DYNAMIC DESIGN:

Tools and Strategies for Electronic Media

By David Kallas Young

Bachelor of Arts, University of California, Santa Cruz, 1986

Submitted to the Media Arts and Sciences Section in the
School of Architecture and Planning in partial fulfillment
of the requirements for the degree of

Master of Science in Visual Studies

at the Massachusetts Institute of Technology, September 1991.

Author ______________________________
David Kallas Young
Media Arts and Sciences Section
9 August 1991

Certified by ______________________________
Muriel Cooper
Thesis Supervisor
Professor of Visual Studies

Accepted by ______________________________
Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students

DYNAMIC DESIGN:

Tools and Strategies for Electronic Media

By David Kallas Young

Submitted to the Media Arts and Sciences Section in the School of Architecture and Planning in partial fulfillment of the requirements for the degree of Master of Science in Visual Studies, September 1991.

ABSTRACT

Dynamics are the perceived dimensionalities generated through the process of looking at a visual composition. Dynamics can be used to provide complex non-verbal messages that supplement the concrete words and images with which we communicate. As designers begin working with electronic media, the need to explore and understand methods of creating dynamics increases, since it is more difficult for people to envision the complex relationships which might be used to create them. Current computer-based design systems, however, either give the user detailed control over the graphic objects or allow the designer to specify design goals which the system then satisfies.

This thesis develops a methodology for constructing active and interactive dynamic compositions by developing tools which support a more improvisational work style. Motions and behaviors of graphic objects can be specified in relation to user inputs in order to explore the use of motion to convey information. Higher level design specifications, based on visual strategies and object constraints, are used to facilitate the generation and abstraction of complex designs.

By allowing designers to specify high level changes to objects they can more easily create and explore design alternatives. As a result, productivity and creativity will be enhanced.

Thesis Supervisor: Muriel Cooper
Title: Professor of Visual Studies

This work has been supported, in part, by Hewlett Packard, NYNEX Corporation, and Paws Inc.

ACKNOWLEDGMENTS

I would like to thank Muriel Cooper for giving me the opportunity to study at the Visible Language Workshop.

Special thanks to:

Laura Scholl

Michelle

Stone (Idea Boy)

bob

Dave

Grace

Didier

Laura Robin

Ming

Tetsuya

Eva

Neil

Mom and Caren

Dad, Jill, Robbie, and Greg

✌ ♡ ❀

CONTENTS

1. Preface

> *The environment of the man living today has a complexity which cannot be compared with any environment of any previous age. The skyscrapers, the street with its kaleidoscopic vibration of colors, the window-displays with their multiple mirroring images, the street cars and motor cars, produce a dynamic simultaneity of visual impression which cannot be perceived in the terms of inherited visual habits. In this optical turmoil the fixed objects appear utterly insufficient as the measuring tape of the events.* [Kepes 1961, p176]

MOTIVATION

Just as it was in Kepes' industrial world of over thirty years ago, today's technological world is more complex than it has ever been before. The demands it makes upon us can, at times, seem overwhelming. In response to this new environment, we need to change our perceptions in order to better understand the ways in which it functions and how we function within it. These new perceptions should take into account its dynamic, active qualities. And, just as generations have done previously, we search for means of expression with which we can better comprehend and make sense of our perceptions.

The development of digital electronics has paralleled (or perhaps, produced) this increasing complexity. Today, electronic tools provide us with new ranges of expression with which we can attempt to better understand ourselves. For example, digital sound samplers have allowed a new generation of musicians to create music in response to the overwhelming stimuli and turbulence within our urban centers. Music videos, made using electronic editing and effects tools, have also adapted this form of sampling in order to provoke emotional responses sympathetic to their subjects.

Electronic media, particularly computers, provide us with new ranges of visual expression. They provide us with new ways to view and represent our world. No longer must we express our thoughts and ideas with static images. Instead we can create designs which contain movement. We can create visual compositions of objects which have behaviors that are intelligent and responsive. These new compositions enable us to perceive our world not only as one of active forces, but in terms of forces which are constantly changing and interacting with each other.

By creating dynamic visual designs with motion we are able to incorporate the complexity of our external world into that of our internal worlds. As a result we begin to change the way we perceive the variety of stimuli that we receive every day. We become more conscious of the dynamic qualities of our experiences and are become better able to communicate in the world.

THESIS OVERVIEW

This thesis is divided into five main chapters. The first chapter provides motivation for the work and discusses the need for dynamic communication. The second chapter introduces dynamics and describes a scenario system in which dynamics are used to convey non-verbal information. The third chapter discusses some of the ways in which dynamics are generated in static media and ways in which they can be generated in active media. In addition, primitives for communicating with dynamics are described. The fourth chapter investigates the design process, methods of generating designs at a higher-level, and the requirements users place on design tools. Finally, the fifth chapter describes a set of tools implemented to explore the specification of active designs, and how the designs generated can be used to communicate dynamically.

2. Non-Verbal Communication

Communication can occur at several levels. *Literal* (or *verbal* – although it does not necessarily mean spoken) communication is equivalent to the meanings contained in words themselves – disembodied from the more general ideas that the words might represent. Achieving this separation, however, is practically impossible since words' symbolic meanings reverberate – even after thousands of years of being in alphabetic form.

When words are spoken, the aural qualities of the words, generated by the speaker, conveys additional information – based, in part, on the tones and accents of the speaker's voice. This aural information encodes abstract information separate from the ideas and symbols contained by the words. Similarly, when we substitute images for words, information is communicated that is distinct from the symbolic meanings of the figures.

We define *non-verbal* communication, therefore, to be communication at the abstract level – removed from the concrete meanings contained within words and images. Such communication can be used to generate either general or specific responses from the receiver.

Communication that occurs at multiple levels simultaneously (existing both verbally and non-verbally) can be tremendously rewarding and informative. In perceiving such multi-modal communications, one has the rich ambiguity of feeling that there is something additional being communicated but not being quite sure what it is. Such aspects of communication permit one to leave the demarcated spaces of literal communication and enter realms of understanding that are unique to each person's perception and imagination.

This chapter introduces the concept of dynamics and presents a scenario system in which the use of dynamics enhances the communication of the information presented at a non-verbal level. The scenario is devised in order to provide a framework for further discussion of potential computer-based applications of dynamics and interaction and to understand the tools required to construct such applications. A more detailed definition of dynamics and descriptions of specific types of dynamics are presented in the third chapter.

DYNAMIC COMMUNICATION

Dynamics are the perceptions obtained through the process of looking at a visual composition. They are generated, in part, by the viewer's eyes scanning the composition in an attempt to organize its many component elements into a meaningful whole. Dynamics can be perceived in two and three dimensional static media (such as print and architecture) as well as in active media (such as video and kinetic sculpture).

Through dynamics, information can be communicated to the viewer at a non-verbal level. Since dynamics are different for each viewer, the specific messages can vary. However, general abstract, emotional, and descriptive qualities can be conveyed to enhance the literal communicative elements of a design.

The use of dynamics in electronic media, in order to provide additional information, is attractive. By doing so, a designer can present information more clearly and with less ambiguity. Dynamics can be used to enhance the information communicated – whether it is specific (such as news, private communication, or geographic directions) or abstract (such as music, personal expression, or corporate identity). By using the active elements available in electronic media, designs can be created which change over time, and whose dynamics are sympathetic to the changing properties of the source material.

A DYNAMIC COMMUNICATION SYSTEM

One potential use of dynamics is to assist in communicating information through hyper-media systems. As users are faced with increasing amounts of information to browse through, dynamics can provide important cues as to the nature and significance of the information available, and to assist in associating related objects.

This section describes a project, developed as part of this thesis, in which dynamics are used in a computer-based system for news communication. By designing an application which incorporates dynamics we can better understand what issues and techniques are involved in active design. With this knowledge we can determine what tools we need to develop.

The project grew from the question of whether there exist a set of motion primitives that describe information in an active communication system. And, if such a system were developed, what the primitives for communication motion would be. The resulting work was the conception of *Dynamic Communication Systems* in which the motions of the objects displayed conveys information in a way that is just as relevant as the literal information embodied in the objects. Systems could be constructed to convey visual information which enhanced source material from music, or news or other verbal information.

The implemented Dynamic Communication System is a world of dynamic news information objects (Note A). Using the system, a person sits at a console which could potentially be connected to a variety of news [or other data] sources. Each news object, as it was received, would be automatically tagged with attributes such as news category, content, size, priority, date/age, etc. In addition, each object has a motion, or a set of motions, which are related to its content and to the user's interests in the various attributes which are associated with the object. The system is an attempt to understand the dynamics, or non-verbal messages, created when objects are associated with various behaviors, and how those messages might change as the behaviors are modified over time.

Note A. The implemented system ran as two processes. The first was responsible for storing the content information of the objects and of their behaviors, and for responding to user inputs by changing object parameters. The second process was an animation system [*Cleve* - developed by Bob Sabiston] which was responsible for displaying the objects in graphic form. While the creation and specification of objects, and their linkages to input devices, was limited, it was the potential meanings which might be derived from their behaviors that was of interest.

As a user begins to use the system a collection of information objects are moving about the screen. The starting state for the system might be similar to the front page of a newspaper where there is a collection of reasonably related, generally significant news items. The "page" could then be changed and customized as the user works with the system. The objects appear on the screen with basic motions. If objects are related they can be grouped, causing them to stay close to each other – making it easier for the user to notice their similarity.

The objects start with default motions, but as the user informs the system of his or her interests and priorities, the objects' motions (of position or other element attributes) are modified to accommodate the user. As an example, Images A–E demonstrate, in schematic form, how changes in object position, shape, and color can be used to supplement literal information. The starting state (Image A) shows several primary issues: 1) a **Trouble** story on a particular region, 2) another story regarding a similar, perhaps related, **Conflict**, 3) general news about a current **Election** and 4) one candidate in particular (**Smith**), and 5) a **Weather** forecast. Image B shows the progression of these stories as the events proceed with little change. Notice, however, that the story on **Smith** is orbiting the **Election** coverage to indicate its relatedness. In Image C a second political candidate, **Jones**, has become significant and the **Trouble** has turned into a **War**. In addition, the user has placed an object into the world to attract stories of a specific subject category – in this case related to the war. Articles in this category are attracted to the object – as indicated by the change in the **Conflict** story's velocity. Image D shows the **Tension** story orbiting the attractor object and candidate **Smith** becoming less important (as shown by its lighter color). Finally, in Image E, candidate **Smith** is no longer significant in the **Election** and so leaves the display.

The power of this system is derived from its ability to be customized by the user according to his or her interests and needs, and in its potential to learn. As the user customizes their environment it becomes easier for the system to understand a user's general, longer term, interests, and for it to display new news objects of relevance. It could function as an intelligent editor, sympathetic to the user's needs.

To a naive user the graphic behaviors provide an easy way of seeing the varieties of attributes for the different information objects. While the user's understanding of the specific behaviors may be rudimentary, they will be

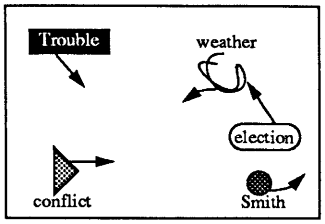


Image A: Starting configuration. All objects have simple movements except Smith which is orbiting Election.

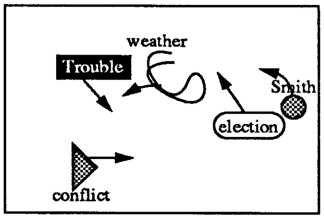


Image B. Objects continue movements. Note changes to weather's shape and Smith's color.

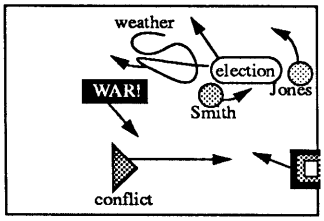


Image C. New candidate and attractor (which draws conflict object) objects appear.

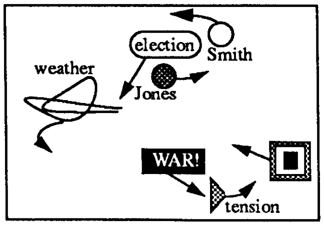


Image D. Tension in orbit around attractor. Election bounces off top and begins to move down.

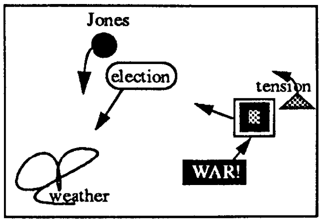


Image E. Smith leaves screen.

able to perceive information which would not otherwise be recognized from the stationary objects. This additional information may be similar to the dynamics perceived when reading a newspaper – generated by the use of spatial, graphic, and typographic cues to provide non-verbal categorization and emphasis of the articles. Hopefully the dynamics and messages provided by this system are more complex and informative. And, as users become familiar with the system, their ability to read and understand the behavioral information will improve.

3. Dynamics and Interaction

This chapter addresses the use of dynamics and interaction as methods of communicating non-verbal information. It examines work that has previously been done with dynamics, both for static and active media. In addition, specific types of dynamics are examined, and methods to generate them in active media are discussed. By better understanding the means by which dynamics are generated and used we begin to see what tools are appropriate to generate motion for Dynamic Communication Systems.

GENERATING DYNAMICS

All media are based on sets of component visual elements. *Elements* are the basic substance of what we see. The collection of elements which define *static media* include shape, position, color, texture, transparency, and focus. We define *active media* to be those media that contain the additional elements of motion and interaction. Active media includes not only computational media, but also film and television. *Motion* is the process whereby an element changes over time. For example, an object might change its position or transparency. *Interaction* is the ability of the viewer, or any outside force, to affect the description of elements within a composition.

In both static and active media, techniques have been developed to trick the viewer into responding to images as though they contained additional dimensionalities. We call the perceptions obtained through these techniques *dynamics.*

Techniques used to generate dynamics can be difficult to use since they can cause distortions. They are problematic because they can draw upon our perceptions of experiences from everyday life, which are unique for each person and each generation. However, such visual techniques provide nonverbal information to the viewer that can be enormously effective in provoking a variety of emotional responses from the viewer. A design is successful when its dynamics succeed at illustrating the relationship between the content of the design and its composition.

STATIC MEDIA

Static images communicate directly only during the time they hold the viewer's visual attention. In order for an image to remain interesting it must contain relationships which intrigue the viewer and hold his attention. Attention is kept while the viewer is creating their own understanding of how the image is assembled through the relationships of its components. Part of this understanding can be communicated through dynamics generated by the movements of the viewer's eyes, creating paths within the composition.

An image whose relationships are too easily understood quickly becomes uninteresting (Image A). So too does the image whose relationships are so complex that they are beyond our ability to understand them (Image B). The image which succeeds is that which captures our attention and interest (Image C). The longer we are drawn to it – the more we obtain from our interaction with it – the more successful it is considered.

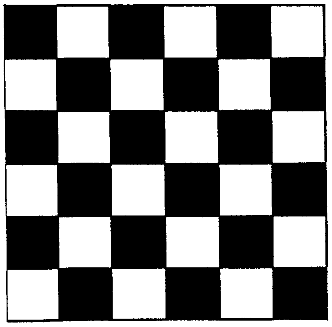


Image A. The ordered and repetitive relationships in this image are too easy to understand and result in the viewer quickly loosing interest in them.

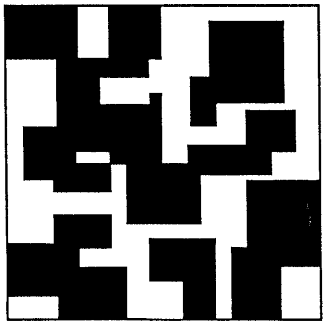


Image B. The elements within this image are too difficult to relate to each other and so the viewer looses interest in attempting to understand them.

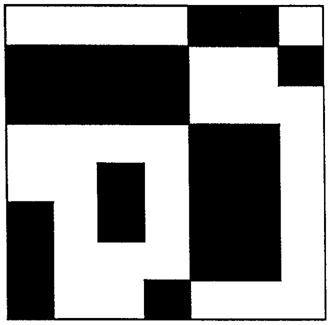


Image C. The relationships in this image are organized and controlled, yet complex enough to be interesting to the viewer (redrawn from [Kepes 1961, p59]).

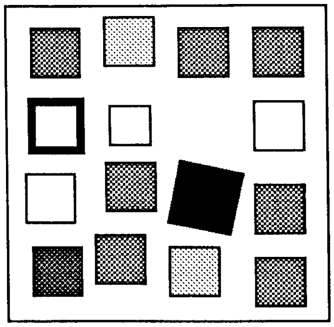


Image A. Active dynamic, without a specific trajectory, generated by the viewer exploring the variations within the composition.

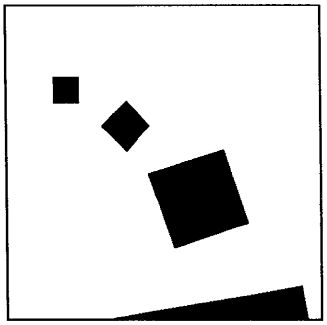


Image B. Activity through perceived dynamic of an object following a path – generated by variations in size and rotation.

In static images the degree and type of added dynamics depends on what approach is taken. For example to make an image more active, without generating a sense of motion, contrasting colors or shapes can be used (Image A). There are also techniques to describe literal motion. For example, by varying the size and rotation of an object we can give the viewer a path to follow – generating the dynamic of an object approaching from a distance (Image B). The power of the image is derived by the designer's understanding of how to create and use these effects.

SEMI-STATIC MEDIA

When working with designs that extend over several pages, such as magazines and books, dynamic potential becomes extremely important. No longer is the image confined to a single page. Instead, movement can be generated through the progression of pages as one moves through the text. This can be accomplished by a design which changes as the content of the source material changes, echoing the rhythms that exist within it.

By keeping the content and design closely related, dynamics can be used to reveal the underlying structure of the concepts being presented. Referring to her design of *Learning from Las Vegas*, Cooper describes this process as "...an exercise in using design to resonate content with subject. The visual materials were not only graphically rich, but as content-laden as the text, so the interdependent rhythms of those relationships were important... Creating virtual time and space in two dimensions has always intrigued me" [Friedman 1989, p98].

The use of "virtual time" as a form of motion can be used to hold the reader's attention. It becomes a type of interaction, too, since the reader must turn the pages, and in doing so perceives the dynamic quality of the design. According to Cooper, this form of design reveals "...the conceptual structure of the book as would a stop-motion movie of the construction of a building over time, or a seed growing into a blossoming flower" [Friedman 1989, p98]. A stop-motion film of the pages of the book *The Bauhaus* [Wingler 1969] has, in fact been made to illustrate this dynamic.

If the source material is less formally structured, a dynamic sense can still be generated, but less literally. Instead the dynamic corresponds to a tempo, or activity level, created by the use and size of illustrations and graphics, and the degree to which text is either fragmented or kept as a whole. This type of dynamic, used in magazines, determines whether it is fast or slow paced.

ACTIVE MEDIA

By adding motion to our palette of design elements we must re-evaluate the general techniques used to create dynamics – to move past the static, two-dimensional methods of representation. We need to view the world as a collection of intelligent processes rather than one of flat motions. By understanding our world in this way we will be better able to understand how to design with motion. By adding motion (and therefore time), the specific ways in which the viewer's attention are held change. Yet the underlying relationships between object elements and the intended viewer perceptions of them should be maintained – even when the viewer is changing and interacting with them.

When working with active media it would be a mistake to simply take the implied motions used in static designs and activate them. Such a design in motion would no longer have the unique dynamics generated by each viewer's examination. Upon seeing the work once, for its duration, a viewer would have experienced the work and be able to get little more from it. The relationships would be understood too quickly and the work would loose its ability to hold the viewer's attention for longer than the length of time of its component motions.

Motion must be used as one of the primitive elements of composition. We can then combine it with other elements in order to create the dynamics of higher order motion. For example, by combining motion with the shape of an object we obtain an active object whose shape changes (Image A). We could similarly combine motion with an object's color or transparency to get other active effects.

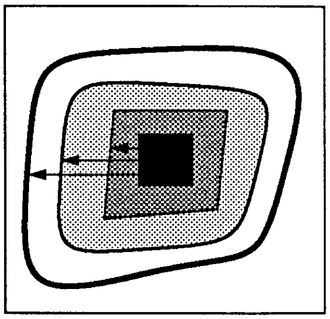


Image A. A repeatable motion sequence illustrating the object's changes in shape and color.

Instead of using motion to illustrate literal movement, we want to use it to create active elements of a composition which can be combined to create meanings greater than the component parts. One method of generating dynamics is by repeating motions, with variations.

Musically this is analogous to some of the music by Steve Reich. Reich has combined repeating sequences, either at different rates or with different lengths, to create works which generate sound dynamics that are not directly notated in his scores (see Image B for a related example)

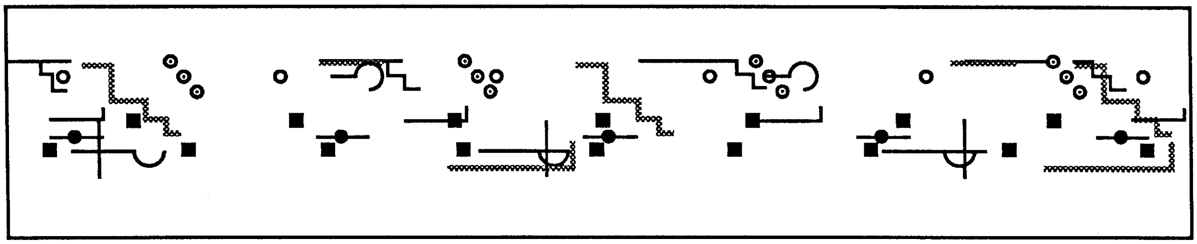


Image B. Section of the score to *1/2* from *Ambient #1: Music for Airports* by Brian Eno [Eno 1978] (redrawn). The piece repeats its component elements at different rates to generate aural dynamics.

Active objects in a composition do not necessarily have to be "physical." Instead they can be transformations which occur to other objects. For examples, a transformation could occur which repetitively made an area (possibly changing in size or position) of an image in and out of focus, or changed its color levels.

In computational media, dynamics can also be generated without motion. Visual components can have understandings of their behavior and, possibly, their intended role in the composition. The elements can be associated together into groups, each of which maintain a balanced sense of motion within themselves, with respect to another group, or in response to viewer interaction. In a Dynamic Communication System, this would be useful for associating behaviors with specific categories or types of information objects.

DYNAMIC COMMUNICATION PRIMITIVES

As in static media, the basic visual elements of active media are combined in order to generate dynamics. The Dynamic Communication System described in the second chapter uses several types of dynamics in order to convey specific types of non-verbal information. The following describes these dynamics and how they were generated through the combination of basic elements. In addition, these dynamics are compared to their analogues in static media in order to better understand their meaning.

INACTIVITY

Inactivity, or the perception of an object at rest, is a dynamic which conveys a sense of stability. In compositions where there are competing dynamics, inactive objects can be used to either represent permanence and importance or to cause an object to appear neutral (Image A). Inactivity can also be used to make an object less noticeable than the other elements of the composition.

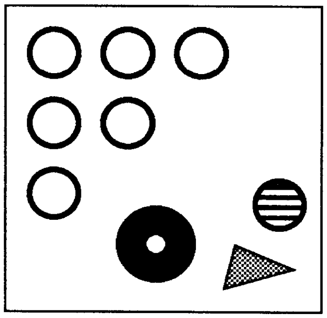


Image A. The objects in the top left area are less active than those in the bottom right due to their regular arrangement and coloring.

In an active medium, inactivity is also generated in ways dependent on context. If the majority of elements in a composition are moving then making an object static would cause it to stand out. To avoid this unwanted effect, an inactive type of motion should be adopted. In the implemented scenario system, inactivity is created by a generic, non-obtrusive motion of bouncing. Bouncing is a simple movement in which an object moves in a particular direction until it collides with another object or the edge of the screen (Image B). It is considered non-obtrusive since, presumably, objects of greater importance have complex, more noticeable, motions. The specific type of bouncing behavior used, whether it is a simple vector model or based on dynamic forces (possibly including gravity), affects its meaning and determines whether or not it a non-intrusive behavior. These parameters can be determined by the nature and context of the information.

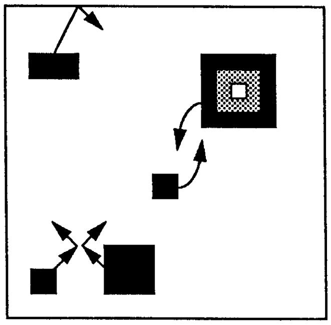


Image B. The top bottom left objects demonstrate simple bouncing motions. The pair of objects on the right side demonstrate more complex position and color activity intended to draw our attention.

GROUPING

When several objects are related to each other we would like to group them. By having the objects behave as a collection, their relationships can be made more apparent to the viewer. In static media grouping can be accomplished through the use of similarly shaped or placed objects (in Image A, the collection of objects in the top left corner).

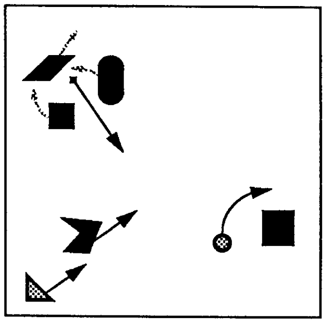


Image A. Three types of active grouping behaviors (clockwise from top left): 1) a group gets a motion, 2) an object rotates around another object, and 3) one object follows another.

In active media grouping can be used to indicate both object similarity and individuality. One method of grouping implemented in the scenario system is to have the objects cluster together, yet move separately (Image A). Additional methods of grouping, whereby behaviors of individual objects are associated with other objects, are also implemented. In this way an object can follow or orbit another object in order to illustrate its secondary nature.

In an interactive medium, grouping can also be done manually by the user. This ability to group could be used to see relationships among objects that might otherwise be difficult to notice. (In the fourth chapter there is discussion of grouping by elements rather than objects.)

SPATIAL FORCES

Objects which in some way reference other objects have a directional energy perceived as a spatial force. In a static composition these force dynamics can be generated by the perception of objects attracting or repelling one another (Image B).

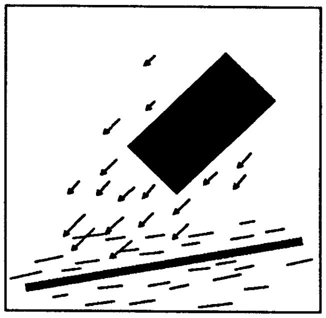


Image B. Spatial Forces, illustrated with arrows, in a static image (redrawn from [Kepes 1961, p 29])

In the Dynamic Communication System we have conceived spatial forces through the use of "magnet" objects. A magnet, associated with a particular subject of category is one possible way to collect relevant information items towards a known location – based, in part, on our intuitive understanding of physical experience. For example, the user could place a subject magnet which would attract objects with a particular subject (Image C). The magnet would search for appropriate stories and influence their motion towards the magnet. Magnets could be considered a method of grouping

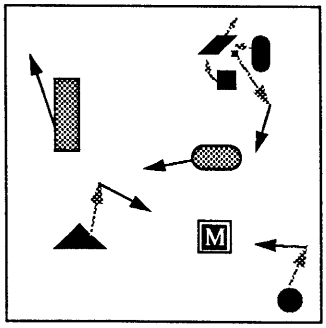


Image C. Spatial forces in active media. The magnet object ("M") is attracting black objects towards it (including the group in the top left corner).

Magnets also have a concept of strength. The stronger a magnet is, the more quickly objects will be attracted to it. Once an object approaches the magnet it should maintain some variation of its original motion, but be constrained in range so that it remains attracted to the magnet.

INTERACTION

Interaction is one of the primary means by which a composition holds the viewer's attention. In static media this interaction is often the result of the way in which the design causes the viewer's eyes to move within it, or, in the case of magazines and books, by its ability to have the reader turn pages and move through the work.

In active media interaction is not only a dynamic but also one of the basic elements. Just as in static media, the interactive dynamic is the result of a composition's ability to hold the viewer's interest. In the implemented scenario system user input is used to specify interests to the system. By increasing an object's interest rating we correspondingly decrease that of the other objects. When the user specifies a low interest in the object it should also reduce the amount and quality of the literal information displayed.

4. Design Specification

The task of creating visual compositions, whether they are posters, advertisements, books, abstract designs, paintings, animations, or hyper-media documents (among other things), is a complex process. In order to generate the desired dynamics the early stages of creating these designs involves a great deal of experimentation. This is particularly true in electronic media with its newly available active elements of motion and interaction. Currently, however, the computer is primarily used only after the design's concept has been finalized. It is delegated to an non-creative, production-oriented role. This chapter addresses the issues involved in bringing the computer into the earlier phases of design experimentation and conceptualization.

This chapter examines the design process. It looks at work styles and methods of describing and creating designs. In particular, it explores one specific way designers tend to work with general concepts – interacting and responding to the visual appearance of a composition at the varying stages of its development. This method of reacting to a design is described in terms of improvisation. We also examine a category of design concepts, strategies, and their powerful structure. Examples of strategies for both static and active media are given. Finally there is a discussion of methodologies for developing tools which support the design process by allowing easier translation of abstract concepts into implemented designs.

THE DESIGN PROCESS

The design process is the process by which a person (or other agent) creates a design. Depending on the intended audience, the design process can vary with respect to its structure, detail, and intimacy. For designs within a highly constrained environment, there is little room for experimentation since much of the composition has already been specified. For more unstructured or experimental design tasks, the designer works by trying a wide range of ideas and approaches.

The design process does not easily lend itself to discussion since it is, in a sense, a description of each person's individual language by which ideas are communicated to themselves and to others. People can attempt to learn or copy design processes, yet they will be interpreted through the individual's personality.

While design processes are specifically unique to each person, there seems to be a common underlying structure. This structure is based on general design approaches which can be combined and modified on an individual basis. The task of describing the design process, therefore, is not that of describing a single, common, process, but a task of describing these more general approaches. By understanding them we can better describe specific design techniques.

IMPROVISATION

The experimental process of design is often one of *improvisation*. The designer has a vague idea of what they are trying to accomplish, but it is through working that they are able to explore their ideas and, as they work, get a more concrete image of what they are working towards. This is similar to the adage, "I'll know it when I see it."

Improvisation focuses on the real-time aspects of the design process. While improvising, a person works, or composes, without stopping to consider the academic aspects of the creation. It as an intuitive process, yet one which is mentally challenging since the mind must concentrate on many factors simultaneously – the mind has little time to focus on the detailed aspects of the work [Minsky 1981].

Through improvisation, ideas can be triggered in response to the feedback provided by the process. Musically this feedback can be provided by other musicians or the sounds generated. For visual compositions, it can be through the intermediate compositions generated or by the physical the tools being used provide – in the case of electronic tools this feedback might also be analytic information regarding the composition.

Improvisation is a type of *flow* experience (Note A) in which the focusing of attention on an activity is central to its success. One condition for this focusing is the designer's interest to create the composition. This interest facilitates, for the designer, the sense of discovery. The discovery can be of various things, and at various levels. These levels seem to correspond to the four major foci of an aesthetic encounter: perceptual, emotional, intellectual, and communicative [Csikszentmihalyi and Robinson 1990].

Note A. Flow can be described as the state in which a person experiences a loss of self-consciousness while performing a task. Doing the task becomes the goal in itself. Among the conditions necessary (but not sufficient) for a flow state to occur are: 1) the participant must be in possession of skills which are adequate to overcome challenges, and 2) have clear goals and clear feedback when performing those goals.

Because improvisation functions at a variety of complex levels there is no specific way of describing it. However we can abstract it to a way of taking a general idea or concept and transforming it into something more complete. The result is a better understanding of the foci of the improvisation.

HIGH LEVEL DESIGN

This section examines strategies both as a method of easily specifying a design, and as a tool with which dynamics can be constructed.

STRATEGIES

Visual communication is based on sets of primitives, called *strategies*. Strategies are, along with elements, one of the two basic categories of visual constructs used to visually express ideas and content [Dondis 1973]. Used to communicate some general meaning, they exist as pairs of opposites, which can be modified to lesser degrees of intensity like step tones between black and white. Examples of strategy pairs include balance/instability, symmetry/asymmetry, regularity/irregularity, and accuracy/distortion.

Strategies are high level descriptions of design. They are used to relate objects or characteristics of their component elements. Strategies are based on human habits of visual perception. Some of these habits are based on our commonly evolved cognitive makeup, while others are culturally learned. The learned habits are what force strategy definitions to be general enough to cover extremely broad ranges of human perception.

By combining strategies, dynamics can be generated and complex messages communicated. These dynamics can evoke perceptions based, not on the strategies used but on a derivation from the visual elements of the composition. Thus, using complex sets of strategies can have unexpected results.

When designing with strategies, the method and content of a communication can radically change. Specifically, a graphic designer might begin by basing the composition on a specific visual relationship, such as balance. This relationship could have been chosen because it was suggested by the emotional response the designer is trying to generate. The designer might then create two graphic regions upon which to base the visual balance. Using the two areas, the designer can add detail, placing within them collections of related objects. With the new, more complex, image, the designer might then see how the balance between the two can be modified. However, the designer might also see the drawing as suggesting an approach other than balance, one which is more appropriate to the current goal. The image might even suggest a new goal. This process of adding detail and modifying the visual relationships among the objects continues until the designer is satisfied with the composition.

The ability to more easily generalize and modify the strategies used in a composition could prove to be a valuable tool in creating and understanding designs. As we begin incorporating active elements, and the resulting complexity of designs increases, the use of strategies may become a necessity.

When designing with active elements, strategies are no less appropriate. By adding motion we can use strategies to express relationships between the dynamic elements. For example, one very basic strategy, *balance*, has meanings in both static and active media. In static media, balance is used as a means of suspending two elements, or components of a design, on either side of a center. In an active medium, balance can be applied to the changes occurring to elements over time. As an object becomes larger, for example, other objects might become brighter in order to remain as significant in the composition (Image A). By applying balance to changing elements a dynamic relationship can be established between them.

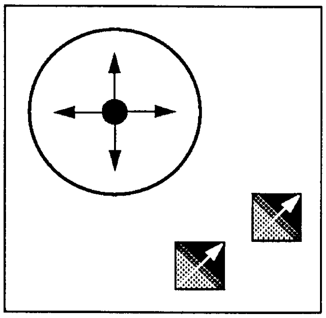


Image A. Active Balance. As the circular object becomes larger, the square objects change their brightness in order to draw equal viewer attention.

TOOL DEVELOPMENT METHODOLOGY

One uses visual design in an attempt to communicate non-verbally. A design begins as a non-verbal concept or idea, and the designer's general understanding of how to communicate it. It is through the act of communication (the process of using tools) that ideas are refined and made more understandable to ourselves and others.

Computers are no more useful a medium for communication than blank pieces of paper. Just as a person needs pencils and other tools with which to access the paper's potential, we, too, need tools with which to utilize the computer's expressive potential. Unfortunately, current computer design tools are often problematic. Designers can get stuck translating their ideas into terms the computer understands. They get so involved with that process that they loose sight of what their original goals were.

There is a wide range of Macintosh programs, such as MacDraw and Pagemaker, which illustrate the way in which a tool can interfere with the experimental design process. These programs give the designer a great deal of direct control over object characteristics such as shape, position, color, and size. However, they store little or no information as to what the objects represent, their relationships to one another, or that the designer's visual strategies were in creating the composition. In order to reconfigure a design a designer must often start from scratch. The tools are primarily useful after the designer has already conceptualized the design and needs only to enter it into the computer for production.

There is a second category of design systems which are those tools which are more "intelligent." These systems tend to use rules to determine characteristics of the visual elements and the relationships between them. For example, Designer [Weitzman 1988] analyzes user designs (primarily of user interfaces) in terms of consistency and effectiveness, and understands designs in terms of making images "consistent, unambiguous and visually effective" [Weitzman 1988, p6]. Lieberman proposes a system which would take design goals (such as "attract attention" or "communicate information"), as specified by the user, and modify existing designs from a design library so that a final design is created that meets the specified goals [Lieberman].

While these tools can be useful in automatically generating designs, they are not particularly appropriate for generating a design concept or for designing in an improvisational mode. These systems function in a similar way as the production tools – the designer typically needs to already know the design goals and strategies to be used.

We place conflicting demands on our tools. We want tools which are specific enough so that we can easily translate our abstract thoughts into concepts that the computer can understand. Yet these tools must be general enough so that they can be used by a wide range of people. We do not want to have to spend too much time learning languages with which to communicate [our ideas] to the computer. Specifically, tools should be understandable, but they should still provide challenges to master. As Cohen states, an artist's tools "have to be difficult enough to use to stimulate a sufficient level of creative performance" [McCorduck 1991, p50].

Because of the computers versatility, there are infinite number of tools which we create. How then do we begin? Kepes writes "Before one begins to use the visual language for the communication of a concrete message, he should learn the greatest possible variety of spatial sensations inherent in the relationships of the forces acting on the picture surface" [Kepes 1961, p23]. We need, therefore, to develop general-purpose tools with which we can better explore the dynamic and interactive visual possibilities made available with computers. Later, once we understand the sensations available, we can go on to develop specialized tools for our particular compositional tasks.

THE SPEAKER'S VOICE

In other domains research in Artificial Intelligence has offered techniques which allow people to more effectively use and communicate with computers. In these domains the structure of the information being manipulated is well understood. In domains in which the knowledge has not been codified extensive knowledge engineering must be done. Unfortunately, the knowledge of how we can use the new features of computers doesn't yet exist. We don't understand the structure of dynamic visual information in active design (let alone static design) in a general sense.

Techniques which acquire knowledge by user examples may prove to be useful. Lieberman discusses potential *learning by example* systems where designers present example design problems to the machine and then demonstrate the steps of solving it [Lieberman 1988]. Throughout the process, however, the designer must present some explanation of why each step was taken and how this explanation can be generalized. While this can be useful in generating design knowledge, it harms the improvisational process by forcing the designer to focus on detailed aspects of the work.

In written and verbal communication we have specific languages which can be used to be a carrier for our ideas. The way in which the ideas are communicated depends on the source's representation of the ideas, and the techniques used to generate the text. Each author has a specific style based on their life experiences and skills as a writer. In computer systems using AI techniques to generate natural language, the structure of the underlying concepts and the method for generating the text result in the output having unique styles. To an experienced reader the speaker's voice can be heard/ discerned through the text.

In visual compositions, this voice exists, too. Someone who is skilled at perceiving art can tell the creator of a visual work. They can also see the influences which resulted in the creator's individual style. This should be no less true for compositions produced in electronic media. A successful example of this is the program Aaron, written by Harold Cohen. While the program generates images based on knowledge of real world objects and their representation and construction, Cohen's own artistic influence is still apparent in the works produced. When using electronic design systems, it is important that individuals be able to modify and adapt their tools – at whatever level they function – so that users are better able to use and find their voice.

5. Tools

With any communication system there is the possibility that the communication channel will degrade the quality of the information being communicated. In any Dynamic Communication System the user's concentration might be reduced if the graphic behaviors attract more attention than the information contained within the objects. This is particularly true if the designers and users of such systems do not carefully choose only those combinations of active elements that are appropriate solutions to the particular context. As the amount of information available to us increases, and systems are developed to assist in organizing and filtering this information for personalized browsing, the need to understand how these elements can be combined becomes paramount.

In constructing a Dynamic Communication System, it is clear that there are tremendous numbers of ways in which non-verbal messages, conveyed using dynamics, can be constructed. It becomes difficult to visualize what messages might be perceived when combinations of dynamic relationships and interactions are used in various contexts. We therefore need ways to explore these relationships. If we are able to experiment with the various combinations, we can explore how the parameters relate to each other in order to create meaningful associations. Those associations that are appropriate can then be expanded into more complete applications and/or designs for dynamic communication.

This section describes four projects, developed as part of this thesis, which explore how the various graphic, dynamic, and interactive communicative elements of electronic media can be combined. Each addresses this issue with a different focus, but similar concerns. Namely, how to effectively use computers (and dynamics) to enhance the communication of information. Each project provides methods for users to experiment with constructing active compositions. The projects are described in chronological order.

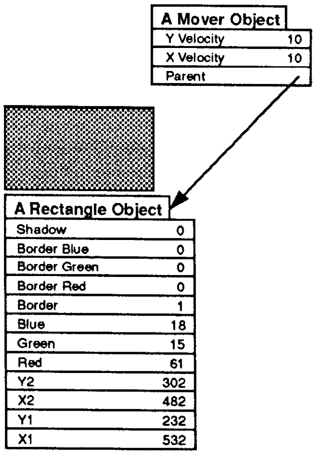
TALK

In the implemented Dynamic Communication System (described in Chapter 2) the associations between objects and their behaviors are described with C code. After implementing scenarios for the system it became clear that specifying behaviors and connections between the objects through non-interactive programming was unacceptable. In order to test a new association, or to elaborate an object, it was necessary to do low level programming. This immediately suggested the need for a more direct method of specifying and changing these relationships. The system that resulted, *Talk* (Note A), is an extension of the scenario system, providing the ability to more easily create graphic objects with various behaviors, and associate them together.

Note A. Named Talk because it attempts to assist in developing communication systems.

Talk provides a set of rudimentary graphic object types, such as rectangles, lines, and anti-aliased text. Additionally, there are non-graphic objects which can be used to construct behaviors. Each object, once placed, has a corresponding editor which appears next to it. The editors, when visible, provide access to the parameters which define the object. If a parameter is a number or a string, a literal value can be entered. If the parameter is a references to other object, a connection can be made to an object.

As an example, we make a rectangle and a mover object. Mover objects are used to move objects that have a position. By linking the rectangle object and the mover object and then activating the system, the rectangle object will move with the direction and speed specified by the motion object (Images A and B)



A Mover Object	
Y Velocity	10
X Velocity	10
Parent	

A Rectangle Object	
Shadow	0
Border Blue	0
Border Green	0
Border Red	0
Border	1
Blue	18
Green	15
Red	61
Y2	302
X2	482
Y1	232
X1	532

Image A. Screen display (redrawn) of a rectangle object and a mover object and their associated editors. Note the graphic link of the mover's parent value to the rectangle object.

By defining more complex structures it was hoped that we could begin to explore how our Dynamic Communication System might function. Yet with Talk, all elements in a composition, including behaviors, are separate objects. This separation of graphic objects and their behaviors is problematic. How can we build more complex behaviors for objects if their component behaviors are fragmented among other objects? To continue using Talk would require complex object relationships which would be difficult to develop and even more difficult to debug.

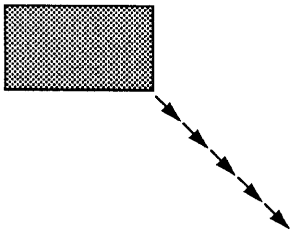


Image B. Arrows indicate rectangle's motion, specified in Image A, for five time steps.

EDEN

Instead of extending Talk we chose to use a different model of objects – one in which the various object behaviors and contents are more closely linked. In the desired model objects control their behaviors rather than being passive recipients of their commands. Fortunately we had a library of routines which closely paralleled this model. These routines, the VLW's window system *BadWindows*, provide access to arbitrarily complex graphic objects and input devices.

The resulting system, *Eden* (Note A), is intended to provide a means of combining graphic elements, complex behaviors, and interaction in order to create dynamic designs. Since these designs are based on BadWindows, Eden also functions as an interface builder and editor and so can be used to examine and modify existing programs developed with BadWindows.

Note A. Named Eden because it exists at a higher level than BadWindows – one which is almost heavenly. ☺

One result of using a window system as an underlying base is the unforeseen expansion of our definition of a Dynamic Communication System to include broad categories of application programs. This is not as unreasonable as it may sound since, in order to create dynamic compositions, one must use some sort of specification language. The language can be at any level of abstraction – including that of a low level programming language such as C.

BADWINDOWS

In order to understand how Eden is used, it is useful to have a basic understanding of the structure of BadWindows. The basic BadWindow object, a *window*, has a position, size, and a list of messages that the window can respond to. Each message has a list of commands consisting of some data and a function which uses that data (see Image A on the following page). The command can also access data within the window object. Typical commands provide specifications of how an object draws itself or responds to input from a physical device.

For example, we can attach commands to a DRAW message which deter-

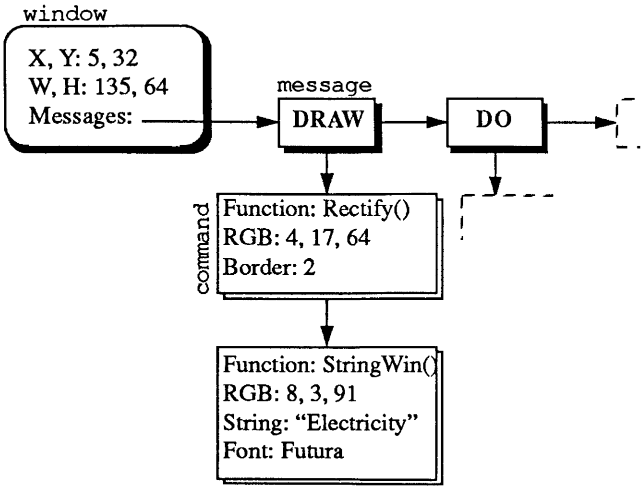


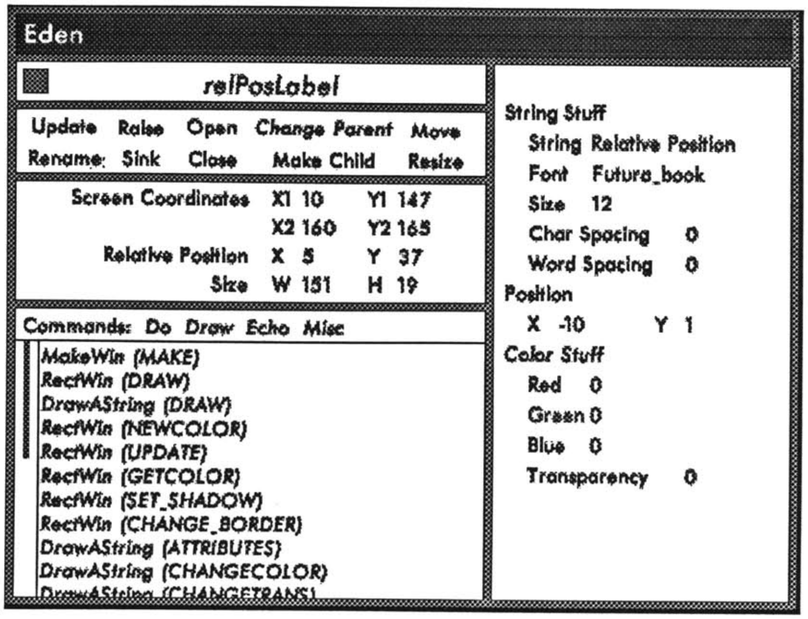
Image A: Underlying BadWindows object structure for a window with DRAW commands. The attached commands draw a rectangular window with a string inside it.

mine how the window displays itself (Image A). The first command attached to the DRAW message might cause the window to display itself as a solid colored rectangle. This command's data stores the color and transparency of the window. A second attached command might then display a string. The command's data stores color, font, size, and text information for the string. When the window receives a DRAW message it finds its list of DRAW commands and executes them in the order in which they are attached.

INTERFACE

Eden runs as a part of a larger program. The larger program can either be an existing one which the designer/programmer intends to modify, or it can be a shell program, in which a new design/program will be created. Eden appears within its host program as a self-contained window (see Image A on the following page).

Eden is divided into two primary components – corresponding to its left and right halves. The left half provides information and operational controls for the currently selected window object – all operations performed using Eden are done to this window. The top bar on the left side shows the name of the currently selected window. Below it is a collection of operations which can be performed on the current window. These include changing its size and parent. This is also where new windows can be created and existing windows destroyed. The center region displays the data contained


Eden
Update Raise Open Change Parent Move
Rename: Sink Close Make Child Resize
Commands: Do Draw Echo Misc

relPosLabel

```
Screen Coordinates  X1 10    Y1 147
                    X2 160   Y2 165
Relative Position   X  5     Y  37
             Size   W  151   H  19
```

```
MakeWin (MAKE)
RectWin (DRAW)
DrawAString (DRAW)
RectWin (NEWCOLOR)
RectWin (UPDATE)
RectWin (GETCOLOR)
RectWin (SET_SHADOW)
RectWin (CHANGE_BORDER)
DrawAString (ATTRIBUTES)
DrawAString (CHANGECOLOR)
DrawAString (CHANGETRANS)
```

```
String Stuff
   String Relative Position
   Font   Futura_book
   Size   12
   Char Spacing   0
   Word Spacing   0
Position
   X -10        Y 1
Color Stuff
   Red    0
   Green  0
   Blue   0
   Transparency   0
```

Image A. Eden interface.

within the window object itself – such as its position and size. The bottom left area provides menus to attach commands to the current window, and displays those commands which are currently attached. An attached command can be moved, removed, or edited. The right half of Eden displays an editor for the currently selected command. When a user chooses to edit a command an editor panel for that command appears providing access to those values used by the command which are user modifiable.

The process of constructing a program or composition with Eden is one of creating window objects and attaching commands to them which provide the desired results. Complex commands can be attached (or constructed) in order to describe window behaviors in relation to user inputs, internal messages, or other windows. By changing commands attached to a window and their sets of associated data, designers can experiment with active and interactive graphic relationships and the dynamics they generate.

One simple but rich experiment constructed with Eden associated user inputs with the color of a window and a string within it. This composition allows a user to associate the window's color and attributes of the string (size, position, color, and character spacing) with attributes of the cursor (x and y positions, and pressure (Note A)). In this way dynamic relationships can

Note A. Pressure information is derived from the pressure sensitive Wacom input tablet used.

be created for use in more complete communication systems.

Like other interface builders, Eden allows relatively easy high-level construction of interactive compositions. It differs from other builders, such as that by NeXT in the ways in which BadWindows differs from other window systems. In addition, unlike the NeXT Interface Builder, Eden can be used to modify applications that have already been compiled and for which source code is not available.

SCRIPT

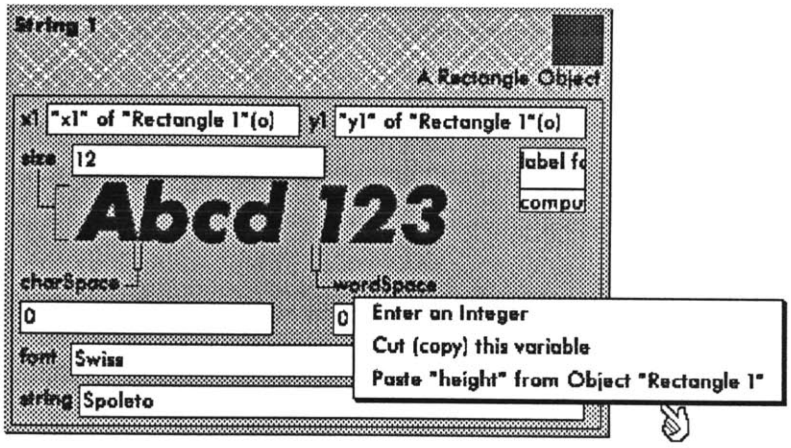
Script (Note B) is an attempt to take the graphic directness of Talk and apply it to the structure made available through Eden. It is intended to allow designers to more easily create complex associations among objects without writing C code (as was required for Eden). It also provides scripting as a means of inspecting object behaviors, storing intermediate design states, and to more easily allow the construction of objects whose behaviors change over time.

Note B. Named Script because it was the first of the described tools to include scripting capabilities.

The structure of objects in Script is similar to that of Eden and BadWindows – objects contain basic information related to their type (rectangles contain their points, strings contain their font and size, etc.) and have a list of attached commands. Unlike BadWindows, the attached commands can be scripted to change over time.

One of the problems with Eden (and, more generally, of working at the level of a C program) is the method of specifying links between objects and variables. In Eden it is not possible for the user to specify links since variables are raw memory addresses, not abstracted objects. Script attempts to solve this problem by categorizing both variables and their values – including pointers – as abstract objects.

With Script, all variables contain information as to what type of value they require. The user can either enter a literal value or link the variable to any other appropriately typed variable. By doing this, values stored in objects



x1 "x1" of "Rectangle 1"(o)
y1 "y1" of "Rectangle 1"(o)
size 12
charSpace 0
wordSpace 0
font Swiss
string Spoleto

Image A: String object editor from Script. Menu displayed for "wordSpace" variable shows options for setting variables.

and activities can easily be connected to one another. (Image A). By clicking on a variable slot in an object editor a menu of choices appears. The choices available are to 1) enter a literal value, 2) copy a reference to this variable, or 3) paste a reference to a variable which has been previously copied. Script does not differentiate object and activity variables, so the two can easily be connected.

Unlike Talk, Script does not include direct visual links between objects. One of the problems that begins to occur with Talk is that the more objects there are the more cluttered the display becomes, reducing the utility of the visible links. Such links could be useful if they were only displayed for objects of interest, but, since Script activities are not objects and do not appear in the same display area as the graphic elements, such links did not seem appropriate. Instead, Okay variables display textual references to their values. Possible references include "32" for a literal value, and "'x1' of 'Rectangle 1' (o)" for a reference to the variable 'x1' of object 'Rectangle 1'. (Note the different values displayed in the variable slots of Image A).

Among the objects provided, Script includes input devices, such as the knob box. Input devices are simple objects which have slots for their values and an activity to poll the physical device for the current value(s). By linking input device values to other objects or activities, a wide range of interactive behaviors can be achieved.

SCRIPTING

Scripting provides the ability to store objects at various states, with the different values of their elements and activities. By using a script designers can examine alternative compositions already defined/specified without having to remember and manually modify all the parameters involved. If designers save intermediate stages of their work, the script can be used to provide a design history. (This history mechanism could also be automated to provide undo capabilities.)

Scripting is also useful because of its ability to interpolate between compositions. Using interpolation allows us to see in-between stages of designs or to create designs whose behaviors change over time. The script can be used to change the commands attached to an object or to change values stored in variables.

A variety of scripting tools have been developed for electronic media. Yet these tools tend to store only object values and not the higher-level information used to determine those values. The script provided in S-Dynamics [Symbolics 1988] provides automatic inbetween of object values and tools to replicate sequences of time, yet all these operations must be performed manually without specifying their purpose. An even more basic scripting tool, provided with VideoWorks [MacroMind 1987] requires users to store generated inbetween values on the script – making key frames more difficult to find. VideoWorks, however, does provide a variety of powerful cut and paste editing tools.

Using a script imposes a structure on compositions which appears to be fundamentally linear. Robin, however, in her work on adaptive multimedia [Robin 1990], has demonstrated that scripts can also be powerful structures for storing hyper-media descriptive information. In future work more complex scripting operations could be added to support such operations as conditional branching and repetition.

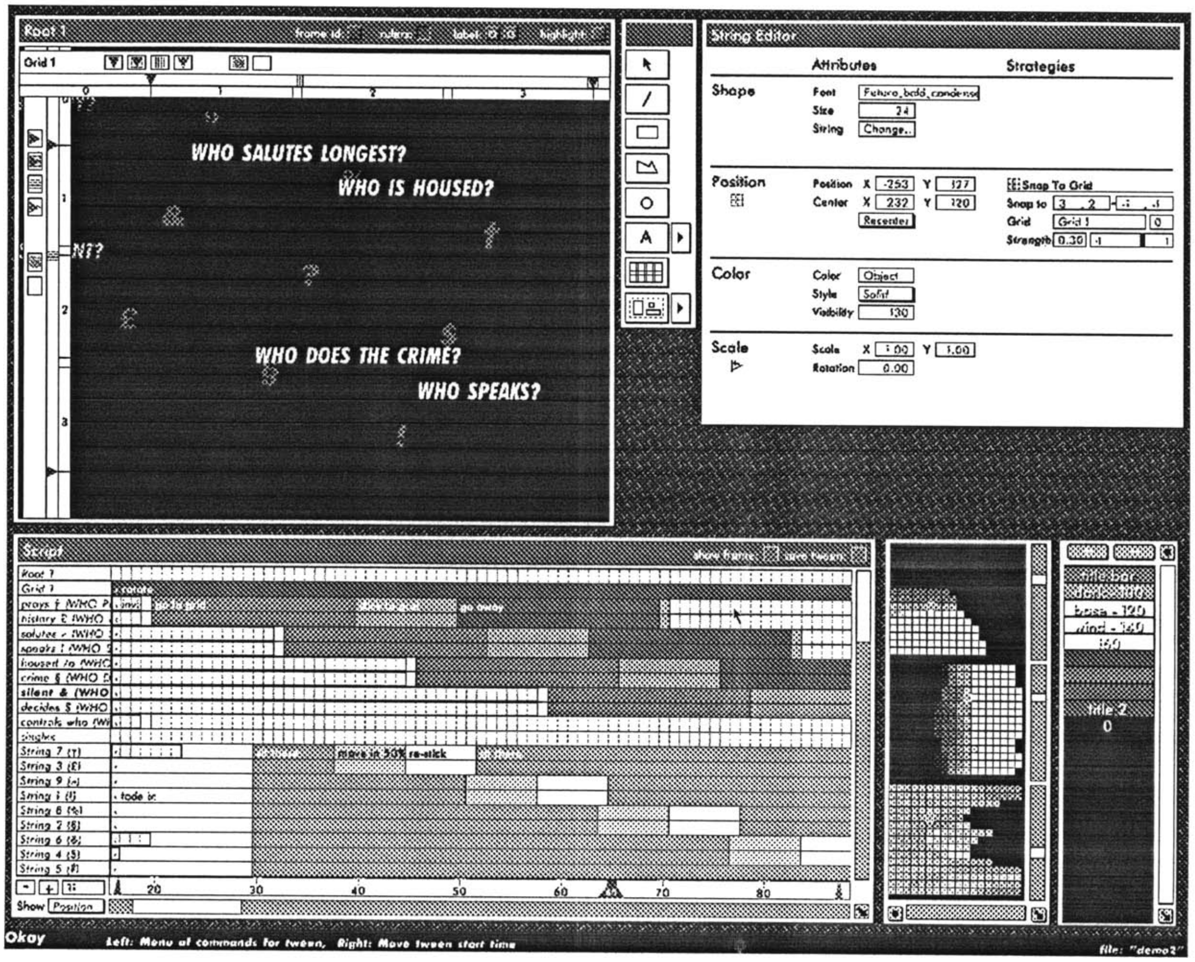
The script implemented for Script has a number of useful features which help the user to better understand the information contained within it. The script can change its level-of-detail by showing time at different resolutions. This way more time can be displayed, but at a lower resolution, or less time can be displayed at a higher resolution. The script allows objects to display themselves using variable numbers of lines, so that their activities can be clearly displayed. Additionally, when an activity is added to an object, the script automatically determines how it should be displayed. Finally, activities have default methods of labeling themselves on the script so that they display some relevant information about themselves. If the user wishes, they can assign their own label and color to the activity.

In building Dynamic Communication Systems that receive input from external sources, we would like a variety of other inputs such as news sources or MIDI control streams. For example, Lytle and Sabiston have each developed tools which assist in automating the synchronization of object behaviors and transformations to musical (MIDI) events. While animations produced by these systems (*More Bells and Whistles* by Lytle and *Beat Dedication* by Sabiston) uses the input to synchronize literal instrument performance, their potential to generate dynamics remains.

OKAY

Okay (Note A), a revised and extended version of Script, explores the specification of designs at a higher level. Specifically, the program is based on the idea that a design is composed at two levels. The first level is that of specific graphic object and their component elements. This level uses an object-oriented graphics system which provides traditional detailed object controls similar to programs such as MacDraw. The second level consists of a higher-level specification facility using principles of abstraction, or strategies, which describe how the object's elements are modified in order to produce a response from the viewer. In addition, Okay also includes a scripting facility to allow storage of design histories, and to explore the application of dynamics to high-level specification. (Image A on the following page shows the interface to Okay.)

Note A. Named Okay for no understandable reason.



String Editor

	Attributes	Strategies
Shape	Font: Futura,bold,condense Size: 24 String: Change..	
Position	Position X -253 Y 127 Center X 232 Y 120 Recenter	Snap To Grid Snap to 3 , 2 - .1 , .1 Grid Grid 1 0 Strength 0.30 -1 1
Color	Color: Object Style: Solid Visibility: 130	
Scale	Scale X 1.00 Y 1.00 Rotation 0.00	

Image A. Okay user interface. The windows (in clockwise order from top left) are 1) root window, 2) object creation palette, 3) object editor, 4) color picker, 5) color palette, and 6) script.

GRAPHIC OBJECTS

Okay provides a variety of graphic object types from which users can build compositions. These types include two dimensional polygons and anti-aliased text. All object types are described by their composite elements and their functionalities. The underlying elements, from which all object are composed, are: Shape, Color, Position, Scale, and Rotation.

The object's functions are responsible for taking information stored in these elements and transforming it into a meaningful representation for its type. For instance, in order to draw itself, a polygon object interprets the points stored in its shape field as a ordered list of vertices, whereas a circle/ellipse object might use those values to determine its centerpoint and radius. By breaking objects into their component elements, and providing the ability to manipulate them directly, we are allowing users to think of the objects as designers conceptualize them – as collections of visual elements which structure a composition.

The structure of object editors is intended to support the perception of objects as sets of discrete elements. While each object type has a different editor, all editors are divided into four horizontal regions corresponding to the four implemented categories of elements: shape, color, position, and scale (which includes rotation). The element editors provide the means to make direct modifications to their values.

GROUPS

A group is a non-graphic object which refers to a set of other items. In order to further encourage users to think of objects as collections of visual elements, groups are defined as collections of similarly typed elements, rather than collections of whole objects. A single object can therefore be referenced by more than one group. By allowing collections of elements, rather than objects, compositions can be organized in terms of their visual structure rather than a more literal collection of objects.

As an example, suppose we have a composition of several objects (Image A). We might want to group the shape elements of objects 1 and 4, and of 2 and 3 so that their shape relations are maintained. We could then perform transformations to the shape groups affecting the shapes of the two objects in a similar manner. Likewise, we might group the color elements of objects 1 and 2, and objects 3 and 4, in order to create separate color regions for the top and bottom of the layout. In this example, objects are referred to by more than one group. Object 1 is a member, with Object 2, of the color element group Group 1; Object 1 is also a member, with Object 4, of the shape element group Group 2.

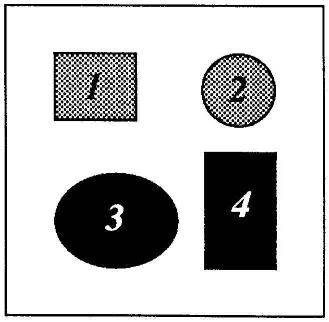
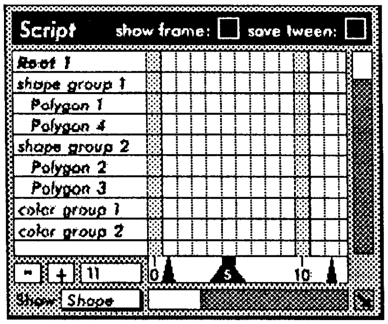


Image A. Composition of four objects. The pairs 1,2 and 3,4 have similar color. Pairs 1,4 and 2,3 have similar shape.

Since an object can belong to more than one group, care must be taken that an element of an object belongs to no more than one group. While allowing an object element to belong to more than one group might result in some interesting designs, the potential for confusion seems to outweigh the initial benefits. Therefore Okay automatically checks that, at creation time, a group does not contain elements that have already been claimed by other groups. Allowing elements to belong to multiple groups, however, might provide interesting results, and merits consideration for inclusion of future work.

Since groups are not linked to entire objects, there is potential for confusion when viewing objects as members of groups. In order to compensate several capabilities have been included in the system in order to more clearly communicate to the user how objects are related to each other. The following describes these capabilities.

SCRIPT DISPLAY OF GROUPS • The list of objects displayed on the script can be reordered according to the element type selected. Using our example, by selecting 'shape' the script will look like (Image A). If we select 'color' it will look like (Image B). It we are not interested in collections of elements, but only the objects themselves, we can view the names in a flat order (Image C). Note that the objects are displayed in the order in which they are draw, from the background forward.

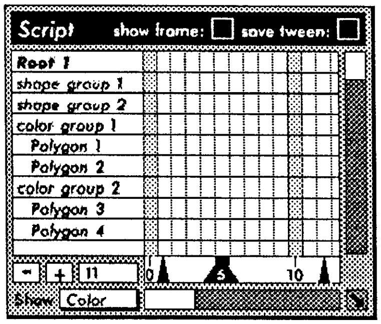


Root 1
shape group 1
Polygon 1
Polygon 4
shape group 2
Polygon 2
Polygon 3
color group 1
color group 2

11

Show: Shape

Image A. Script ordered by shape elements.

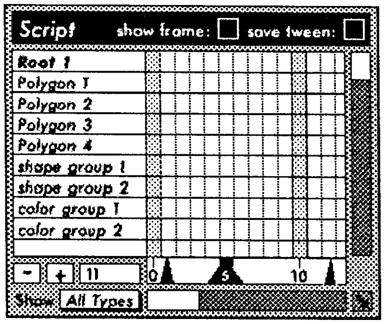


Root 1
shape group 1
shape group 2
color group 1
Polygon 1
Polygon 2
color group 2
Polygon 3
Polygon 4

11

Show: Color

Image B. Script ordered by color elements.



Root 1
Polygon 1
Polygon 2
Polygon 3
Polygon 4
shape group 1
shape group 2
color group 1
color group 2

11

Show: All Types

Image C. Script ordered in object creation order – group hierarchy is not displayed.

SCREEN DISPLAY OF GROUPS • A more graphically direct representation of group relationships is also provided. By activating "highlight objects' on the root window, objects which are not selected are dimmed/darkened. Those objects which are selected are therefore highlighted. If a group object is selected, then the highlighted objects are those that have elements in the selected group.

STRATEGIES

Just as groups operate on objects at the level of their component elements, so too do strategies. As described in the fourth chapter, strategies are transformations performed on an element (or group of elements) of an object. By using strategies, designs (and the design process) can be abstracted from detailed specifications to more general concepts of organization and design. This abstraction can then also be applied to dynamics and the techniques used to generate them.

With Okay, strategies are used in order to explore methods of structuring compositions. At the start of the work the definition of a strategy was relatively specific. Namely, it was the category of visual principles which describe a set of specific techniques that relate formal aspects of visual elements and how they relate to one another. These techniques are often used to describe the structure of a composition both during its construction and its analysis. Strategies appeared to be valuable visual methods of structuring a composition at a variety of levels.

The first strategy to be implemented was symmetry. Reflexive symmetry was used, in which a point on one side of an axis of symmetry is exactly reversed, as in a mirror, on the other side of the axis. However, after implementing several types of symmetry, for different element types, it became clear that the definition of Strategy with which the project began was simultaneously too general and too specific. It was too general because visual communication techniques tend to be general and far reaching. They tend to significantly overlap with each other. Any single technique could be the subject of extended study (for example see [Loeb 1971]). To further explore the interrelations of multiple techniques is beyond my scope of describing designs at high levels. The definition of Strategy was also too specific since strategies tend to be formal, structured ways of describing compositions. This structure would tend to require a significant amount of pre-planning when creating a composition which is detrimental to improvisational processes. We therefore generalized the definition of Strategy to simply be general principles and methods of organizing a design. While this definition is somewhat broad (it considers groupings of objects or elements a strategy) it does permit a more varied, less formal and structured approach to design – something which supports the design process.

Similar methods of structuring compositions have been done using constraint systems [Borning 1987]. Using constraints complex relationships between elements of objects can be defined and maintained. However the focus of Okay was whether the *application* of strategies was a useful method of organizing designs. Based on these results a constraint system might later be introduced in order to allow interactive construction of additional strategies.

With Okay, in order to apply a strategy to an element of an object, that object must first be selected so that its editor is available. On each object editor, each element lists the strategies which are defined and available for it. To apply a strategy a user clicks the appropriate strategy's icon resulting in a sub-editor appearing in the strategy column of the object's editor (see the object editor in Image A on page 36). This strategy editor is used to specify the parameters the particular strategy needs in order to operate on the object element.

One common feature of strategy editors is the ability to vary the degree to which a strategy is applied. This is determined by a strategy strength value between -1.0 and 1.0. A value of 0.0 indicates that the strategy is not active (Image A). This means that the value of the element is that of the base object. A value of 1.0 indicates that the element's values have been set to those values which have been computed by the strategy (Image B). In between values are interpolated between the base-object and strategy values. Negative values, are intended to mean the opposite of a strategy. However negative values are simply further interpolations away from the 1.0 value, rather than towards a negative version of the strategy (Image C).

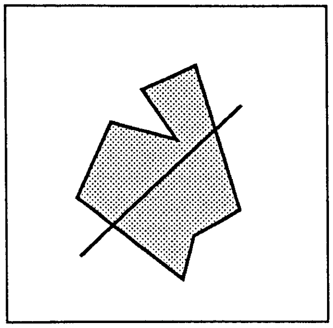


Image A. Shape of polygon as entered by the user.

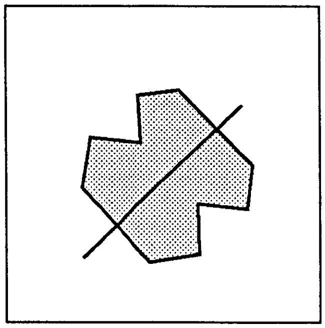


Image B. Polygon with shape symmetry strategy applied to maximum (1.0) value.

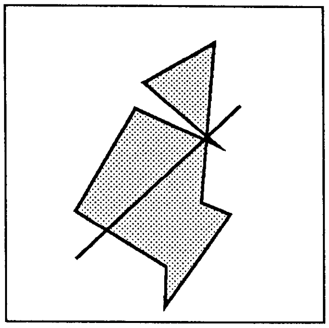


Image C. Polygon with symmetry strategy applied with negative (-1.0) value.

Every time an object is displayed (or any other reference to it is made) its current strategies recompute their element value(s). This form of constraint is performed in order to support hierarchical designs. As a result, if an object's shape is modified by a strategy which is based upon another object, then if that other object changes shape, so too does the primary object.

Just as an object's elements can belong only to a single group, so too an element can only have a single strategy modify its characteristics. In future work it would be interesting to see how the application of multiple strategies to a single element might be used to create less predictable compositions by using those transformations which result at the intersections of particular strategies. However, mentioned earlier, that was beyond the scope of this particular project, but would be interesting in further work.

The following describes three of the strategies implemented in Okay.

SYMMETRY • Symmetry is a form of axial balance in which an element on one side of an axis is reproduced exactly on the other side. For polygons symmetry can be applied in order to make its shape symmetric (Image A). For groups of object position elements, symmetry modifies the position of the individual objects so that they are made symmetric (Image B).

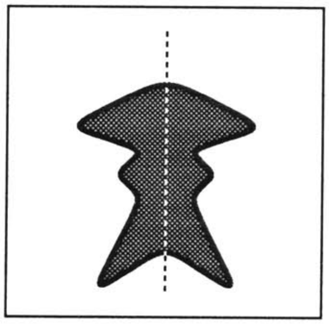


Image A. An object whose shape has been made symmetric with respect to the axis.

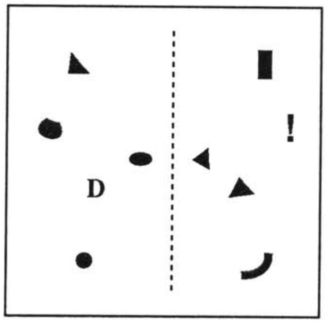


Image B. Collection of objects whose positions have been made symmetric with respect to the axis.

GRID RELATIONS • In order to relate elements of an object to a grid, a grid object must first be created. Grid objects include horizontal and vertical rulers with which grid columns can be defined. A single grid column lines can be fixed or positioned relative to an object (so that as the object moves the grid changes), or a set of grid lines can be distributed between other columns with or without gutters (Image A).

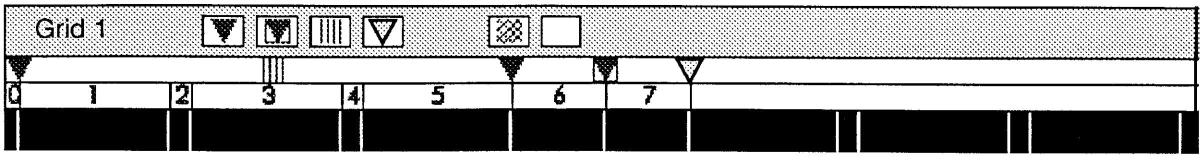


Image A. Grid editor ruler. The tabs placed, from left, are a fixed tab, a series tab (creating columns 1-5), a fixed tab, a connected tab (connected to a graphic object [not shown]), and a repeat tab. Lines below ruler (in black area) indicate grid columns.

An object can be attached to a grid region either by shape or position. (A grid *region* is any rectangular area defined by the grid.) If the object is attached by shape it will adjust its shape so as to be the size of the grid region specified. When attached by position, its position will change so that it is aligned to the specified grid region (either its left, right, top, or bottom sides can be justified, or it can be centered).

ANGULAR RELATIONSHIPS • In order to relate angular components of objects, Okay provides the ability to rotate an object so that an edge of it is aligned to the edge of another object (Image B). This strategy currently can only be applied to the rotation element of polygon and line objects. The angular strategy editor provides the ability to specify in what direction an object is rotated when the strategy is applied (either clockwise, counterclockwise, closest, or longest).

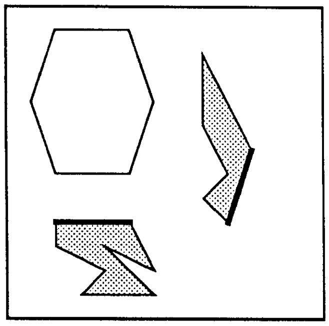


Image B. The shaded objects have been rotated so that their highlighted edges align with associated edges of the hollow object.

SCRIPTING

The script used in Okay is similar to that used in Script. It provides the ability to change the amount of time displayed (effectively changing the amount of detail displayed on the script) and to associate text and color labels to activities on the script. Rather than holding object activities it stores values for objects and their attached strategies. Using these values, interpolation is performed to generate inbetween compositions and dynamics. In addition, by interpolating between two strategies, scripting provides a method of applying more than one strategy to an element.

Although only suggested at in this work, scripting can provide the ability to develop strategies which are, in their nature, active. These strategies might somehow be higher level scripting controls which describe dynamic changes (possible including interaction) in a composition. However, the script in Okay does provide a variety of interpolation types: linear, slow-in, slow-out, slow-in-out, and random.

6. CONCLUSIONS

This thesis has described both theoretical and implemented work that explores the use of elements newly available in active and interactive electronic media. Specifically, it has explored how non-verbal messages can be conveyed using dynamics, and how these messages can assist in communicating information. These principles can be applied to creating Dynamic Communication Systems for use in environments as varied as hyper-media news systems, user interface design, music videos, and personal expression. But there is still a lot more work to do.

Among the tools described in this thesis, the controls used to express ideas to the computer are still too specific and detailed. The user still does a lot of specification at a non-graphic level. In future work it would be better if there was more direct interaction with the elements involved rather than using the artificial object extensions of editors. One problem, however, is that the current systems treat objects as static with activity *applied* to them. The meaning of an object that is *fundamentally* active needs to be explored. (Perhaps interface problems might then arise from how one "catches" an active object.)

While the notion of strategies to structure compositions has proven useful in those cases where the strategies available matched the design goals, it is too difficult to for the user to implement new strategies. Perhaps future work could be done with constraint-based systems in order to more easily create definitions of strategies. Currently, however, the process of defining the desired constraints seems to be too removed from the design process since it requires a too low level conceptualization of the problem – one which is not intuitive enough.

A more fundamental problem of designing with strategies is that they are too precise. Although a designer can alter the intensity with which a strategy is applied, they still are choosing a single rule (or set of rules) to be applied in a rigid way. With strategies we are designing by applying predefined structures to a composition rather than one closer to sketching more abstract ideas. Sketching on paper is a powerful way of exploring ideas because "the graphical nature of the drawing has important meanings, meanings that must not be, but are, for the most part, overlooked in computer graphics" [Negroponte 1975, p65]. We need tools that can take advantage of that meaning.

Finally, and perhaps most importantly, we would like more complex scripting capabilities. When designing with active elements, it is the script which controls their action. A script which could have activities applied to itself (not just its objects) would be a powerful way of creating dynamic strategies.

BIBLIOGRAPHY

Alexander, Christopher (1979). *A Timeless Way of Building.* New York, NY: Oxford University Press.

Arnheim, Rudolf (1986). *New Essays on the Psychology of Art.* Berkeley, CA: University of California Press.

Borning, Alan [et al] (1987). *Constraint Hierarchies.* In: *OOPSLA Conference Proceedings*, special issue of *SIGPLAN Notices.* Vol. 22, No 12, December.

Cage, John (1973). *Silence (Lectures and Writings by John Cage).* Middletown, CT: Wesleyan University Press.

Csikszentmihalyi, Mihaly & Robinson, Rick E. (1990). *The Art of Seeing: An Interpretation of the Aesthetic Encounter.* Malibu, CA: The J. Paul Getty Trust.

Dondis, Donis (1973). *A Primer of Visual Literacy.* Cambridge, MA: The MIT Press.

Eno, Brian (1978). *Ambient #1: Music for Airports.* Reseda, CA: E.G. Music Ltd.

Friedman, Mildred [et al] (1989). *Graphic Design in America: A Visual Language History.* Minneapolis, MN: The Walker Art Center.

Gerstner, Karl (1974). *Compendium for Literates: A System of Writing.* Cambridge, MA: The MIT Press.

Kepes, Gyorgy (1961). *Language of Vision.* Chicago, IL: Paul Theobald & Company.

Klee, Paul (1953). *Pedagogical Sketchbook.* Bauhaus Books, 1925; rpt. London: Farber and Farber.

Klee, Paul (1961). *Notebooks Volume 1: The Thinking Eye*. London: Lund Humphries. Jürg Saller, ed.

Lieberman, Henry. *Teaching a New Dog Old Tricks*. Cambridge, MA: unpublished.

Lieberman, Henry (1988). *Toward Intelligent Interfaces for Graphic Design Applications*. Proceedings of the International Conference on Fifth Generation Computer Systems. ICOT, ed.

Loeb, Arthur L. (1971). *Color and Symmetry*. New York, NY: Wiley-Interscience.

MacroMind Inc. (1987). *VideoWorks II*. Chicago, IL: MacroMind, Inc.

McCorduck, Pamela (1991). *Aaron's Code: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. New York, NY: W. H. Freeman and Company.

Minsky, Marvin L. (1981). *Music, Mind, and Meaning*. Cambridge, MA: MIT, Artificial Intelligence Laboratory.

Muller-Brockmann, Josef (1981). *Grid Systems in Graphic Design: A Visual Communication Manual for Graphic Designers, Typographers, and Three-Dimensional Designers*. New York, NY: Hastings House.

Negroponte, Nicholas (1975). *Soft Architecture Machines*. Cambridge, MA: The MIT Press.

Rand, Paul (1985). *A Designer's Art*. New Haven, CT: Yale University Press.

Robin, Laura H. (1990). *Personalizing Hypermedia: The Role of Adaptive Multimedia Scripts*. M.S.V.S. Thesis in Media Arts and Sciences, MIT.

Shyer, Laurence (1989). *Robert Wilson and His Collaborators*. New York, NY: Theatre Communications Group.

Steele, Guy Lewis, Jr. (1980). *The Definition and Implementation of a Computer Programming Language Based on Constraints*. MIT AI Laboratory. Tech Report 595.

Sutherland, Ivan E. (1963). *Sketchpad: A Man-Machine Graphical Communication System*. Lexington, MA: MIT Lincoln Laboratory.

Symbolics, Inc. (1988). *S-Dynamics*. Document #980191, Cambridge, MA: Symbolics, Inc.

Weitzman, Louis (1988). *Designer: A Knowledge-Based Graphic Design Assistant.* MCC Technical Report Number ACA-HI-017-88.

Wilenski, R. H. (1927). *The Modern Movement in Art.* New York, NY: Frederick A. Stokes Company.

Wingler, Hans M. (1969). *The Bauhaus: Weimar, Dessau, Berlin, Chicago.* Cambridge, MA: The MIT Press.

Wozencroft, Jon (1988). *The Graphic Language of Neville Brody.* New York, NY: Rizzoli International.