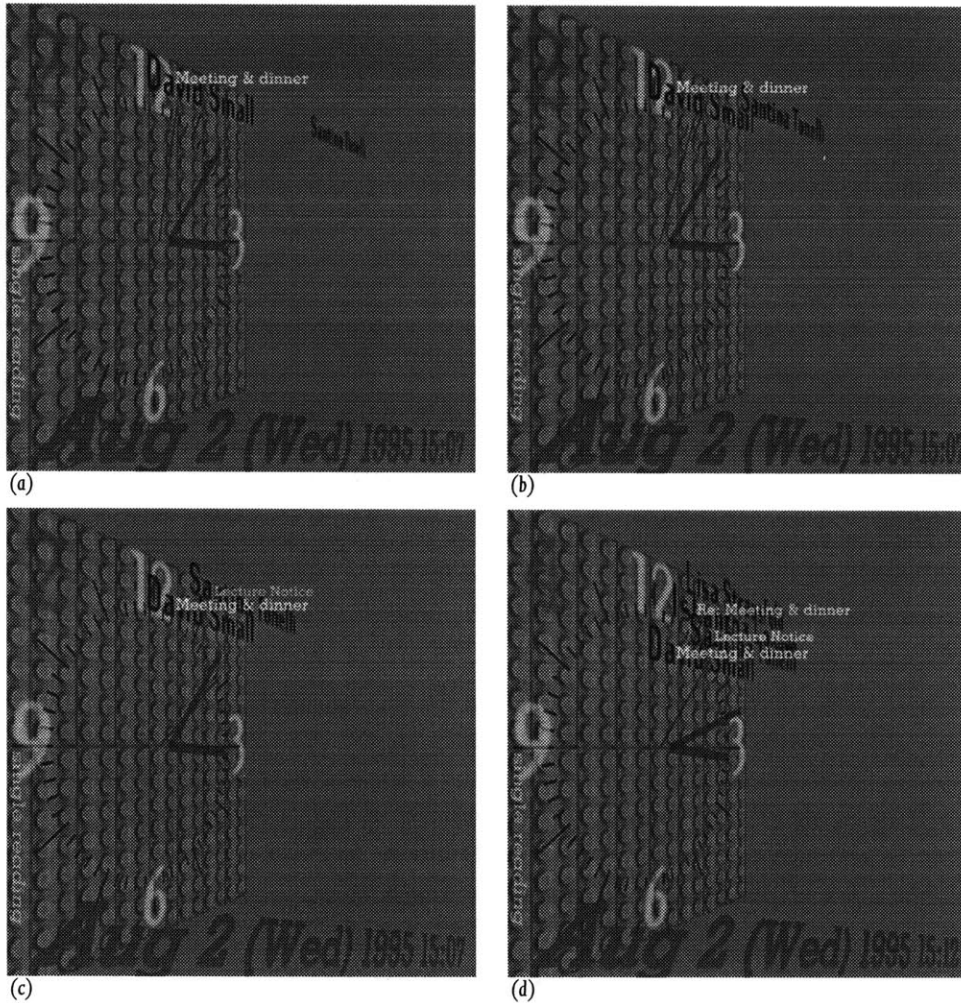


Figure 6.14. The design solution is realized in three dimensional space. The clock agent places its realization on x-y plane, the view point is rotated 45 degrees clockwise around the y axis, and its level is set at the center of the clock.

When a new email message arrives in the mail system, the Sender Agent that is responsible for the message presents its name text at far right in x axis and at about the height of the clock in y axis, and flies it towards the clock. The Sender Agent randomly chooses the destination of its text somewhere between the center and



**Figure 6.15.** A sequence of scenes that shows how a new mail message is presented when it arrives at the system.

right edge of the clock (Figure 6.15.a, b). This random positioning in  $x$  axis is intended to make a sender's name easier to distinguish from adjacent ones, as well as to create a playful zig-zag as an emergent form. After arriving at the clock, the sender agent adjusts the  $y$  position of its text at the top of the clock if it is responsible for the most recent message, or places it just below the next sender text otherwise (Figure 6.15.c). As the text of the Sender Agent arrives at the clock, its associated subject agent coordinates its action such that its text gradually appears next to the sender text facing towards the view point. (Figure 6.15.d). In addition, notice that in Figure 6.15, the message counting agent changes its text by noticing the change in the number of current messages.

The Sender Agent is sensitive to a reader's clicking, and if its text is clicked, it informs its associated message agent and subject agent about its selection. After receiving a message from its associated sender agent, the Message Agent recognizes a new situation, and uses a different strategy to present its text in front of other graphical elements facing towards the view point (Figure 6.16). The Message Agent

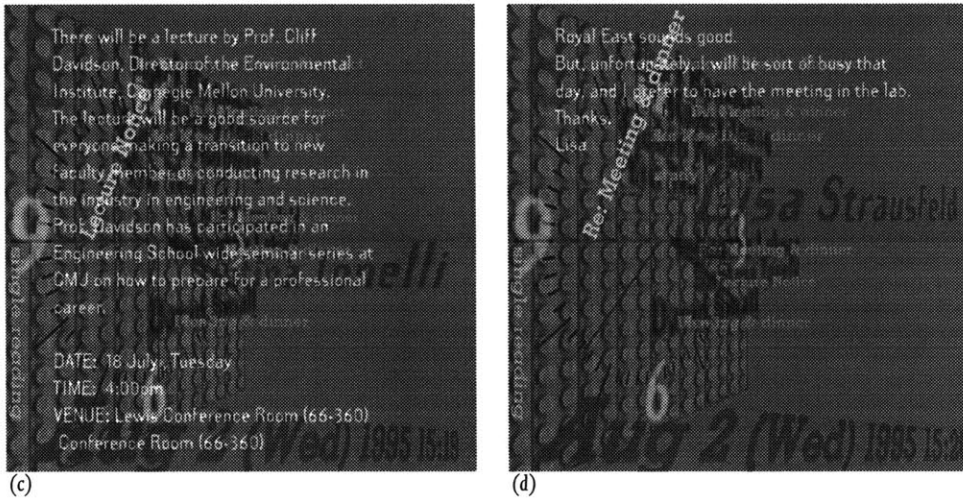


Figure 6.16. Two example scenes while a reader is reading a message content. Notice text presented by the subject agents is moved and rotated; text presented by the sender agents is distorted (dancing).

keeps its text there until a reader clicks it. Simultaneously, when the Sender Agent is clicked, it faces to the view point, becomes larger and translucent, and starts “dancing” in the background. The subject agent associated with the selected sender agent places its text just behind the message and rotates it 60 degrees counter clockwise in order to make it readable. While a message text is presented, in order to make the message easier to read, the other sender agents make their text defocused (which decreases contrast) and other associated subject agents make their text highly translucent.

Now, let us closely look at some of the strategies for the design agents. First, the following is the top level strategy for the Sender Agent.

- ES1 **snd-top-strategy:**  
 use **snd-introduce-strategy**, then  
 use **snd-waiting-strategy**  
 use **snd-terminate-strategy**

The Sender Agent uses **snd-introduce-strategy** to present itself as shown in Figure 6.15.a-b. **snd-introduce-strategy** is a simple strategy that uses a composite action as shown in Figure 6.17. This action uses two partially overlapping formal phrases: one moving from the initial place to the clock, another squashing and stretching as it stops (Figure 6.18). The design intention of this action is influenced by a technique often used in traditional animation in order to provide a viewer a sense of life

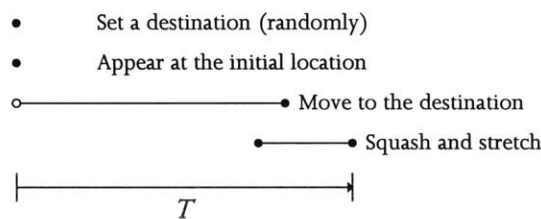


Figure 6.17. A schematic diagram of a composite temporal form that is used as an introductory action for the sender agent.

[Thomas & Johnston 1981][Jones 1989]. As the Sender Agent finishes performing its introductory action, its associated Subject Agent introduces its text by gradually showing the text right next to the name of the sender, using its introductory strategy (Figure 6.19).

After introducing its text, the Sender Agent uses **snd-wait-strategy** until a reader deletes it (by clicking using a right mouse button, instead of a normal left button.).

ES2 **snd-wait-strategy:**  
 While I am not deleted by a reader,  
 if I am just selected  
 use **snd-just-selected-strategy**  
 else if a reader is reading my message  
 use **snd-while-reading-message-strategy**  
 otherwise,  
 use **snd-normal-strategy**

ES3 **snd-just-selected-strategy**  
 use **snd-perform-just-selected-action**

If the Sender Agent is selected, **snd-just-selected-strategy** provides the agent with the ability to remember its selection (by changing its state), and to inform its associated message agent and the subject agent of its selection. Simultaneously, the Sender Agent increases the translucency and size of its text. After receiving a message from its associated senders, the Message Agent, which was hiding, presents its text, and the Subject Agent gradually changes its location and rotation as shown in Figure 6.16. While a reader is reading its associated message content, the Sender Agent uses **snd-while-reading-message-strategy**, which is a simple strategy with one action that makes its text “dance” as shown in Figure 6.16 and 6.20.

I have just presented **snd-just-selected-strategy** and **snd-while-reading-message-strategy**, both of which are active strategies used in response to a reader’s interaction. We now look at a strategy used by other sender agents to whom a reader is not paying attention, namely, **snd-normal-strategy**:

ES4 **snd-normal-strategy:**  
 If a message text other than mine is being presented,  
 if that message is a reply to my message,  
 if I am using the right color for this situation  
 use **snd-hide-behavior-strategy**  
 otherwise,  
 use **snd-change-to-replied-message-color-action**  
 else if I am the reply to that message,  
 if I am using the right color for this situation  
 use **snd-hide-behavior-strategy**  
 otherwise,  
 use **snd-change-to-reply-msg-color-action**  
 otherwise,  
 else if I am using a normal color  
 use **snd-hide-behavior-strategy**  
 otherwise,  
 use **snd-change-to-normal-color-action**

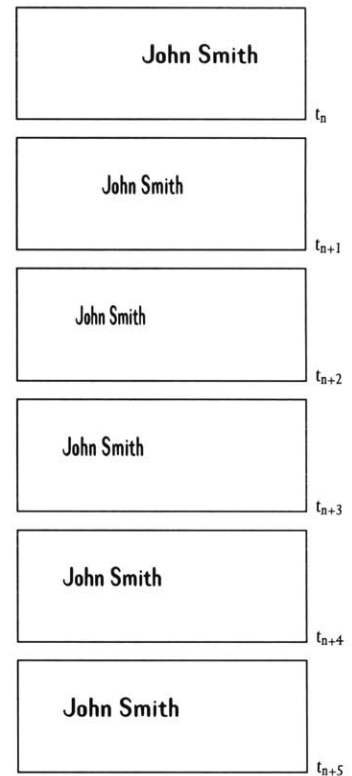


Figure 6.18. The last part of the introductory action for the sender agent.

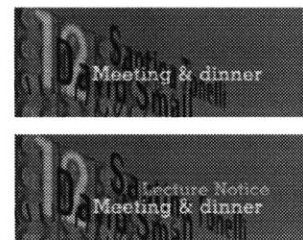


Figure 6.19. The subject agent gradually fade in its text as its associated sender agent finish its introductory action.

This strategy provides the Sender Agent with the ability to distinguish three distinct situations: a situation where the current message is a reply to its message, a situation where its message is the reply to the current message, and a situation where the current message has no relationship to its message. Depending on its situation, the sender agent checks if it is using a correct color: yellow, purple, and dark blue, respectively. And if it is not using a correct color, it changes its text color; otherwise, in all three situations, the sender agent uses **snd-defocus-strategy** to make its text to have less contrast and increase its translucency (Figure 6.21).

I have shown strategies and actions that provide the design agents with abilities to collaborate and solve a continuous and design problem. I have also presented examples of temporal forms used in a communicative fashion. However, individual strategies, as well as group strategies, introduced by now are relatively *short-sighted*. In other words, although a group of agents presents a complex visual message with multiple agents, each of which uses its own strategies in parallel, the presentation remains simply reactive. What if there are more information than that can fit in a screen? What if the designer desires to present a chain of causal relationships in sequence? The next simple example will illustrate how a group of agents can collaborate to create a sequence of presentations.

Most electronic mail systems provide the user with a mechanism to reply to a given message. For example, Figure 6.22 shows a simple diagram of five messages with reply-replied relationships. The message-1 begins the discussion, and the message-2 and 3 replies to it. Later on, message-2 is replied by the message-4 and 5.

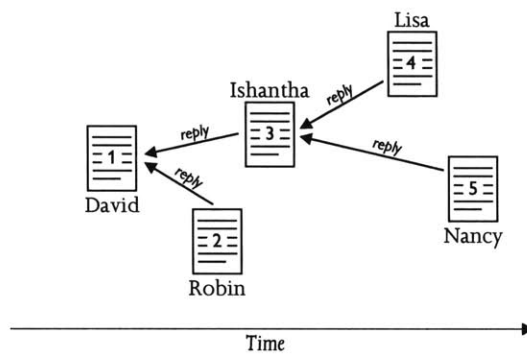


Figure 6.22. A diagram of five email messages with reply-replied relationships.

I have modified **snd-just-selected-strategy** (E53) so that when a sender name is selected, a message to which the selected sender replied (if there is any), the sender's messages itself, and messages that replied to the sender's message (if there is any), are shown in sequence. If there are multiple reply messages, those are ordered by their time of

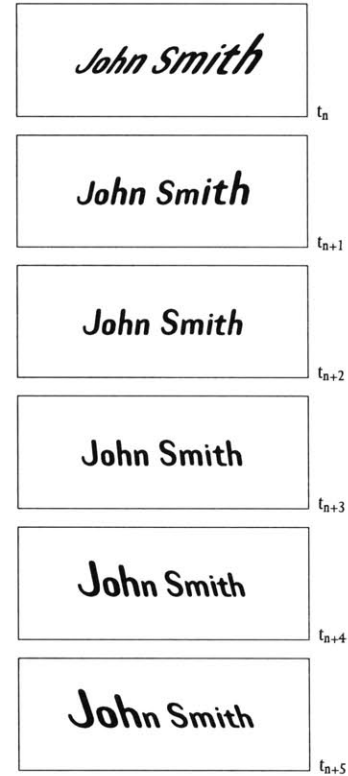


Figure 6.20. The sender agent's dancing action.

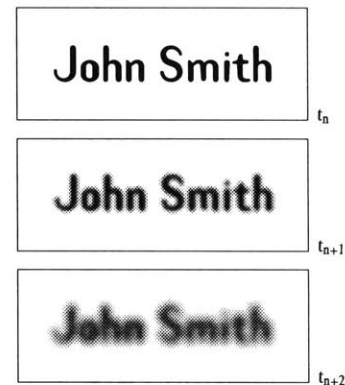


Figure 6.21. A defocusing action used by **snd-defocus-strategy** in order to increase the legibility of a current message.

arrival. For example, if the message-3 is selected, the presentation sequence will be 1-3-4-5. If the message-2 is selected, the presentation sequence will be 1-2. I call this mode of reading “relational reading,” as opposed to the single reading presented earlier. The reading mode can be switched by the Switch Agent, whose simple strategy is to watch for a reader’s cursor and toggle the mode when its text is clicked.

I have described how this sequential presentation is done from a reader’s point of view in the previous paragraph. Now, let us look at this presentation from the agents’ perspective—their behavior description. First, look at the following descriptions for modified **snd-just-selected-strategy**, and a new associated strategy, **snd-presentation-strategy**:

- ES5 **snd-just-selected-strategy**:  
*If the reading mode is single-reading-mode,*  
 perform **snd-just-selected-action**  
*else if I am requested a selection by another agent (not a reader),*  
 perform **snd-just-selected-action**  
*otherwise,*  
 use **snd-presentation-strategy**
- ES6 **snd-presentation-strategy**:  
*If I am a reply to another message,*  
 perform **snd-ask-replied-message-to-present**, and then  
*while the message I replied to is being presented,*  
*if I am using right color as a reply message,*  
 use **snd-while-waiting-to-present-strategy**  
*otherwise,*  
 perform **snd-change-to-reply-message-color-action**
- after the message I replied to is deselected,*  
*or there was no such message;*  
 perform **snd-inform-next-presentator-action**, then,  
 perform **snd-just-selected-action**, then,  
*while a reader is reading my message*  
 use **snd-while-reading-message-strategy**,
- after a reader finishes reading my message,*  
 perform **snd-just-deselected-action**, then,  
*if there is at least one message that replied to my message,*  
 perform **snd-request-reply-messages-to-present**  
*otherwise,*  
**do-nothing**

Using the modified **snd-just-selected-strategy** the Sender Agent checks if the reading mode is single or relational. If it is relational and this selection is not a request from another agent, it uses **snd-presentation-strategy** to start a presentation. **snd-presentation-strategy** provides the agent with the ability to become a temporary leader who coordinates a presentation sequence. First, the agent checks to see if its message is a reply to another message, and if it is, requests the replied message to present. The replied message, in turn, uses **snd-just-selected-strategy**, and select to perform **snd-just-selected-action**, instead of **snd-presentation-strategy** since the selection is requested by another agent. Figure 6.23.a shows a scene after the agent whose sender name is “Ishantha Lokuge” was selected. Since Ishantha’s message was a reply to a message sent by David Small, the agent responsible for presenting Ishantha’s name became a leader, and requested the agent responsible for David’s



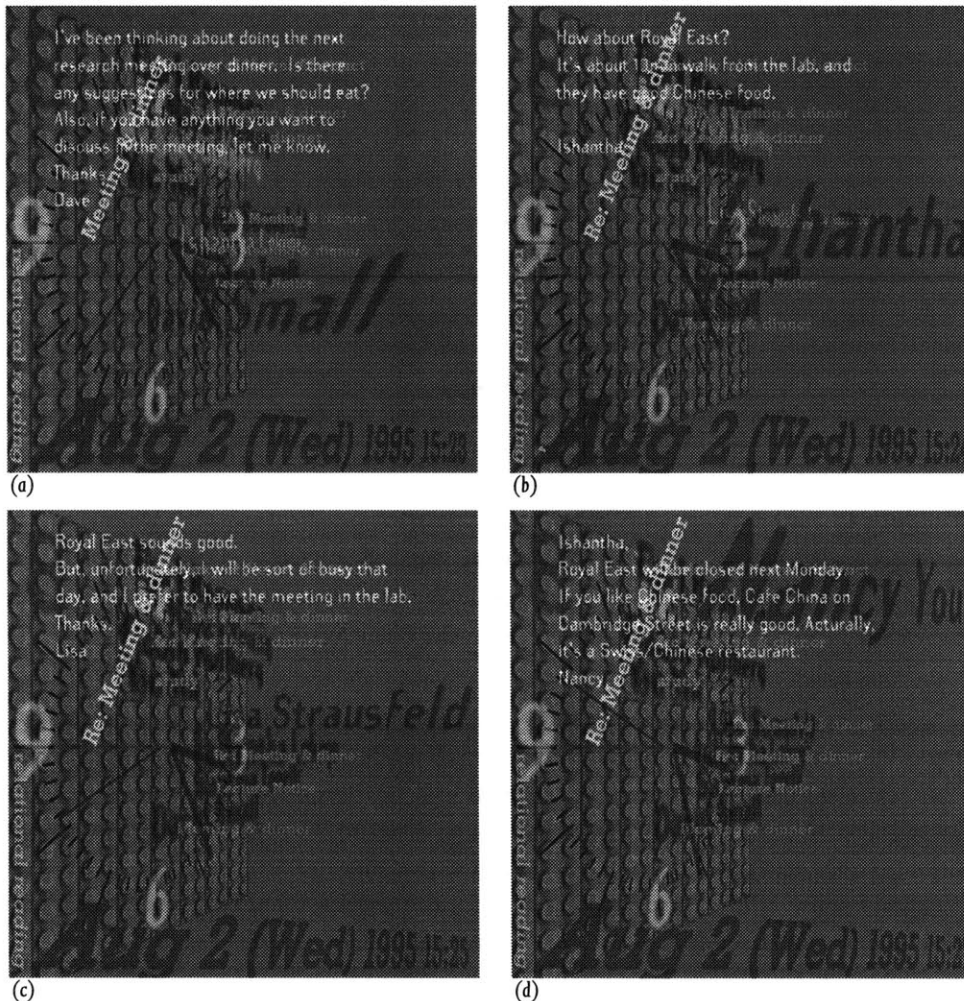
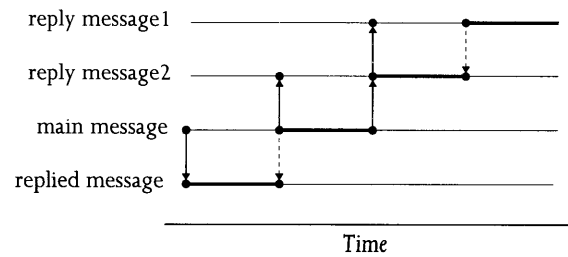


Figure 6.23. A sequence of scenes that shows how a sequential presentation can be generated by a leader agent.

name (**replied agent**) to start presentation. While David's message was being presented, the leader agent (Ishantha) waited and visually indicated the fact that it would perform next, by shrinking and stretching in x axis.

After a reader finished reading David's message (by clicking the message text), the leader agent realized it and began its own presentation (**Figure 6.23.b**). Simultaneously, the leader agent informed **reply agents** (sender agents whose messages are the reply to its message) that they would be requested to present their messages after its presentation is done. This request changed the situation of the reply agents, and the closest reply agent started performing **waiting-action** indicating it would be presenting next. Finally, when a reader finishes reading the message associated with the leader agent (by clicking its message), the leader agent stops its presentation and asks the closest reply agent to start its performance. In **Figure 6.23.a**, there are two reply messages (by Lisa Strausfeld and Nancy Young) sent to Ishantha's message. In general, the first reply agent informs the next reply agent when it is deselected. The

remaining reply agents simply relay their presentation, one by one. **Figure 6.24** shows a historical view of this sequential presentation.



**Figure 6.24.** Historical view of a sequential presentation for the relational reading. Horizontal lines show histories of each agent; black arrows are messages, and dotted arrows are sensing.

Notice also that while this sequential presentation is orchestrated by the leader agent (Ishantha’s), other agents who did not have direct relationships to the leader agent’s message performed in the same manner as usual. For example, they hide when a message text is presented. If there is a reply-replied relationship, the Sender Agent indicates it by color, even though it is not in the chain of presentation.

### 6.3.5. Summary

This second case study experimented with another type of collaboration orchestrated by a leader agent. It has demonstrated that a collection of agents can generate a longer presentation sequence, as opposed to a short-sighted reactive design solutions. I also utilized temporal forms in more meaningful ways in communicating messages. For example, the squash/stretch action for the Sender Agent is used to generate a sense of life; the dancing action for the Sender Agent is used as a “moving icon” expressing a sender’s name for the message currently presented.

## 6.4. Interactive Poetry

### 6.4.1. Design Problem

Both Dynamic News Display and Email Display used practical information as their communication domain. This section presents the use of the Model of Dynamic Design in the rather artistic domain of poetry. The task here is to create a poem with which a reader can interact. I have selected an introductory part of “The Deer” written by Mary Oliver [Oliver 1990] for this experiment.

### 6.4.2. Design Intention

My intention in creating this interactive poem was to present it slowly word by word, generating a sense of quiet atmosphere. I also wanted to play with typographic form and the words in such a way that a reader’s interaction would add some complexity to the original poetic message.

### 6.4.3. Decomposition

The decomposition of this poetic work is extremely simple. There is only one type of design agent—Character Agent (**Figure 6.25**). Every word in the poem is represented by a set of character agents, each of which only knows about its character, and the agent who is responsible for presenting its adjacent character (if there is

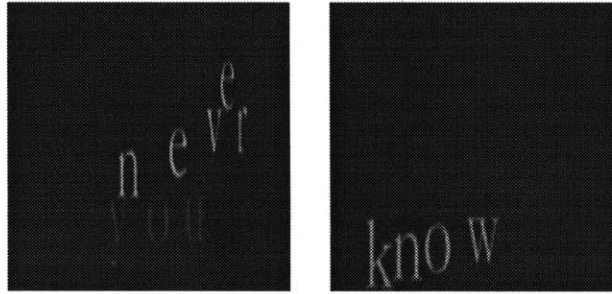


Figure 6.25. Interactive Poetry uses only one type of design agent, the Character Agent.

any). For example, the agent responsible for the second “e” in the word “never” knows about the the agent responsible for the first character “n.”

#### 6.4.4. Agents’ behaviors

The Character Agent’s behavior is simple as well. The Character Agent simply presents its character slightly to the right of the preceding character presented by another agent. If there is no previous character (i.e., first character within a word), it presents its character at the current cursor position controlled by the reader. Words in a poem are designed to appear in sequence. In other words, every component character agent representing a word appears at a particular timing specified in a script. Figure 6.26 shows sequence of scenes from an interactive presentation.

An analogy can be made to a theatrical performance, where there are many groups of dancers, each of which is assigned its showtime (i.e., time to appear and disappear). Each group may have different numbers of dancers, wear different costumes, or sing different songs. When a group shows up on stage, dancers jump into random locations. Then, a leading dancer starts following a spot of light controlled by a lighting person, and other dancers follow each other as they improvise their dance form. When the group’s showtime is over, dancers slowly start to fade out into the backstage, as the next group of dancers fade in.

In this experimental design, the poem is written in a ASCII file, in which each word is tagged with its showtime. When the system creates character agents by reading a file, each agent is assigned a unique showtime.

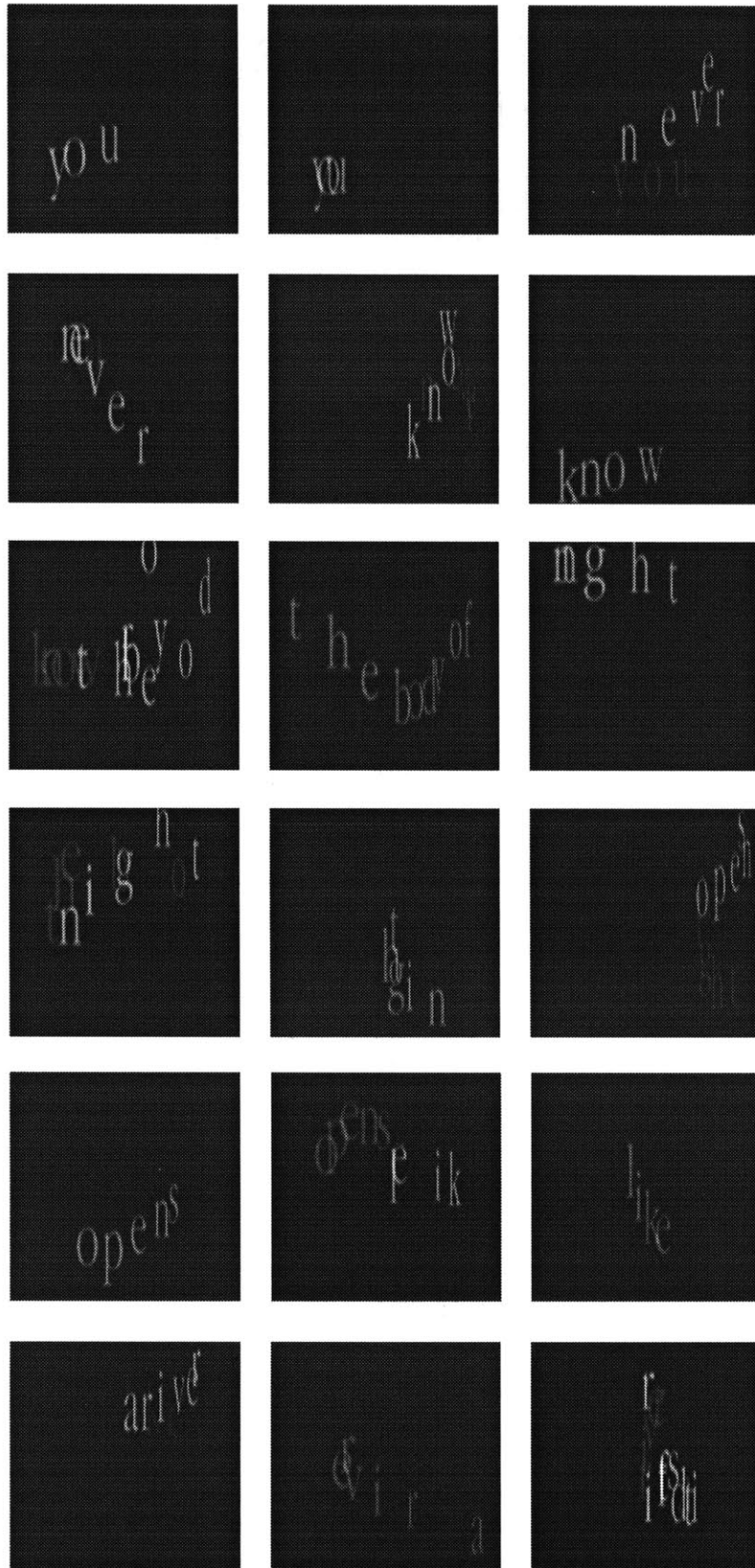
The simple behavior of the character agent can be described as a set of simple strategies and actions. First let us look at the top level strategy:

```

PS1  char-top-strategy:
      While the system is running,
      if it is my show time,
        use char-show-time-strategy
      otherwise,
        use char-back-stage-strategy

```

**char-top-strategy** provide the Character Agent with an ability to determine whether or not it is currently its show time by checking a time sensor, and to choose **char-show-time-strategy** if it is, and **char-back-stage-strategy** otherwise. **char-show-time-strategy** is defined as the following:



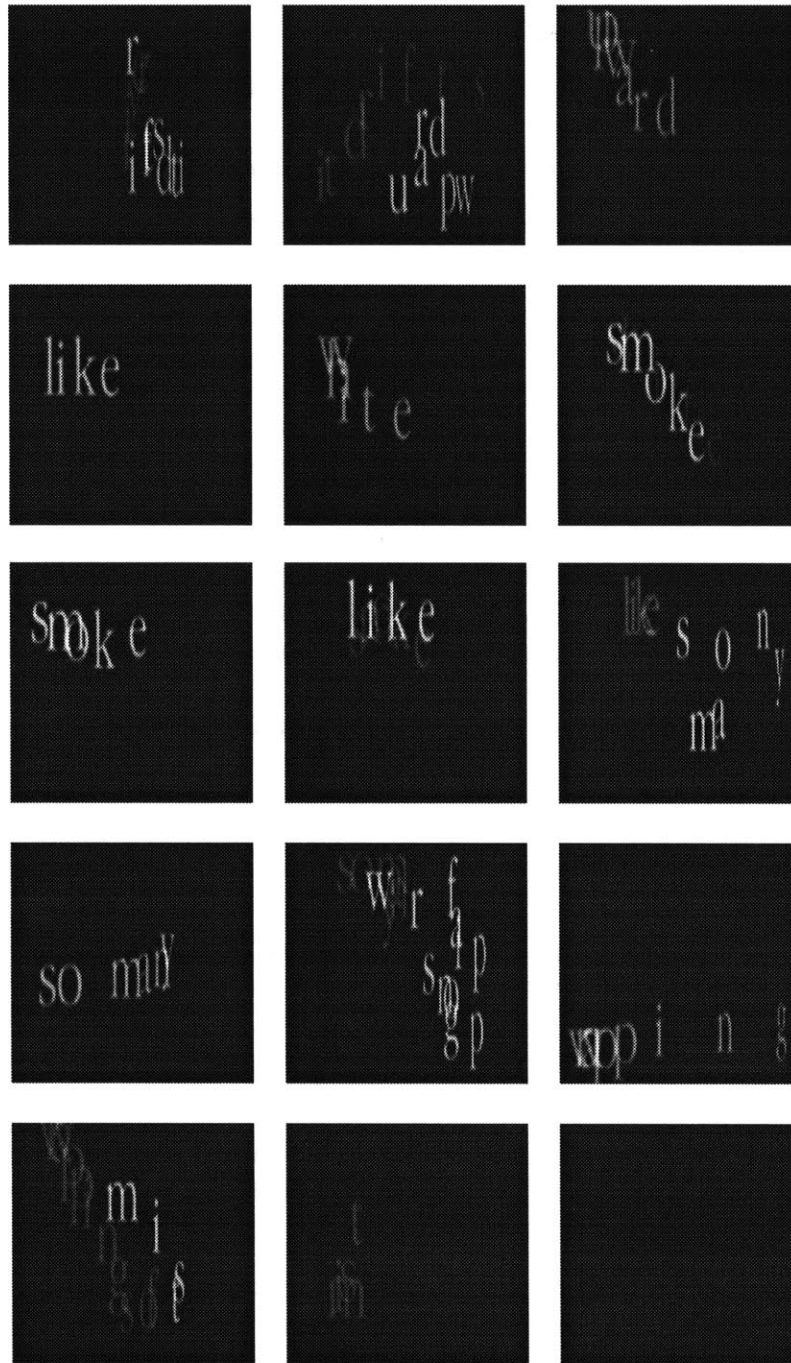


Figure 6.26. A sequence of scenes recorded from an interactive presentation of the poetry.

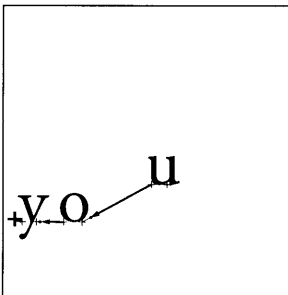
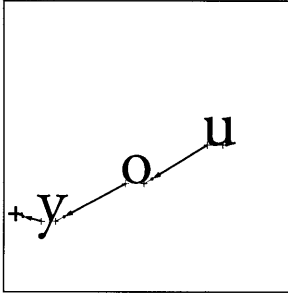
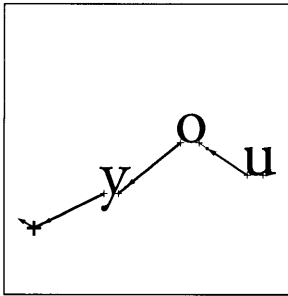


Figure 6.27. A diagrammatic view of char-follow-pre-action and follow-mouse-action.

PS2 **char-show-time-strategy:**  
 If my character is not visible yet,  
   perform **char-become-visible-action**  
 otherwise,  
   if there is a previous character,  
     perform **char-follow-character-action**  
 otherwise,  
   perform **char-follow-cursor-action**

Using **char-show-time-strategy** the character agent checks if it has already made its character visible; and if it hasn't, it performs **char-become-visible-action**, an instant action that makes its character visible in low translucency at a random location within a frame. Otherwise, if it is already visible, the agent performs **char-follow-character-action** or **char-follow-cursor-action**, depending on if it has a previous character or not. **char-follow-character-action** and **char-follow-cursor-action** are almost identical actions except for the target they make the character agent follow. Both actions are instant actions as shown in Figure 6.27. In addition, these two actions are a composite action which changes the horizontal scaling of its character proportional to its x position (Figure 6.28). **char-backstage-strategy** simply hides its character if it is visible, and does nothing otherwise.

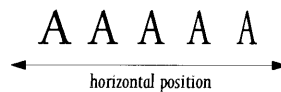


Figure 6.28. A component action (phrase) of **char-follow-character/cursor-action** changes horizontal scale of a character.

#### 6.4.5. Summary

An application of the Model of Dynamic Design has presented in the non-information intensive domain of poetry. It has shown the additional type of design solutions for which the model can be used.

Because of the nature of written language, the poem tends to be linear. However, this dynamic design solution challenges the reading of text that is interactively expressed over time, using form and meaning that are created as a result of interaction. The Model of Dynamic Design also suggests a possibility of parallel and simultaneous presentation of written messages. The description of this dynamic design solution is extremely simple, but it generates seemingly complex emergent forms. This simple description is enabled by the distributed nature of the multiagent model.

### 6.5. Geographic Information Display

The following case study is the preliminary work done during the development of the Model of Dynamic Design, and prior to the implementation of *perForm*. Although the experimental design solution is not implemented using *perForm*, with

the agent description language, it fundamentally shares the dynamic and distributed nature of its solution with that of other studies. The particular design solution presented in this section was called GeoSpace, created by Lokuge, building upon my preliminary implementation [Lokuge & Ishizaki 1995][Ishizaki & Lokuge 1995].

#### **6.5.1. Design Problem**

The design problem here is to create an interface for an interactive geographic information database. The database contains a set of data elements, such as placename, highway segment, hospital, and schools. The system already has a mechanism to compute how important each information element is according to the queries given by a reader. For example, if a reader's query is "Show me Cambridge," the system gradually increases the importance of relevant information elements, such as highway segments connecting to Cambridge, and schools and hospitals within Cambridge. Simultaneously, the importance of other information is decreased depending on their relationship to the current and previous queries. The system gradually changes the importance values to accomplish a smooth contextual transition from one query to another.

#### **6.5.2. Design Intention**

One of the important requirements for the design of the visual interface to this database was to avoid an overwhelmingly dense display, which is often caused by the nature of geographic information (i.e., many overlapping layers). Consequently, I wanted to visually clarify the display to avoid a crowded visual scene while maintaining the over all context and guide the reader's visual attention through the information s/he is interested as s/he asks questions. In order to approach this problem, I have applied traditional visual design techniques such as translucency and typographic size, however in a dynamic fashion.

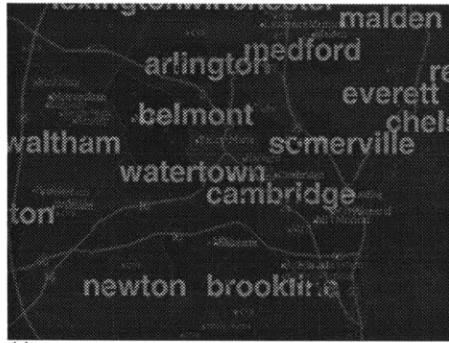
#### **6.5.3. Decomposition**

Although this design solution was not created with *perForm* with an explicit representation of design agents, the following types of design agents were used in this solution: Highway Agent, Hospital Agent, Pharmacy Agent, Placename Agent, Schools Agent, Crime Data Agent. The roles of these design agents are to visually represent what their names imply, representing various types of geographic information. The fundamental task of these agents is to dynamically change their visual representation according to their importance.

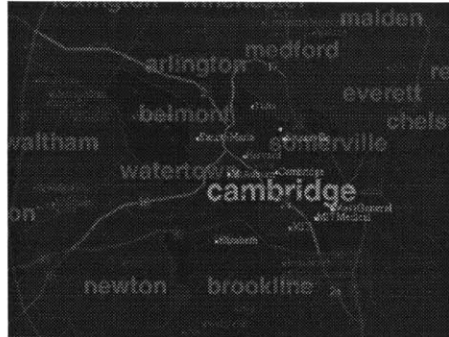
#### **6.5.4. Agents' Behavior**

Behaviors of design agents in this solution are simple. Each type of agent has a unique hue associated with it and changes its translucency proportional to its importance value. Hospital Agent, School Agent, and Placename Agent also have unique typefaces and type sizes associated with them. In addition to the translucency, Placename Agent also changes its typographic size according to its importance value, which is discovered through a sensor.

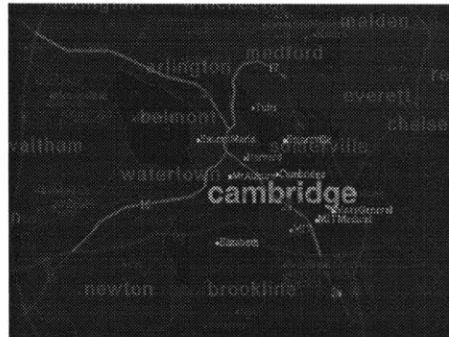
Figure 6.29.a shows the display before any query is asked, showing the visual density of the data space. At this point, each piece of information is equally important. Then, after the query "Where is Cambridge?" is requested by a reader, the system deter-



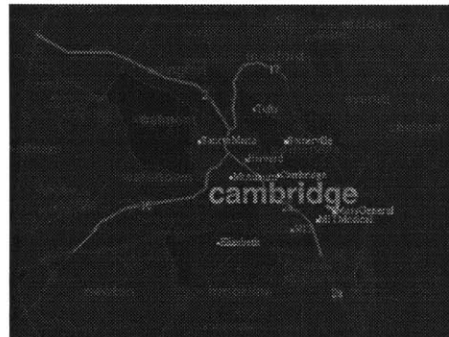
(a)



(b)



(c)



(d)

Figure 6.29. A sequence of scenes after a query “Where is Cambridge” is requested.





### 6.5.5. Summary

The dynamic design solution emerges as a result of the design agents' decentralized activities, reflecting upon the changes in a reader's information seeking goals (i.e., intention). In GeoSpace, there is no global description for how the design solution must be designed. Furthermore, the agents do not use communication for their coordination. Although, each agent is independent and acts only on its local situation, because the system's interpretation on the reader's intention is meaningful, the design solution as a whole emerges as a meaningful expression.

This preliminary study encouraged the possibility of describing a dynamic design solution as a collection of active design agents. Also, it has indicated that a decentralized description can be a natural model for a designer to think "with." However, it has also raised issues that had to be further investigated. Particularly, the concept of formal actions (or temporal form) was weak in this example, in that the agent's physical form was simply manipulated as a function of importance. Without the explicit notion of action, the model did not highlight a temporal form as a meaningful unit for communication. This limitation led to a solution which is relatively static, although the transition between queries is dynamic. For example, there was no easy conceptual framework to support describing a repetitive motion as a label for important information. In addition to actions, the lack of explicit communication as well as strategies made the agents' behaviors short sighted.

Consequently, this preliminary study has led to the development of the Model of Dynamic Design.

## 6.6. Expressive Typography

The following design examples use the abstraction of temporal form as a descriptive structure for specifying the precise form of typographic messages presented over time. The particular design solutions presented here were created by Wong [Wong 1995]. Based on the descriptive language provided by the abstraction, Wong explored the issue of characterizing expressive qualities of temporal typography. These examples particularly demonstrate the complexity of temporal design solutions that can be built up using the abstraction.

Wong did not explicitly use the concept of design agent in solving her design problems. However, each design solution is decomposed (implicitly) into a meaningful set of dynamic entities, each of which is scripted to present a meaningful segment of a message. Thus, although those dynamic design elements were not particularly considered autonomous, I will look at Wong's expressive messages from a perspective of dynamic design—namely design solutions as emergent behavior of a collection of active agents.

### 6.6.1. Design Problem and Intention

The general design problem is to create electronic mail messages that are expressed over time using typography. The intention of design solutions is to convey the tone of voice as well as emotional qualities in temporal forms and their visual interac-

tions, in such a way that the final solution would enrich the meaning of the message. The two messages used in this study is the following:

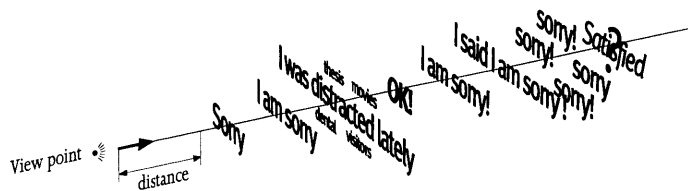
- Message 1**      Sorry.  
                       I am sorry.  
                       I was distracted lately (dental, thesis, visitors, movies).  
                       OK! I am sorry!  
                       I said I am sorry! sorry! sorry! sorry! sorry!  
                       Satisfied?
- Message 2**      Glug! glug! glug! Apologies for being so tardy!  
                       We got into a big crunch that just overwhelmed us.  
                       Hopefully, things are back on track!

### 6.6.2. Decomposition

In these examples, a meaningful phrase or a word is considered an agent. For example in the **message 1**, “Sorry,” “I am,” or “OK!” are agents. In the second message there is an additional agent whose physical realization is a rectangular object.

### 6.6.4. Agents’ Behavior

This section introduces selected scenes from two messages expressed over time, and discusses how they are described by the Model of Dynamic Design. Both messages are presented in a three dimensional space (similar to the Dynamic News Display and the Email Display). But they have different interaction styles. **Message 1** is an interactive message in which a reader uses a mouse button to fly through a three dimensional space, forward as well as backward, as shown in **Figure 6.31**. Each agents senses the distance between the text it is presenting and a viewpoint, and performs expressive formal actions accordingly. On the other hand, **message 2** is not an interactive message. In **message 2**, agents performs their formal actions in the three dimensional space based on their assigned showtime, similar to the poetry example.



**Figure 6.31.** Interactive three dimensional space used in the design of **message 1**.

**Figure 3.32.a** through **c** show a sequence of scenes when two agents responsible for presenting “I am” and “sorry!” are performing. They stay still (doing nothing, but being there) until the viewpoint is far from their text. Then, after the viewpoint is within a critical distance, the agent responsible for “sorry!” starts to stretch its text horizontally, adjusting its form by sensing the distance, as shown in **Figure 6.33**. Similarly, **Figure 3.32.d-f** show a sentence “I said I am sorry!” which is decompose d into four agents, “I,” “said,” “I am,” and “sorry!” Notice that the agent responsible for “said” increases its text size as the viewpoint approaches (**Figure 3.32.e**). Then,

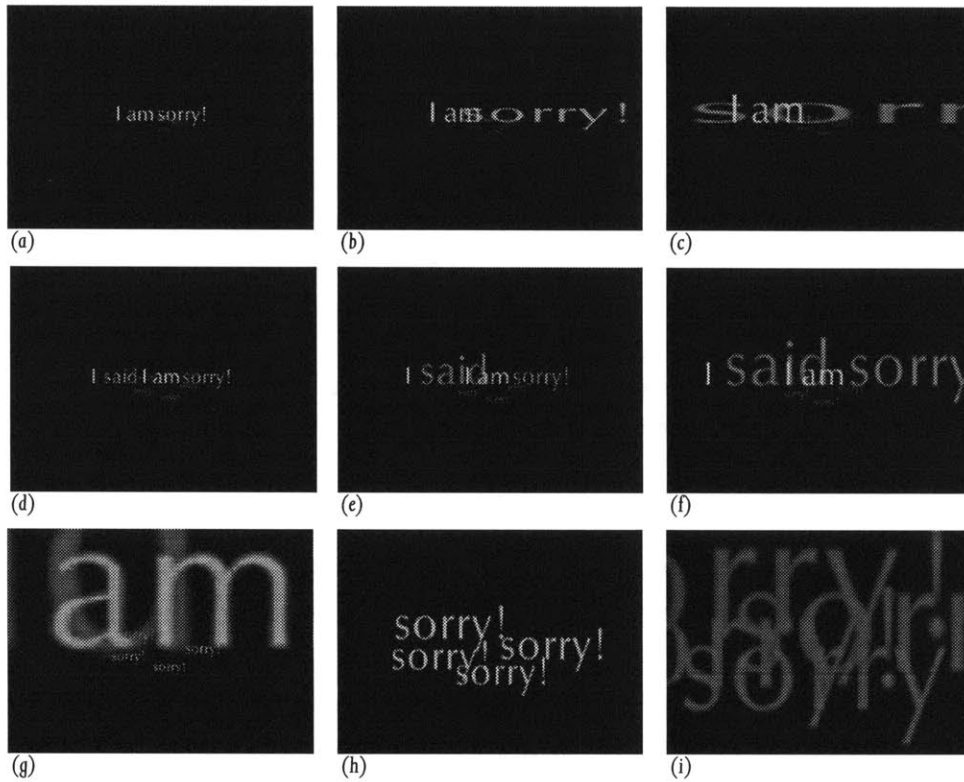


Figure 6.32. A sequence of scenes from an expressive/interactive electronic mail message.

slightly after that, the agent responsible for “sorry” starts performing the same formal action (Figure 3.32.f). Similar to the way we use the same bold typeface in more than two places in a page composition, in a dynamic design, a temporal form can be used in more than two temporal locations. Figure 6.32.g-i show another sequence for “sorry! sorry! sorry!” represented by four design agents. They perform the same action that simply increases the text at different locations, again triggered by the view distance.

Message 2 is expressed without any interaction. Figure 6.34.a through d show a sequence of scenes for “Glug! glug! glug!” that is decomposed into three agents “Glug!” “glug!” and “glug!” These agent

performs the same bouncing action starting at slightly different timing. The action is a fairly complex action which is realized by simulating a box spring, as shown in Figure 6.35.

Final expression of this action is determined by the height and angle of a text when it jumps into the scene.

Figures 6.34.d through f partially show a sequence of scenes for “Hopefully, things are back on track!”

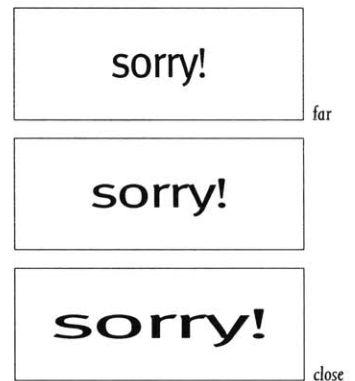


Figure 6.33. Interactive three dimensional space used in the design of message 1.

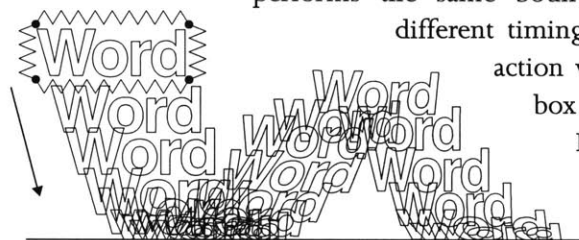


Figure 6.35. An illustration of the bouncing action, simulating the physical motion of a spring box.

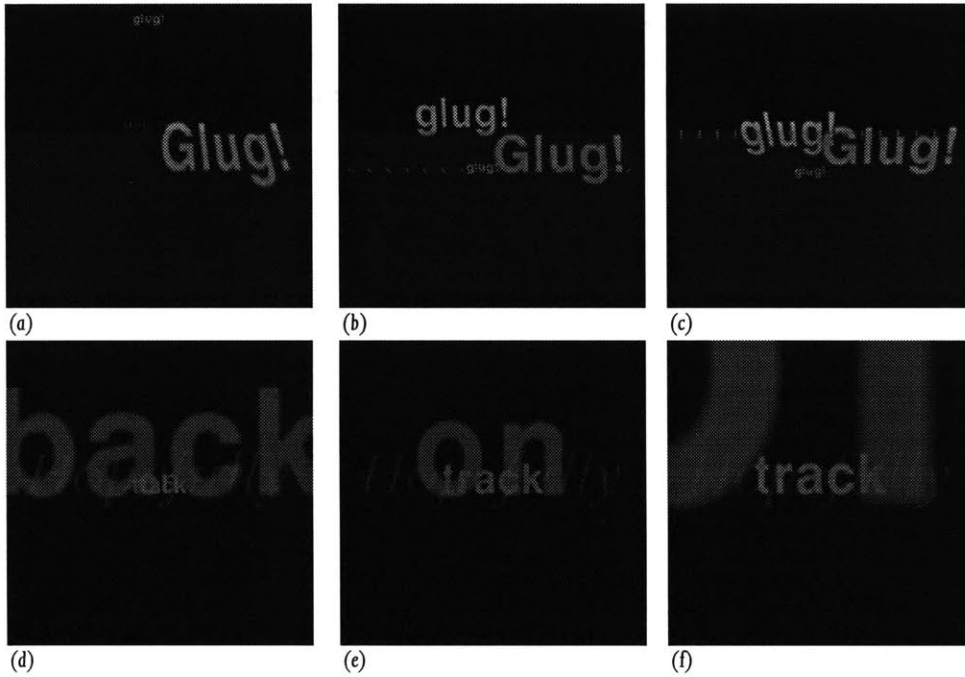


Figure 6.34. A sequence of scenes from an expressive electronic mail message 2.

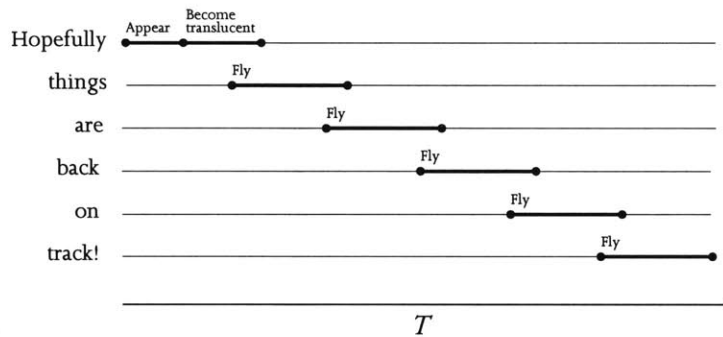


Figure 6.36. A simplified historical diagram of “Hopefully, things are back on track!”

that is decomposed into six agents responsible for “Hopefully,” “things,” “are,” “back,” “on,” and “track!” The figure only shows “Hopefully,” “back,” “on,” and “track.” The agent responsible for “Hopefully” appears first at the medium distance from a view point. Then, soon after it becomes translucent, other agents show their text, using the same action which flies a text from far back through viewpoint. This flying action makes a text fly slowly first, and it increases its speed as it approaches the view point. Figure 6.36 shows a historical view of this presentation.

#### 6.6.5. Summary

I have shown two compositions of temporal forms in the domain of expressive messages. Various temporal forms, such as stretch, grow, and fly actions, have been created for words and phrases and composed over time. Each of these temporal forms are composed of primitive phrases. In the course of designing, the structural

description provided the abstraction of temporal form helped explore different kinds of temporal forms analytically. In addition, the abstractions have become a foundation for the development of Wong's characterization scheme [Wong 1995].

## 7. Reflection

Designing is a conversation between a designer and an artifact—an iterative process of generation and reflection. One postulates a form, reflects upon its appropriateness, and reformulates the form. This process brings about a convergent form, that is, a design solution. Development of design theory is also a conversation, but with an additional participant—a theory. The Model of Dynamic Design is an result of such a reflective three part conversation.

This chapter summarizes the reflective discussions occurred during the process of examining the Model of Dynamic Design with concrete design problems. I first discuss how the hypotheses postulated in Part 1 were reflected upon the design examples, then present design issues raised during the course of developing these solutions.

### 7.1. Reflections

#### 7.1.1. *Dynamic Problems and Solutions*

A role of the Model of Dynamic Design is to enable designers to see design problems in digital communication as fluid entity, and to develop solutions which can continuously adapt to such problems over time. More precisely, it is intended to help designers to understand and structure dynamic and continuous design problems, so as to create design solutions that can respond to changes in the information and the reader's intention.

Using the Model of Dynamic Design, information could be perceived as a collection of meaningful smaller segments of information. Further more, these small information segments could be considered fluid entity that responds to changes in the context. The descriptive model of the design agent strongly influenced the way design problems were analyzed in solving concrete design problems. Unlike traditional perceptions of design where a design problem consists of a set of finite information elements, a design problem was naturally understood as a fluid entity consisting of some number of pieces of information each of which is dynamic. Each

piece of information was understood as having its associated value, e.g., age or importance, that potentially change over time.

### 7.1.2. Form and Content

The notion of active and responsive design agent provided a view that consider content clearly separated from its form. That is, an agent is an abstract entity which is responsible for appropriately presenting a piece of information depending on its immediate context. In other words, the agent is not bound to any particular form, and the agent potentially changes its form over time in order to best achieve its communicative role, which also changes over time. Thus, the model suggested understanding the form as an situated concept that is independent of the content. One would argue that content is inseparable from form, and vice versa. I agree that form is an integral part of the visual message that contributes to communication: thus the interaction of form and content must be carefully treated in design. However, in dynamic design, we must recognize that the same content may have different value at different times, depending on the changes in information itself and the reader's intention. Hence, the same information content may be expressed in different forms, in order to be appropriate to the communication. The abstraction of agent, along with strategy, could emphasize the need for appropriating the form of the content over time.

For example, the importance of a news article is determined relative to other articles (e.g., if some natural disaster happens, other news may become less important), and depends on what a reader is interested at a particular time (e.g., a reader can be interested in the world news at one time, or in terrorism related news at another time). Such changes in context demand changes in their presentation. Traditional design has not emphasized this *change in the value of information*. It was instead natural to assign a fixed communicative role to the content (often by averaging situations!), since the design artifacts must be fixed onto some physical media, such as paper or film. This is no longer true in digital communication. The model of design provides a natural means to separate content from form, enabling a conception of dynamic design solutions that can appropriate *forms* to the changes in the context.

### 7.1.3. Construction of Dynamic Solution

Not only does the Model of Dynamic Design provides a means to perceive a design problem as a fluid entity consisting of smaller information elements, it also provided constructive devices that can be used to describe dynamic design solutions. As shown in the examples, the model covers different levels of abstractions that range from the formal dimensions to an emergent solution. First, the most primitive constituent of a design solution is a dynamic form described by using the abstraction of temporal forms. The abstraction emphasized changes in form as conceptual, but concrete objects that can be manipulated, and used purposefully. Temporal form is then used as a basis for describing actions which the design agent can perform to express its message content. The abstraction of action particularly highlighted the concept of temporal form as a meaningful unit of communication (rather than an interpolation between two static states).



Building upon the abstractions of action and temporal form, the abstraction of strategy, along with the concepts of sensor and message, provide a means of describing dynamic design solutions which can meaningfully respond to the changes in context. The concept of hierarchical situations was found to be useful in localizing dynamic responses of the design agent. The concept of strategy also highlights the dynamic nature of design solutions.

#### **7.1.4. Design Process**

The prescriptive process model presented in Section 4.2 was found to be natural and useful. As discussed earlier, this model is not a linear process; rather it is intended as an iterative process. Throughout the process of creating design examples, I have found the model appropriate. Although, it was not necessary to follow the model step by step, the existence of an explicit model was useful to refer to, as well as to reflect upon one's own design activities.

Not only is the design process iterative, but it was found to be a process of specialization. After the initial decomposition is done, the process tended to begin with describing agents' strategies for relatively general situations. Then, gradually, sub-strategies are created to enable agents to respond to further detailed or exceptional situations. New situations were discovered as the design process progressed, and new strategies as well as new actions were created. In the process, various strategies and actions were often compared to discover appropriate ones.

The process in traditional graphic design is also iterative. A designer draws a sketch of a layout, views it from a short distance (e.g., with his eyes squinted), and refines the sketch. This process often requires pen and paper first, but as a solution matures, more sophisticated production tools are used so as to simulate the final print. In dynamic design, it is hardly possible to sketch dynamic ideas on a paper (at this time). For the design of linear presentation, such as film or video, storyboarding provides an appropriate means for sketching temporal presentation (although it is not perfect). However, in dynamic design, where a design solution must respond to the changes in context, it is difficult to sketch dynamic interactions of active agents. Consequently, in the course of creating design examples, *perForm* was inevitable. In other words, *perForm* was not only used as a tool to describe the complete design solution, it was rather used as a tool with which a designer develops ideas. It could have been impossible to complete the experimental solutions without a tool that allows a simulation of dynamic design solutions.

## **7.2. Design Issues**

### **7.2.1. Scale and Predictability**

There is a question regarding the scale of design problems and predictability of design solutions. Can the proposed model be used to describe large design problems? Can some dynamic design solution become untraceable and uncontrollable? Furthermore, how do you measure the scale of a problem?

The concept of scale is not a one-dimensional concept, and it not only depends on the information being presented, but it also relates to how a designer decomposes the information. Through the course of experimental design, I have identified at least three kinds of scales that relate to each other: the scale of original data set that has to be presented at a time, and the number of design agents active at a time, the number of agent types. The size of data set often directly influences the average number of agents active at a time. However, neither of them are directly related to the number of agent types. The number of agent types is a function of the structure of the original data set (not its size) and the intended use of the design solution.

The predictability is found to be an interesting issue. Predictability here is used to mean both the predictability of a solution from the description, as well as the predictability of the description from a particular solution. Although the design of individual agents is relatively simple, as the number of agents grows and the number of strategies increases, agents' emergent behavior may become unpredictable and incomprehensible. Originally, I hypothesized that with the aid of a multiagent simulation system (i.e., *perForm*) a designer can clearly observe and reason activities of large numbers of agents (although it does not mean that the agents activities are observable for the reader). Through the case studies, I have found that it was true for most cases. Although a resulting behavior, or a solution, often does not express information as I expected, I could easily discover which agent and where it was located in the description of strategies or actions caused the unexpected result. Predictability also did not seem to be strongly influenced by the number of agents.

However, there are some cases where a description became difficult to comprehend. Through the experiments, I discovered that predictability is influenced by the agent's states that are not physically represented. Particularly when an agent's state is changed based on a message or a sensor that is not reflected on the agent's realization, since the changes in states are not perceivable for a designer, it was difficult to foresee a part that is ill-acting. This observation suggests that it would be useful to provide a computational tool which allows a visualization of internal activities among design agents.

#### 7.2.2. Agent's Organization and Types of Design Problem

This research proposes to describe a dynamic solution as an emergent behavior of distributed agents with some temporal leadership, rather than a centralized hierarchy. I consider the creation of a particular organizational structure to be a part of design process—i.e., decomposition. A designer is expected to create an organizational structure by carefully analyzing a problem.

Nonetheless, I have found there are two useful perspectives from which a designer can look at the organization of design agents: types of collaboration and roles of agents within a group. Agents can collaborate either implicitly or explicitly. Implicit collaborations occur as a result of group strategy. Since each agent within a group uses its appropriate strategy as they rehearsed, although there is not direct communication among agents, the emergent solution becomes coherent. For example, headline and placename agents in the Dynamic News Display all become translucent

as soon as some other story agent starts presenting its text. The introductory coordination performed between sender and subject agents is also an example of implicit collaboration.

On the other hand, agents can collaborate by communicating directly. For example, a placename informs associated headline agents if it is focused, selected, or neither. Then, the headline agents in turn use strategy accordingly, such as to change color, or to increase its size. Note that even though there is a direct communication, the headline agent's behavior is independent of the placename, in this example. In other words, an agent which sends a message to another agent does not have to know how exactly it is interpreted. The agent simply trusts the ability of the receiving agent. One disadvantage of direct communication is that when messages are chained, it is often difficult to track what is happening.

The second perspective is the role of an agent with respect to its group. An agent can play a leader's role, and manage the others. An agent can be a temporary leader or a permanent leader. The placename agent in the Dynamic News Display is an example of long-term local leader with respect to its associated headlines. Long-term leaders often have a long life span, and the information they are responsible for is usually static or relatively slow paced in its changes. On the other hand, the sender agent in the E-Mail Display (under relational reading mode) is an example of a temporary leader that orchestrates other agents to create a presentation sequence. Another way to look at a role of an agent is by its communicative role. Communicative role is determined by a kind of message content an agent is presenting. One may ask if an agent is responsible for potentially important information, or if it is always a background.

In addition, using the Model of Dynamic Design, the same collaborative behavior can be described in more than two ways. I consider this an advantage, rather than an incompleteness, of the model. Like any other language, the designer can select the most appropriate description that suits a particular problem at hand, or his or her own design style.

### **7.2.3. Expressing Stories**

One of the disadvantages of reactive agents recognized in Multiagent Systems research is their short-sighted nature. If the proposed theory inherits this short-sightedness, it would be difficult to design a dynamic solution which could present complex messages over longer duration. The abstraction of the group strategies suggested by Singh was intended for a generation of temporally longer solutions—or a story.

There are two types of what I call *story* in a dynamic design solution: scripted story and interactive story. Scripted story is analogous to a partially scripted theatrical performance, or an offense formation in a football game. In the case study of Electric Email Display, I have presented an example of how a presentation sequence can be orchestrated by a leader agent. This scripted story is generally achieved by explicit

communications. However, note that not only does a scripted story involve those agents communicating, it is also supported by other agents simultaneously acting on that situation (i.e., agent in the background).

The second type of story, interactive story, resembles a non-linear reading. Using the strategies, agents can follow the changes in the reader's intention, often expressed by some input device, such as a mouse or voice. For example, in the Dynamic News Display, the sequence of browsing placenames, focusing and selecting a particular placename, reading headlines, and finally reading a particular news story, is nothing but a story. This example solution is achieved by having strategies that are described hierarchically according to its contexts, i.e., a reader's interest. An interactive story can also be achieved by simple reactive agents, like the ones used in GeoSpace. Because of the meaningful changes in information, appropriate set of agents' actions generates an interactive story as response to a reader's queries.

#### **7.2.4. Design Failure**

Although a designer may carefully create a dynamic design solution, there can be a chance that the design solution encounters an unknown situation. It seems there is no guarantee that an individual agent or a multiagent design solution as a whole can avoid failure. For example, if there is an unexpectedly large number of news articles that belong to a particular place, a group of agents may not be able to find a readable solution. One method of solving this problem would be to simply ignore the failure and make the multiagent system wait until the immediate context changes (e.g., by assuming the reader to provide further specific request). One may suggest adding a set of failure-recovery strategies for agents. However, if one could define a situation in which a design solution may fail, it simply becomes another situation. What we are worried about is a situation that the designer could not anticipate at the time of designing.

Obviously, it is impossible to predict a situation that a designer could not anticipate at the time of designing. Then, how do we deal with design failures? Although there is no guarantee, there are some practical methods that reduce the risk by controlling the information content. First, it is important to control types of information the dynamic design needs to solve. For example, the size of data set can be controlled to avoid situations where design agents cannot find any readable solution. If there is information which already has physical quality, such as photographs and maps, controlling the size and proportion of those would also reduce the risk. Finally, and obviously, it is recommended that a dynamic design solution must be tested against many realistic situations in order to discover as many situations as possible.

#### **7.2.5. On Composition**

The abstraction for temporal forms has the potential to provide designers with building blocks for developing further complex temporal expressions. Having designed a set of temporal forms, or actions, for various classes of design agents, a designer obtains a set of vocabularies for dynamic design. These vocabularies constitute a temporal visual language. Note that the sense of the term "language" as in "temporal visual language" is different from the sense I have been using when I say

a “descriptive language” of temporal expressions. The former is the ordinary use of visual language in graphic design sense, and the latter is a meta-language, or a structure, by which visual language can be represented.

After creating vocabularies of temporal expressions, a designer can also consider a compositional grammar that can help orchestrate design agents over time in a systematic fashion. In the Model of Dynamic Design, grammar is an abstraction over strategies and actions. It is represented as a collection of strategies that compose an expression by appropriately arranging actions. Moreover, the abstraction of grammar can be further enriched by concepts that have been developed in music. For example, to design strategies and actions, a tempo, or temporal unit, of the composition may be used. In addition, we may also borrow the concept of measure, similar to that of the spatial grid in traditional graphic design; hence we may call it the temporal grid. Together with the spatial grid, a designer may design the temporal grid to support the spatiotemporal structure of the dynamic design. We can then also introduce the concepts of rhythm and accent, again borrowed from music and dance. I envision that these concepts can be adopted and internalized, and will comprise a critical language which extends designers’ ability to discuss and evaluate temporal design solutions. Such an extension is made possible only with a descriptive language like the one proposed.

### **7.3. Summary**

I have summarized how the Model of Dynamic Design served a role in the development of dynamic design solutions that were demonstrated in Chapter 6. I have also presented various issues concerning the use of the model, as well as its shortcomings. In general, I have shown that the Model of Dynamic Design is an appropriate approach to the design of digital communication, by highlighting various characteristics of design solutions that are unique to dynamic design problems.

Throughout Part 1 and 2, I have presented the Model of Dynamic Design from a perspective of designing. In the next concluding part, I situate the Model of Dynamic Design within a broader picture of digital communication. Roles of designers and design systems are identified. Criteria for the development of design systems are also be discussed.



## Part 3

The following chapters conclude this dissertation by presenting how the model proposed in this research will benefit the design of digital communication. Chapter 8 identifies a role of computational design systems as a means of improving the communication in digital communication environments, and postulate the model of dynamic design as one of the suitable approach. Chapter 9 summarizes the contribution of this research and presents the further research directions that are enabled by this work.





## 8. Expressive Design Systems

Although the theoretical framework developed in this dissertation is independent of computational implementation, I have envisioned dynamic design solutions to be implemented in a computational form. This chapter develops a framework for the development of computational design systems, which are used in digital media to solve design problems at run-time. I first present components that constitute computer-based communication, followed by a discussion on desired qualities of design systems. Finally, I look at existing design systems from the perspective of the framework presented in this chapter.

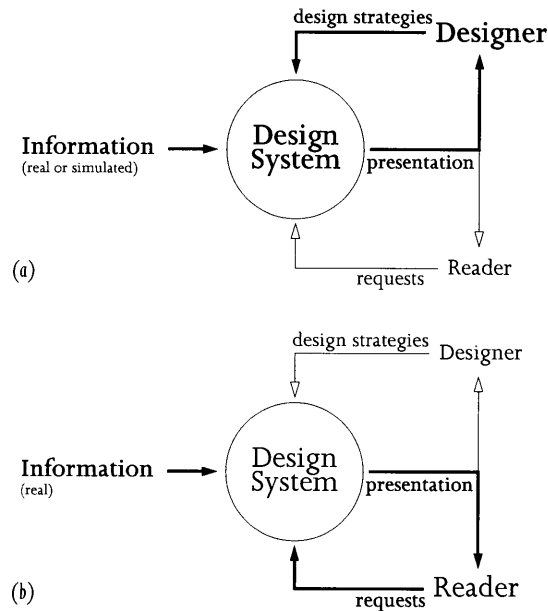
### 8.1. Design Systems

#### 8.1.1. Components of computer-based communication

A *design system* is a software module that is responsible for generating design solutions. It usually resides in some application program, such as electronic news or database systems. A design system initially is like an empty container, and its contents are filled by a designer. In addition, design systems are often referred to as automatic design systems. However, I prefer not to use the term automatic because of its cultural connotation suggesting the mass production of average quality products. I envision individual design systems carefully created by designers.

Figure 2.1 is a schematic diagram that shows the relationships among the components in computer-based communication. Diagram 2.1.a shows the designing stage where a designer specifies the behaviors of a design system. At this stage, real or simulated information is used to evaluate the performance of the design system. Diagram 2.1.b shows the communication stage where the reader reads information presented by a design system. A reader may also make requests in various forms through the application's interface. The design system receives information from an application, and it generates a design solution according to an immediate context.

The design system can be considered a closed world with four communication channels, each of which has a specific language. The first is the channel where information contents are fed. Information must often be simulated in the design stage. In



**Figure 8.1.** Schematic diagrams of components in compute-based communication with a design system (a) design stage, (b) communication stage. (The layer of application program is not indicated for simplicity.)

the on-line news display example given earlier, this is the channel where news articles are given to the design system. The second is the channel that is used to communicate with a designer. A designer uses some language to specify the behaviors of a design system. In the on-line news display, this is the channel where the designer encodes behaviors for design elements, such as headlines, articles, and photographs. The third channel receives requests from a reader through some interface (e.g., natural language, menu, direct manipulation, and etc.). The reader's request is often interpreted by an application system before it is received by the design system. In the on-line news display, a user may enter a particular request (e.g., to read news by subject, or location) through this channel using a menu or natural language interface. Finally, the last channel, which is an output channel, presents design solutions generated by the design system. The presentation channel is used in both the design stage and communication stage. In the on-line news display, this is the channel where the design system sends out final specification to the software module that physically displays design solutions on a display.

In the design stage, the presentation channel and request channel are used to simulate the performance of the design system so that the design solution can be evaluated. In the communication stage, the presentation channel is used as a response to the reader from the application.

### 8.1.2. Designing Design Systems

In the previous section, I outlined the design system from the larger perspective of computer-based communication. This section will focus on issues in the development of design systems from the perspective of designing.

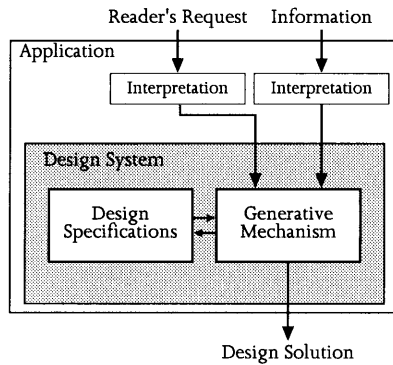


Figure 8.2. Schematic diagram of the architecture of design systems.

Figure 8.2 presents a schematic diagram of the architecture of a typical design system. A design system is composed of a description of design specifications (i.e., how a designer wishes the system to behave) and a generative mechanism that uses the strategies to solve problems. A special descriptive language is used to represent design strategies in a design system. The representation language determines the capability of the design systems. Without an appropriate language, a designer cannot specify desired behaviors to a design system. The generative mechanism is also important; even though the design strategies can be described, a design system may not be able to generate appropriate solutions without a proper generative mechanism. In the on-line news display scenario, for example, information is a set of news articles entered through a network service. A user of the news display, or a reader, may use various interface tools to tell the system what is requested. News articles and user's intentions are interpreted by the application system and sent to the design system. Given immediate news articles and user intentions, the design system's generative mechanism solves a design problem based on design specifications provided by a designer. Then, final solutions are sent to a display.

Figure 8.3 shows a design system building tool in relation to the design system shown in Figure 8.2. A design system building tool is used by a designer in order to encode a particular design solution into a design system. As described in the previous section, the design system is programmed through a communication channel. A design system building tool allows a designer to encode design specifications in a design

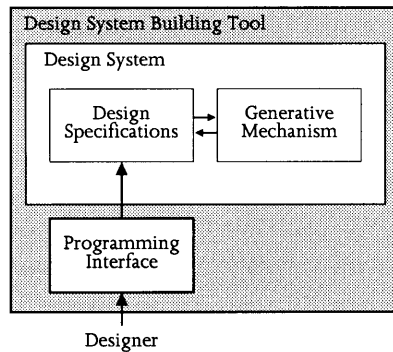


Figure 8.3. Schematic diagram of the relationship between design system and design system building tool.

system. Without an appropriate design system building tool, even if the design system is properly developed, the designer's design specifications may be difficult to encode.

In order to make computer-based communication rich and comprehensive, we must develop design systems that are capable of representing and generating a wide range of design solutions. We must also provide designers with a means of encoding their design concepts; hence design system building tools must be carefully designed. The following sections will discuss issues that I consider important in the development of design systems and design system building tools.

## 8.2. Issues in the development of design systems

### 8.2.1. Expressivity

In order to create dynamic design solutions that are rich and communicative, we want to make a design system highly expressive. In other words, a design system must be designed so that it maximizes the range of design solutions it can generate. Of course, like any other media, computer-based communication media has its own limitations (*e.g.*, resolution and display size). However, we must distinguish between the limitation caused by the nature of presentation media and the limitation caused by tools and the language/models used to generate design solutions.

For example, suppose a designer wants a typographic element to behave in a certain manner (*e.g.*, move from left from right with a slight vibration) for a particular situation. In order for a designer to encode this idea into a design system, design systems must be able to (a) represent that idea, and (b) generate the intended behavior using the specification at an appropriate situation. In order to represent that idea, we must develop a language that allows designers to describe it. It is hard to imagine a single language capable of representing all possible design ideas. Selection of an appropriate representation language for a particular domain is necessary, and sometimes the use of multiple representations may be helpful. A partial list of design ideas that must be represented includes: spatial and temporal relations among design elements, motion and other temporal changes of design elements, types of design elements, types of information, and the reader's intention.

Once design specifications are represented in a design system, a generative mechanism must be able to realize the intended behavior. The ability of a design system to describe design strategies is not equivalent to the ability to use them. Design systems must have an appropriate mechanism that can generate solutions according to the designer's specifications and the immediate context. The generative mechanism involves two major parts: to find the solution when it is desired, and to physically realize the solution.

Since the model of design is a conceptual framework, it does not restrict the expressivity. However, the model and languages proposed in this framework encourage the use of expressive forms presented over time, which has not been highlighted in the

traditional model of thinking. *perForm* supports a relatively wide range of expressivity. Since its agent description language is LISP based, any expressions that the hardware supports can be used.

I have discussed the desired capability of design systems in terms of expressivity. However, the expressivity of a system alone does not guarantee the expressive possibility of a designer. We must also consider programmability, or how designers can encode design specifications into design systems.

### 8.2.2. Programmability

Programmability of the design system is determined by two major factors. First, as described in Section 8.1, regarding the communication channel between the designer and the design system, designers must be able to communicate (or encode) their design strategies to a design system so that those strategies can be represented in the system. One method is to provide designers with the representation language used in the design system so that the communication becomes direct. However, for designers, the direct use of a representation language may be too complex, or too remote from design ideas, depending on the type of language. For example, if if-then rules are used as a representation language, all the design strategies must be described by text, which often prohibits designers from representing some of the visual ideas. Another method is to provide a special language that is more familiar to designers. This helps designers but requires a translation, which may not correctly interpret a designer's intention. The use of visual language, such as sketching, as a means of communicating design ideas to a computer is an example of this method. When such a visual language is used, it must be translated into a textual form for computation. But this translation is not guaranteed to be accurate. Designers of design system building tools must carefully consider this trade off.

The second factor that determines programmability of the design system depends on the nature of the representation scheme itself. That is, a model used in a representation scheme influences the way designers think about design strategies. Various representation schemes may be capable of representing design concepts equally well (expressivity), but, this does not necessarily suggest that their models are equally suitable for designers to "think with," or suitable for a problem at hand.

This research particularly focuses on this second issue, in other words, the nature of representation. The proposed Model of Dynamic Design provides a framework that is suitable for the design of computer-based communication, where both the information and the reader's intention are dynamic. In terms of communication, I decided to provide the representation language as a means for designers to communicate design specifications to the design system. In *perForm*, the designer and the design system communicate directly through the representation language, *persona*. Intentionally, no easy-to-use graphical interface was developed, since the purpose of the research is to first understand the appropriateness of the proposed model through the use of *persona*.

### 8.2.3. Predictability

The performance of design systems must be predictable. There must be a clear correspondence between what a designer programs into a design system and the design solutions that the design system generates. In other words, when the system does not produce what a designer expects, the designer must not have described the desired results. The design system should not generate solutions that are not specified by the designer. I call this condition “constancy requirement,” which the proposed Model of Dynamic Design assumes, and which must be assured in any design system. Predictability in a design system relies on the relationship between the representation language and the generative mechanism. A representation language must therefore reflect the way a design system generates solutions.

Like any other field of design, the design process in dynamic design is likely to be incremental, requiring an iteration of designing (programming) and evaluating (testing the system). Predictability is particularly important in this iterative process of refining design specifications. It would also be helpful for the design system building tools to be able to explain how the design system that generates design solutions.

Throughout the case studies, I found that the model of dynamic design, along with *perForm* and *persona*, provided a relatively high predictability. Since the description of each agent is simply structured, its behaviors are easy to predict from its specifications (although a reader may not necessarily be able to predict the group behavior). However, as discussed in Chapter 7, when chains of communication actions are performed, the behaviors of particular agents may become difficult to trace. It would be useful to provide a tool that allows designers to observe agents’ internal states and communicative actions.

### 8.2.4. Normative independence

It must be recognized that expressivity does not imply good design. The print medium, for instance, has limited expressive capacity; however, the medium does not impose any evaluative design decisions. Similarly, a design system for designers must be designed in such a way that it does not reflect any normative positions (*i.e.*, belief about what the good design should be).

There are two possible ways to impose normative positions in the design system. First, a particular school of thought can be pre-coded into the design system. Most of the existing design systems can be considered to fit into this category: these systems use so-called general design principles, which are often based on psychological findings and traditional design conventions. Researchers of these design systems have previously failed to see individual designers as an integral part of developing design systems, and aim to make a general design system. This is not to state that these systems are inappropriately designed; rather, my argument here is more methodological than substantive.

Second, a design system can be biased by what the system designer believes to be a good design. This type of influence is difficult to avoid completely, since both system designers and designers may not be conscious of their beliefs about good design (i.e., they may blindly accept conventional design principles).

The more expressive design systems are, the more design ideas can be explored by designers. A design system is like a white canvas on which designers can draw their design specifications. We must prepare the canvas so that a wide range of design ideas can be expressed. As we live in a society in which our communication depends largely on visual media, we must not limit digital communication to a range of design principles from the past. We must recognize that the development of an architecture for design systems is different from the creation of good design solutions. As discussed in Chapter 2, the proposed Model of Dynamic Design does not postulate what good design ought to be.

### 8.3. Review of Previous Design Systems

This section presents research efforts in the development of generative design systems and discusses how they relate to the proposed research. The goal of most researchers in generative design systems has been to develop a software architecture that is capable of generating design solutions “automatically,” and has not emphasized issues of designing or the role of the designer. However, these researchers share some problems in the design of dynamic communication media, which is addressed in this dissertation. I will look at their research from a perspective of my research, that is, the conceptual models used behind their techniques.

**Formal Languages:** Studies of formal languages consider graphical presentations as sentences of graphical language, which define precise syntax and semantics based on expressiveness criteria. On one hand, the formal language provides a framework for describing design ideas. Since syntax and semantics of the language are precisely defined, a description can be highly comprehensive. On the other hand, the language can restrict the expressiveness of the final design solution to the given set. Formal languages are often used to represent effectiveness of the communication—“what good design” should be. APT is one of the earliest attempts in the development of automatic design systems [Mackinlay 1986]: it provides a formal language to describe graphic design concepts in the domain of chart and diagram creation. Some other systems have been built based on Mackinlay’s work. For example, SAGE [Roth et al. 1991] extends APT by adding more presentation styles and data characteristics; BOZ [Casner 1991] uses APT-like effectiveness criteria but adds task-based criteria to suit solutions to a reader’s intention.

The Model of Dynamic Design provides a meta-language, or a structure, by which dynamic visual languages can potentially be described. However, it does not propose a specific syntax or semantics of a particular visual language.

**Rule-based model:** If-then rules have been commonly used in the development of design systems (e.g., [Feiner 1991][Mackinlay 1986]). Generally in rule-based systems, individual rules in the form of “if x then y” are easy to write. However, when the number of rules grows, a designer (who is also a novice knowledge engineer) must be experienced in making rules. Since each rule is independently defined, complex inter-dependency among rules may result in an incomprehensive description. Nonetheless, rule-based description can become more accessible when an appropriate abstraction scheme is provided. VIA is a design system that automatically generates page layouts [Weitzman 1994, 1995]. In principle, VIA can be considered a rule-based system, but it uses a domain-specific abstraction called “relational grammar,” which provides designers with a conceptual model to represent spatial and temporal relationships among design elements. Weitzman model makes it easier to describe design specifications and improves the predictability of the system. However, such an abstraction may sacrifice the expressivity of the system unless a rich set of expressive graphical language is supplied (or pre-defined).

Some abstract design concepts are difficult to represent using rules. For example, Feiner recognizes that fundamental visual concepts like balance or rhythm are difficult to represent using rules [Feiner 1988]. Those abstract design concepts are usually difficult for a designer to articulate. Nonetheless, I argue a rule-based approach could convey such abstract concepts by embedding them into rather specific rules, such as alignment and the color of an individual element.

Behaviors of agents in the Model of Dynamic Design are described by using situation-action rules. Although it uses if-then rules, it does not use abstract pattern-directed chaining and all the rules are concrete. In previous chapters, I have postulated that concrete rules are more natural to identify than abstract ones.

**Case-based model:** Design concepts can be represented in the form of cases, or specific design examples. A new design problem is matched against the case library, and a similar case is used to solve the new problem (e.g., [Colby 1992][MacNeil 1989, 1990]). On one hand, some abstract and complex design concepts (e.g., visual balance) are easy to grasp with concrete examples [Lieberman 1993][Ishizaki 1989] and are often used as a means of communication between designers. However, an appropriate representation of design cases are still an open question. On the other hand, there are certain design concepts that can be rather easily described (e.g., alignment). When a design concept can be articulated, the case-based model may become frustrating for a designer.

Behavior rules used in the Model of Dynamic Design are similar to cases in case-based model since these rules are described by concrete situation-actions pairs. However, in the proposed model, cases are partially defined and distributed among the behavior descriptions (strategies) of individual agents, rather than defined for an entire solution.

**Plan-based model:** A model of plan has been used to describe visual presentation. (e.g., Maybury 1993) A plan is viewed as a strategy to communicate a certain kind of message content. For example, if a designer intends to encode a strategy to attract a



reader's attention to a particular text, a plan may include a series of actions such as highlighting, changing size, and moving. A typical plan consists of preconditions, effects, and decomposition (a method for executing a plan). A plan can also include other plans in its decomposition. Similar to VIA, with its high level abstraction, the plan-based approach provides a natural means of expressing design ideas. However, since plan-based descriptions are highly abstract, expressivity relies on the range of primitive communicative acts, or expressions, provided by the design system. In addition, the plan-based approach assumes that the visual design is a collection of purposeful actions, or presentation plans; hence it may not be suitable to represent pure form relationships.

In the Model of Dynamic Design, there is no explicit representation of plan to create a sequence of presentation. Rather, a plan-like behavior is generated by a collection of agents coordinating their behaviors according to their shared situation.

**Declarative vs. action-based description:** The description of a design solution can be either declarative or action-based. Most of the previous systems employ the declarative description. In the declarative approach, design solutions are described in terms of a declarative statement of relationships (e.g., A and B are left-aligned). In the action-based approach, design solutions are described in terms of design acts (e.g., Left-align B to A). The declarative approach assumes that the design concept can be conveyed in a solution, whereas the action-based approach assumes that the process itself conveys design concepts directly. Although both approaches can accomplish the same result, the conceptual model is significantly different. Most previous research used the declarative approach to represent design, except TYRO [MacNeil 1990], which combines both approaches. Since the solution itself is a dynamic process, the proposed theory uses the action-based description. The proposed model is developed based on the action based description.

Previously, researchers in generative design systems have emphasized the representation and generation of design concepts, but have not considered the role of designers and have failed to evaluate their systems from the broader perspective of visual communication. However, the Model of Dynamic Design benefits from the conceptual models and languages used in the development of previous systems. I conjecture that a model of design must incorporate both an appropriate design-specific abstraction and a descriptive language for formal details.

An appropriate abstraction scheme makes a model of design more natural and accessible. However, higher level abstraction tends to sacrifice expressivity because of its indirect nature. Consequently, in order to develop a design system that can generate expressive design solutions, levels of abstractions that range from an appropriate descriptive language for specifying formal details to a higher level of abstraction are necessary.

#### 8.4. Summary

I have presented ways in which the Model of Dynamic Design can be situated in the development of expressive design systems. I have identified the role of and desired qualities for design systems, within the broader picture of computer-based communication. I have also clarified the relationship among designers, readers, and design systems. Using this framework, I have also examined existing design systems in order to compare their models and languages to those of the model proposed in this dissertation.

This Model of Dynamic Design is intended to provide a foundation for building expressive design systems. *perForm*, along with *persona*, has been developed as a prototype of a design system, and has demonstrated the potential of the model of dynamic design through a series of concrete design examples, which are presented in Part 2.

## 9. Conclusion

Any new theoretical framework in design needs to be tested against concrete design problems by practitioners over a long period of time. It has to be experimented with and internalized by designers. Alexander once wrote that "No one will become a better designer ... by following any method blindly." [Alexander 1964] Otherwise, it simply becomes a theory for the sake of an intellectual game—which I think is absurd.

This dissertation has examined the Model of Dynamic Design with the five concrete design problem domains. These experimental design solutions by no means provide complete proof for the validity of the model. Indeed, I rather envision that the model will be tested by many designers over a long period of time. On the other hand, I also believe that these dynamic design solutions are not only the illustration of the theory, but they also demonstrate the potential and possibility of the model in practice.

No one new theoretical framework must, or can, supersede the others. A new framework in design is not meant to shift a paradigm; rather, it contributes to an expansion of our repertoire of designing. The Model of Dynamic Design, of course, is not an exception. The model is fundamentally rooted in the tradition of graphic design, and tries to extend its repertoire in the realm of digital communication—which is dynamic and continuous. Resnick argues that it is always an advantage to possess multiple ways to understand anything [Resnick 1992]. I envision that the Model of Dynamic Design also provides designers with a richer understanding of design problems as well as their solutions.

This research has specially focused on ways in which design problems and solutions in digital communication are perceived. I have argued that the graphic design field currently lacks models and languages to address design problems that are continuous, or design solutions that responds to such continuous problems. The Model of Dynamic Design—the multiagent model along with the abstraction of temporal form—is the result of an attempt to developing such a model.

The model borrows a conceptual and descriptive framework from improvisational performances such as dance and music. The conceptual model of improvisation has particularly provided a unique approach for describing responsive design solutions. A design solution is described as a emergent expression of active and responsive agents—*performers*—each of which is responsible for presenting a segment of information. Similar to a performer in an improvisational performance, each agent has a set of expressive actions and strategies that are carefully designed—or instructed—by a designer, and it selects an appropriate action in response to its changing situation, or context.

Through a series of dynamic design solutions, I have shown that the Model of Dynamic Design can provide a conceptual framework with which a designer can think. One of the advantages of the proposed model is that it emphasizes and highlights important concepts in dynamic design. It helps designers to perceive design problems and solutions in digital communication as stream-like fluid entities; to analyze the structure of dynamic contexts in a systematic manner using the abstraction of situations; to utilize temporal forms as meaningful constituents for a temporal composition; and to understand the nature of information content through the process of decomposition. These concepts have not been emphasized by the traditional declarative models currently exist in the field of graphic design.

### 9.1. Summary of Contributions

This research contributes to the field of graphic design as well as to the development of digital media. The following is a summary of specific contributions.

- *The multiagent model provides a theoretical framework for describing and analyzing design solutions that can continuously respond to its changing context. Using the model of improvisation, the model provides a unique method for describing responsive design solutions.*
- *The abstraction of temporal forms provides a means of describing and analyzing formal changes of the design element. It also encourages the use of temporal form as a meaningful unit of communication.*
- *The model provides a foundation for the software architecture of generative design systems in the development of digital communication media, as well as a foundation for developing a computer language for designers to describe dynamic design solutions.*
- *Exemplary design solutions visually demonstrate the feasibility of creating dynamic design solutions as emergent behaviors of active and improvisational design agents.*

### 9.2. Future Directions

In the course of developing the model, as well as in implementing design solutions, this research raised various issues to be further investigated. This section presents research directions of which the Model of Dynamic Design can be the basis.

### **9.2.1. Design System Developing Tool**

It is often assumed that the designer is not expected to *program*, or else it is argued that an easy-to-use interface (i.e., graphical) must be provided. I argue that the most important aspect of the model and language of design is their expressivity—the range of expressions that can be described by the language. Easy-to-use interfaces often limit expressivity by preventing a designer from dealing with subtle formal details, and are often biased by conventional design methods. This research has focused on the development of a theoretical model that enables the designer to describe dynamic design solutions. Nonetheless, after developing the theoretical framework, we can begin to think about computational tools which support the process of dynamic design. Through the course of creating experimental design solutions, I have identified some of the important features such a tool must accommodate, including a tool for visualizing agent’s activities, and a graphical interface for defining and editing actions and strategies.

### **9.2.2. Design Principles**

I have emphasized that the Model of Dynamic Design is not a normative theory. That is, it does not impose any judgment on what a “good design” should be. However, after developing a model for describing dynamic solutions, we now can start to discuss and analyze principles of dynamic design. Similar to the study done by Klee and Kandinsky, I envision that we can begin to develop a deeper understanding of dynamic visual languages, based upon the Model of Dynamic Design.

### **9.2.3 More Examples**

I envision that the Model of Dynamic Design will be further tested with many other design problems. I would also like to see other designers using the Model of Dynamic Design. The more we experience the use of the model with concrete design problems, the better we could be at using it in solving dynamic design problems. I believe this is one of the most exciting future directions that became possible by this research.

### **9.2.4 Physically-Distributed Design Solutions**

The distributed nature of the multiagent model of design provides an opportunity to explore the possibility of developing physically-distributed design solutions. In the traditional communication, information content is designed for an individual device or medium, such a newspaper, a magazine, or a television. Having the multiagent model of design, we can naturally start to think about a dynamic design solution where agents are distributed across multiple media. For example, a set of design agents representing morning news that you couldn’t finish reading on your computer can follow you around by migrating to your watch and your car, as you leave home, while agents responsible for news articles that are related to children may stay on your computer. Such a distributed and dynamic design solution is a completely different approach to the design of information compared to the traditional design.

### **9.2.5 Education**

Finally, the Model of Dynamic Design can also be introduced to design students. Current methodology in the education of communication design is largely based on the design of traditional media, even though there has been an increasing interest in introducing the design of digital communication media. I hope that the Model of Dynamic Design can help future designers extend their repertoire of design thinking in this new field of dynamic and continuous design.

## Epilogue

*It was almost a decade ago that I conceived a little seed of the idea I present in this dissertation. Technology has dramatically changed since then, but the problems and opportunities that motivated me have remained relatively untouched to this day.*

*At that time, as a young design student, I witnessed the changes in ways we exchange information, caused by the advent of computers. Without knowing much about the technology, I was concerned about the phenomena in which designers had no involvement in the development of digital communication.*

*Of course, my first reaction was to resist the emergent technology and stay with the tradition—i.e., print medium. However, I was increasingly frustrated by the fact that even though I could avoid participating in generating low quality products for digital communication, increasing numbers of such artifacts were starting to surround my visual environment.*

*Lessons in the history of design were helpful—the failure of the craftsman’s resistance to the factory-made objects after the industrial revolution, and the effort of the modern design movement which accepted the industrial manufacturing process and incorporated design into the factory-made objects. As a fearless student, I dreamt of solving the problem of design for digital communication, without really knowing what the real problems were.*

*Between 1987 and 1989, as a master’s student at the Media Laboratory, I was exposed to various ideas, technology, and people, all of which shaped my philosophy towards the design of digital communication. My first project was to build a system that designs—a baby version of what I presented in this thesis. The need for meta-design and the changes in the role of the designer in digital communication, which are addressed in this dissertation, started to form a core motivation of my investigation. In those days, I also started to realize the problem of designing information that changes over time, which became the important driving force for this research.*

*It has been a little more than three years since I started to work on this doctoral dissertation. But the ideas presented here are the result of my struggles over a decade. Throughout my investigation, I have tried to maintain my mindset as a craftsman, while working on theoretical and technological problems. For a traditionally trained designer, it has been a iterative process of resistance and resolution. My emotion had to be justified by reasons, and reasons had to be accepted by my mind.*

*The ideas presented in this dissertation challenge the ways we approach the design problem, or more generally, our communication—i.e., conventions and norms our society has gradually developed over the years. For so many years, we have developed our perception toward the use of visible languages almost exclusively on a flat and static medium of paper. I hope that my research will be able to contribute to the evolution of language use in the realm of digital communication.*



## Appendix / References



## Appendix: Agent Description Language

This appendix describes the syntax of the agent description language, *persona*. The language consists of a small set of LISP macros that enable the description of the design agent's dynamic behaviors.

### defdesigner

---

**defdesigner** *name state-list sensor-list &key :realization-type*

<i>name</i>	A string that is used as the name of a class.
<i>state-list</i>	A list of lists, each of which contains a state-name, an initial value (optional), and a value type (optional).
<i>sensor-list</i>	A list of sensors that the agent uses.
<i>:realization-type</i>	One of <i>text-expression</i> , <i>image-expression</i> , <i>rectangle-expression</i> , or <i>3D-obj-expression</i> .

Description:

*Defdesigner* defines a class of agents with a set of states, and a type of physical realization. Only sensors contained in the sensor-list are used by the agent.

Example:

```
(defdesigner headline-designer
  ((text :init-value "no headline"
         :type string)
   (translation :init-value (make-vec3 0.0 0.0 1.5)
                :type array)
   (rotation :init-value (make-vec3 0.0 0.0 0.0)
              :type array)
   (scaling :init-value 1.0
             :type float)
   (lineoffset :init-value (make-vec3 0.0 1.0 0.0)
                :type array)
   (size :init-value (make-vec2 20.0 3.0)
           :type array)
   (alignment :init-value :flush-left
               :type symbol)
   (color :init-value (make-pccs-color *HL-FLUSH-COLOR* 1.0)
           :type color)
   (fontname :init-value "Slabserif"
              :type string))
```

```

(fontsize      :init-value 3.0
               :type float)
(fontstyle     :init-value "bold"
               :type string)
(visibleP     :init-value t
               :type symbol)
(selectedP    :init-value t
               :type symbol)
(news-obj     :init-value nil
               :type t)
(age          :init-value 0
               :type integer)
(story-designer :init-value nil
               :type t)
(placename-designer :init-value nil
                    :type t)
(placename-status :init-value 0
                  :type integer)
(prev-headline  :init-value nil
                :type t)
(next-headline  :init-value nil
                :type t)
(clock-designer :init-value nil
                :type t)
(destination    :init-value (make-vec3 0.0 0.0 0.0)
                    :type array)
(motion-vector  :init-value (make-vec3 0.0 0.0 0.0)
                    :type array)
(motion-step    :init-value 3
                :type integer)
(node-addr      :init-value nil
                :type integer)
(selectedP      :init-value nil
                :type symbol))
(hl-importance-sensor hl-age-sensor hl-mouse-sensor) ;; sensors
:realization-type .text-expression)

```

## defstrategy

---

```

defstrategy strategy-name form &key specializer
strategy-name  A name for the strategy.
form           form is a procedure that can be defined using the following
                constructs:
                %if test strategy-1 strategy-2
                %while test strategy-2
                %do (action-1 action-2 .... action-n)
                %seq strategy-1 strategy-2 ... strategy-2 (Note: %seq can be omitted.)
:specializer   A class name of an agent, or nil (default) for a generic strategy.

```

Description:

Defstrategy defines a strategy for a class of design agent, or a generic strategy shared by all the agents. The argument form is a lisp-like procedure composed of strategies and actions.

Examples:

```

(defstrategy hl-while-my-news-is-less-than-12h-old
  (%while hl-is-my-news-less-than-12h-old?
    (%if hl-placename-focused?
      hl-placename-focused-strategy
      (%if hl-placename-selected?
        hl-placename-selected-strategy
        hl-placename-not-focused-strategy))))
:specializer headline-designer)

```

```
(defstrategy hl-placename-focused-strategy
  (%if lh-is-my-form-right?
    (%seq hl-update-form-by-age-strategy hl-placename-normal-strategy)
    hl-change-to-focused-form-strategy))
:specializer headline-designer)

(defstrategy hl-change-to-focused-form-strategy
  (%do hl-change-to-focused-form-action)
:specializer headline-designer)
```

## defaction

---

**defaction** *action-name state-list sensor-list form &key :action-type :specializer*

*action-name* A unique name string for the action.  
*state-list* A list of state the action uses and changes.  
*sensor-list* A list of sensors the action uses.  
*form* A method that realizes the action.  
*:duration* A duration of the action.  
*:action-type* A type of action, which must be either *primitive* or *composite*.  
*:specializer* A class name of agent, or nil (default) for a generic action.

Description:

*Defaction* defines an action for a class of design agents specialized by *:specializer*, or for all types of design agents. An action can be either *primitive* where *form* is a procedure that directly change states, or *composite* where *form* consists of a layers of other actions (primitive or composite). As described in Chapter 4, an action, or temporal form, is composed of a set of phrases. An action can be defined by using a primitive action with multiple phrases, or by using a composite action with multiple actions.

Examples:

```
(defaction hl-change-to-focused-form-act (fontsize color) (hl-importance-sensor)
  ((setf fontsize (+ 1.0 (/ ctime 2.0)))
   (if (= ctime 1)
       (if (> hl-importance-sensor 40.0)
           (set-pccs-color color *HL-IMPORTANT-COLOR*)
           (set-pccs-color color *HL-FOCUSED-COLOR*))))
  :duration 3
  :action-type :primitive
  :specializer headline-designer)

(defaction hl-update-transp-by-age (color) (hl-age-sensor)
  ((let ((age (/ hl-age-sensor *ONE-HOUR*)))
       (if (< age 12.0)
           (set-transp color (- 1.0 (/ age 12.0)))
           (set-transp color 0.0))))
  :duration 1
  :action-type :primitive
  :specializer headline-designer)

(defaction HL-introduce-action () ()
  ((HL-introductory-scale-action 0)
   (HL-go-towards-destination-3steps 0)
   (HL-set-initial-motion-vector 0)
   (HL-use-motion-step-3))
  :duration 3
  :action-type :composite
  :specializer headline-designer)
```

## defsensor

---

**defsensor** *sensor-name state-list form &key :initial-value :value-type*  
*sensor-name* A unique name string for the sensor.  
*state-list* A list of states used by the sensor.  
*form* A lisp expression which retrieves and returns a desired value.  
*:initial-value* An initial value.  
*:value-type* A type of the sensor value.  
*:specializer* A class name of an agent.

Description:

Defsensor defines a sensor used by a class of agents. The variable *mas* is a pointer to a multiagent system and used in the form to access external information.

Examples:

```
(defsensor hl-importance-sensor (news-obj)
  (get-importance-level mas news-obj)
  :init-value 0.0
  :value-type float
  :specializer headline-designer)
```

```
(defsensor hl-age-sensor (news-obj)
  (age-of news-obj)
  :init-value 0
  :value-type integer
  :specializer headline-designer)
```

## deftest

---

**deftest** *test-name state-list sensor-list form &key :specializer*  
*test-name* A unique name string for this test.  
*state-list* A list of states used for this test.  
*sensor-list* A list of sensors used for this test.  
*form* A testing method.  
*:specializer* A class name of an agent.

Description;

Deftest define a test that provides a class of agents a capability to recognize a particular situation.

Examples:

```
(deftest hl-should-it-be-alive? () (hl-age-sensor)
  (< hl-age-sensor (time-window-of mas))
  :specializer headline-agent)
```

```
(deftest hl-is-placename-focused? (placename-status) ()
  (if (/= (logand placename-status *OVER*) 0) t nil)
  :specializer headline-agent)
```

```
(deftest hl-is-distracting-transp? (color) ()
  (> (transp color) *HL-BACKGROUND-TRANSP*)
  :specializer headline-agent)
```

## defmessage

---

**defmessage** *message-name state-list &key :i-rule :specializer*  
*message-name* A unique name string for this message.  
*state-list* A list of states that this message may influence.  
*:i-rule* A method that interprets the content of a message.  
*:specializer* A class of agent, or nil (default) for a generic message.

Description:

**Defmessage** defines a class of message that can be interpreted by a particular class of agents, or by all types of agents. *msg-content* used in the definition of *:i-rule* can be any lisp data object.

Examples:

```
(defmessage hl-inform-mouse-status (placename-status)
  :i-rule (setf placename-status msg-content)
  :specializer headline-designer)
```

```
(defmessage hl-de-select (selectedP)
  :i-rule (setf selectedP 0)
  :specializer headline-designer)
```





## References

- Akin, Ö. *Psychology of Architectural Design*. Pion. 1986.
- Agre, P.E. and Chapman, D. *What are plans for?* In *Designing Autonomous Agents*. (ed.) P. Maes. Bradford-MIT Press. 1990.
- Alexander, C. *Notes on the Synthesis of Form* (1964). Harvard University Press. 1971.
- Allen, J. *Maintaining knowledge about temporal intervals*. *Communications of the ACM*, vol.26 no.11. 1983.
- Bailey, J. *First we reshape our computers, then our computers reshape us: The broader intellectual impact of parallelism*. *Daedalus*, vol.121 no.1, 1992.
- Blom, L.A. and Chaplin, T.L. *The Moment of Movement: Dance Improvisation*. University of Pittsburgh Press. 1988.
- Bond, A.H. and Gasser, L. (eds.). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann. 1988.
- Bonsiepe, G. *A Method of quantifying order in typographic design*. *ULM* 21. 1968.
- Bork, A. *A preliminary taxonomy of ways of displaying text on screens*. *Information Design Journal*, vol.3 no.3. 1983.
- Brooks, R.A. *Intelligent Without Reason*. A.I.Memo. MIT. 1991.
- Bucciarelli, L.L. and Schön, D.S. *Generic design process in architecture and engineering: A dialogue concerning at least two design worlds*. internal publication MIT. 1992.
- Buchanan, R. *Wicked problems in design thinking*. *Design Issues*, vol.8 no.2. 1992.
- Casner, S.M. *A Task-analytic approach to the automated design of information graphic presentation*. *ACM Transactions on Graphics*, vol.10 no.2. 1991.
- Colby, G. *Intelligent Layout for Information Display: An Approach Using Constraints and Case-based Reasoning*. M.S.Thesis. Massachusetts Institute of Technology. 1992.
- Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M. and Gero, J.S. *Knowledge-based Design Systems*. Addison-Wesley. 1990.
- Cross, N. *Developments in Design Methodology*. John Wiley & Sons. 1984.
- Demazeau, Y. and Müller, J. *Decentralized A.I. 2*. North-Holland. 1991.
- Demazeau, Y. and Müller, J. *Decentralized A.I.* North-Holland. 1990.

- Dondis, D.A. *A Primer of Visual Literacy*. MIT Press. 1986.
- Eisner, W. *Comics & Sequential Art*. Poorhouse Press. 1985.
- Feiner, S.K. *A Grid-based approach to automating display layout*. Proceedings of the Graphics Interface '88. Morgan Kaufmann. 1988.
- Fox, M.S. *An organizational view of distributed systems*. IEEE Transactions on Systems, Man and Cybernetics, vol. 11. 1981.
- Frost, A. and Yarrow, R. *Improvisation in Drama*. St. Martin's Press. 1989.
- Gerstner, K. *Designing Programmes*. Arthur Niggli. 1968.
- Gross, M., Ervin, S., Anderson, J. and Fleisher, A. *Designing with constraints*. In *Computability of Design*. 1987.
- Gross, Mark. *A basis for computer-assisted design*. In *The Electronic Design Studio*. (eds.) W.J. Mitchell, M. McCullough. and P. Purcell. MIT Press. 1990.
- Hickman, S. and Shiels, M. *Situated action as a basis for cooperation*. In *Decentralized A.I. 2*. (eds.) Y. Demazeau and J. Müller. North-Holland. 1991.
- Hiebert, K. *Graphic Design Processes: Universal to Unique*. VanNostrand Reinhold. 1992.
- Ishizaki, S. and Lokuge, I. *Intelligent interactive dynamic maps*. Proceedings of AutoCarto 12. 1995.
- Ishizaki, S. *Example-Based Graphical Programming: An approach for Graphic Design in Electronic Media*. M.S.V.S. Thesis. Massachusetts Institute of Technology. 1989.
- Jones, C. *Chuck Amuck: The life and times of an animated cartoonist*. Avon Books. 1989.
- Kandinsky, W. *Point, Line, and Plane* (1926). Museum of Non-objective Painting. 1947.
- Kaplan, A. *The Conduct of Inquiry*. Chandler. 1964.
- Kepes, G. *Language of Vision*. Paul Theobald. 1961.
- Klee, P. *Pedagogical Sketchbook* (1926). Faber and Faber Ltd. 1953.
- Knight, T.W. *Color grammars: designing with lines and colors*. Environment and Planning B, vol. 16. 1989.
- Lang, J. *Creating Architectural Theory: The Role of the Behavioral Sciences in Environmental Design*. Van Nostrand Reinhold. 1987.
- Lieberman, H. *Mondrian: A Teachable graphical editor*. In *Watch What I do: Programming by Demonstration*, (ed.) A. Cypher. MIT Press. 1993.
- Lindinger, H. (ed.). *ULM Design: The Morality of Objects*. MIT Press. 1991.
- Lokuge, I. and Ishizaki S. *GeoSpace: An interactive visualization system for exploring complex information spaces*. Proceedings of ACM CHI '95. 1995.
- Mackinlay, J.D. *Automating the design of graphical presentations of relational information*. ACM Transactions on Graphics, vol. 5 no. 2. 1986.
- MacNeil, R. *Adaptive perspectives: Case-based reasoning with TYRO, a graphic designers apprentice*. Proceedings of the IEEE Workshop on Visual Languages. 1990.
- MacNeil, R. *TYRO, A Constraint-based graphic designer's assistant*. Proceedings of the IEEE Workshop on Visual Languages. 1989.
- Malone, T.W. *Modeling coordination in organizations and markets*. Management Science, vol. 33 no. 10. 1987.

- Maybury, M.T. *Intelligent Multimedia Interfaces*. MIT Press. 1993.
- Maybury, M.T. *Planning multimedia explanations using communicative acts*. In *Intelligent Multimedia Interfaces*. (ed.) M.T. Maybury. MIT Press. 1993.
- Mitchell, W.J. *Formal representations: a foundation for computer-aided architectural design*. *Environment and Planning B* vol. 13. 1986.
- Mitchell, W.J. *The Logic of Architecture: Design, Computation, and Cognition*. MIT Press. 1990.
- Munsell H.A. *Munsell Book of Color: Defining, Explaining, and Illustrating the Fundamental Characteristics of Color (A Revision and Extension of The Atlas of the Munsell Color System" by A.H.Munsell)*. Munsell Color Company, Inc. 1929.
- Müller-Brockmann, J. *Grid Systems in Graphic Design, 3rd revised edition*. Verlag Gerd Hatje. 1988.
- Oliver. M. *The deer*, In *House of Light*. 1990.
- Papazian, P. *Incommensurability of criteria and focus in design*. In *CAAD Futures '93*. (eds.) U. Flemming and S. Van Wyk. North-Holland. 1993.
- Pressing, J. *Cognitive processes in improvisation*. In *Cognitive Processes in the Perception of Art*. (eds.) W.R. Crozier and A.J. Chapman. Elsevier Science. 1984.
- Resnick, M. *Beyond the Centralized Mindset: Explorations in Massively-Parallel Microworlds*. Ph.D. Thesis. Massachusetts Institute of Technology. 1992.
- Rittel, H.W.J. and Webber M.M. *Planning problems are wicked problems*. In *Developments in Design Methodology*. (ed.) N. Cross. 1984.
- Roth, S.F. and Mattis, J. *Automating the presentation of information*. *Proceedings of the Fourth International Conference on Expert systems for Production and Operations Management*. Miami Beach. 1991.
- Rowe, P.G. *Design Thinking*. MIT Press. 1987.
- Rowe, R. *Interactive Computer Music*. MIT Press. 1993.
- Schön, D.A. and Wiggins, G. *Kinds of seeing and their functions in designing*. *Design Studies*, vol. 13 no.2. 1992.
- Schön, D.A. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books. 1983.
- Simon, H. *Structure of Ill-structured problems*. *Artificial Intelligence* no.4. 1973.
- Singh, M.P. *Group ability and structure*. In *Decentralized A.I. 2*. (eds.) Y. Demazeau and J. Müller. North-Holland. 1991.
- Singh, M.P. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*. Springer Verlag. 1994.
- Sivasankaran, V. and Owen, C. *Data Exploration: Transposition Operations in Dynamic Diagrams*. IIT Design Processes Laboratory 1990.
- Steels, L. *Towards a Theory of Emergent Functionality*. *Proceedings of First International Conference on Simulation of Adaptive Behavior*. MIT Press. 1990.
- Stiny, G. *Introduction to shape and shape grammars*. *Environment and Planning B*, vol.7. 1980.
- Stiny, G. *The Algebras of design*. *Research in Engineering Design*, vol.2. 1991.

- Steeb, R., Cammarata, S., Hayes-Roth, F.A., Thorndyke, P.W. and Wesson, R.B. *Architectures for distributed intelligence for air fleet control*. Technical Report R-2728-ARPA. Rand Corporation. 1981.
- Sycara, K. *Resolving goal conflicts via negotiation*. IAAA Proceedings. 1987.
- Tufte, E.R. *Envisioning Information*. Graphics Press. 1990.
- Thomas, F. and Johnston, O. *The Illusion of Life: Disney Animation*. Hyperion. 1981.
- Weitzman, L. and Kent W. *Automatic presentation of multimedia documents using relational grammars*. Proceedings of ACM Multimedia '94. 1994.
- Weitzman, L. *The Architecture of Information: Interpretation and presentation of information in dynamic environment*. Ph.D. Thesis, Massachusetts Institute of Technology. 1995.
- Whitefield, A. and Warren, C. *A blackboard framework for modeling designers' behavior*. Design Studies, vol.10 no.3. 1989.
- Wong, Y. *Temporal Typography: Characterization of time-varying typographic forms*. M.S.VS Thesis. Massachusetts Institute of Technology. 1995.
- Xiang, W-N. *A GIS/MMP-based coordination model and its application to distributed environment planning*. Environment and Planning B: Planning and Design, vol.20. 1993.

4271-13