# Eloquent Scenery: A Study of Peripheral Visual Communication

by

**Robert Charles Mollitor**

B.S., Computer Science
B.S., Physics
Massachusetts Institute of Technology
Cambridge, Massachusetts
1988

SUBMITTED TO THE MEDIA ARTS AND SCIENCES SECTION, SCHOOL OF ARCHITECTURE AND PLANNING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE

AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1990

[illegible]

Signature of the Author

.......................................................................... [illegible]

Robert C. Mollitor
Media Arts and Sciences Section
February 28, 1990

Certified by

..........................................................................

Walter R. Bender
Principal Research Scientist, Media Laboratory
Thesis Supervisor

Accepted by

..........................................................................

Stephen A. Benton
Chairman
Departmental Committee on Graduate Students

# Eloquent Scenery: A Study of Peripheral Visual Communication

by

Robert Charles Mollitor

Submitted to the Media Arts and Sciences Section, School of Architecture and Planning on February 28, 1990 in partial fulfillment of the requirements of the degree of Master of Science at the Massachusetts Institute of Technology

## Abstract

This thesis proposes a mechanism of background task communication for workstations. Dynamic visual objects called Gestures are used as signs by the computer to reach an otherwise occupied user. The primary, dynamic visual properties of these Gestures are position, texture, intensity, chromaticity, stain, orientation, shape, and size. The dynamism of each property is defined by a set of secondary, static properties. The valued properties are defined by an offset value, a value path, a path extent, and a path period. The imaged properties are defined by a collection of images, an index path, and an image rate.

The different properties can be allocated independently for different message roles. This thesis considers how strong the various properties are at attracting the user's attention for important messages, at visually accenting messages so that relevant messages can be found quickly, and at providing simple information. The specific types of information considered are quantitative or continuous, valued information; ordinal or discrete, ordered information; and nominal or discrete, unordered information. This thesis also briefly considers the coordination of the various properties to form signs analogous to the gestural signs of deaf sign language.

Thesis Supervisor: Walter R. Bender
Title: Principal Research Scientist, Media Laboratory

This work was supported in part by International Business Machines Incorporated.

# Contents

# Chapter 1

# Introduction

At any given moment active computer tasks can be divided into two sets. The first set consists of foreground tasks: those tasks which have the immediate and prolonged attention of a human user. The second set consists of background tasks: those tasks which receive at best only cursory attention from a human user. Foreground tasks often have their graphic presentation carefully designed to minimize frustration and maximize productivity. However, background tasks, especially those designed to run in the foreground, often have a limited ability to present information to the user. They may use a method which is ill-conceived in view of the demands of some of the foreground tasks it will run behind or the clutter from other background presentations.

This thesis presents a system of communication, called Eloquent Scenery, to be used expressly by background tasks. Alternatively, the system can be used to present ancillary information for complex foreground tasks.

The design of this system was motivated primarily by concerns for the availability of necessary or wanted information, the ability to maintain an ordered hierarchy of background information, and the adequacy of schemes for moderated intrusion by important information. Eloquent Scenery addresses the first concern by exploiting the extra display space made available by larger video monitors and multi-monitor workstations. As is done with an automobile's dashboard, background information is presented in the peripheral field of the user's vision, available for quick inspection either directly or out of the corner of one's eyes. Some applications already use this technique in their interface, but only for their own side information when they are run in the foreground.

Eloquent Scenery is a system all applications can use equally. It uses individual visual objects, each with a relatively large number of independently determined, dynamic properties. These properties can flexibly communicate a wide range of separate, evolving statements of information. I call the visual objects *Gestures*, and their dynamic visual properties are position, texture, intensity, chromaticity, stain, orientation, size, and shape.

As workstations become more powerful, users can utilize this additional power by running more and more tasks concurrently. Many of these tasks may be seemingly unrelated to each other. In order to coordinate messages from these disparate tasks, Eloquent Scenery needs a mechanism for visually ordering its Gestures. For instance, more important messages can use striking visual characteristics, and less important ones can use subtle visual characteristics. This thesis presents a rough experiment to grade the strength of the various visual properties.

Although static or slowly changing visual properties are sufficient for forming a visual hierarchy in the peripheral field, they do little to attract the eye of the user from the foreground task. For example, a large, full-white square off to the side may attract attention upon its initial appearance, but it will quickly blend into the rest of the background and be forgotten by the user until the user consciously looks for it in the future. However, if it were to, say, blink, it would continually attract attention, and distract the user from the task at hand. While this distraction is generally undesirable, there are circumstances where the background task has urgent information that requires an immediate user response. Prolonged rapid changes of visual properties can be used to notify the user of important information. The effect of this intrusion cannot overstep the importance of the message, otherwise the user will eventually disable the entire system in order to get work done. This thesis presents an experiment to grade the annoyance of some dynamic variations of visual properties.

The visual properties that are not devoted toward defining a heirarchy or heightening distraction can be used to carry information directly. Either they can operate independently, where the value of a property reflects the content value, or they can operate in conjunction, where the presence of a set of particular properties denotes a particular message.

## 1.1 Thesis Organization

Here is the organization of the chapters of this thesis. Chapter 2 introduces the dynamic visual objects named Gestures and their design details. Chapter 3 describes the two experiments mentioned above, and presents some test results. Chapter 4 describes two general theories of signs, and mentions some examples of indirect information currently available from computer workstations. Chapter 5 questions the effectiveness of using Gestures to present information from a general point of view. Chapter 6 suggests some uses of Gestures and particular Gesture properties, and proposes a way of creating a general language. Chapter 7 considers some points for a possible, future development of a two-way or conversational communication system. Finally, Chapter 8 has my conclusions.

# Chapter 2

# The Gesture

In this chapter, I introduce dynamic, visual objects called Gestures. The first two sections discuss the design and the implementation of Gestures. The final section describes the proposed environment of Gestures called Eloquent Scenery.

## 2.1 Design

A Gesture is a visual object with dynamic primary properties and static secondary properties. These properties define the behavior of the Gesture.

### 2.1.1 Primary Properties

The primary properties can be grouped into the position property, the facial properties, and the silhouette properties. The position property determines the location of a Gesture on the screen. The facial properties specify the use of color information on a Gesture's face. The silhouette properties affect the boundary of a Gesture's face.

#### The Position Property

Position locates the logical center of a Gesture. The logical center is the center of the Gesture's bounding rectangle and not necessarily the center of the Gesture's visible face. The shape property may mask the Gesture's face unevenly, causing the face to be lopsided within the bounding rectangle.

#### The Facial Properties

The facial properties of a Gesture are texture, intensity, chromaticity, and stain.

The texture of a Gesture is the spatial arrangement of colors on the visible region of the Gesture object, on the Gesture's face. The term texture is used loosely and does not imply an abstract, repeating pattern, but rather any image, where an image is defined as a rectangular arrays of color elements. How these images are handled will be discussed in the implementation section and the appendix.

Intensity, chromaticity, and stain scale the natural color values of the texture. Intensity scales the luminance of the individual colors. Higher intensity means brighter colors and lower intensity means dimmer colors. Chromaticity scales the saturation of individual colors. Higher chromaticity means richer colors and lower chromaticity means ‘greyer’ colors. Stain scales the red, green, and blue components. Increasing, say, the red stain simultaneously increases the value of the red component and decreases the values of the green and blue components. Intensity and chromaticity are orthogonal to each other, except when color components saturate or vanish. Changes in one do not cause changes in the other. As defined in this implementation, stain is not strictly orthogonal to either intensity or chromaticity. That is, staining may brighten or darken colors, or it may move colors nearer to or farther from a grey value. Figure 2.1 describes the formulae used to calculate these color scaling properties.

### The Silhouette Properties

The silhouette properties of a Gesture are orientation, size, and shape.

Orientation rotates a Gesture with respect its texture’s natural, upright configuration. The positive direction of rotation is counter-clockwise about the Gesture’s logical center, and the value of orientation is in radians.

Size scales the horizontal and vertical extents of the Gesture’s face. Specifically, texture images are scaled to the appropriate width and height.

Shape determines which of the texture’s color elements are visible and not visible on the Gesture’s face. In other words, shape outlines the visible face. Masks are used to discriminate between visible and non-visible at the color element level. These masks are pseudo-images, or rectangular arrays of boolean elements, and are scaled and rotated so that their elements correspond to the scaled and rotated color elements of the texture.

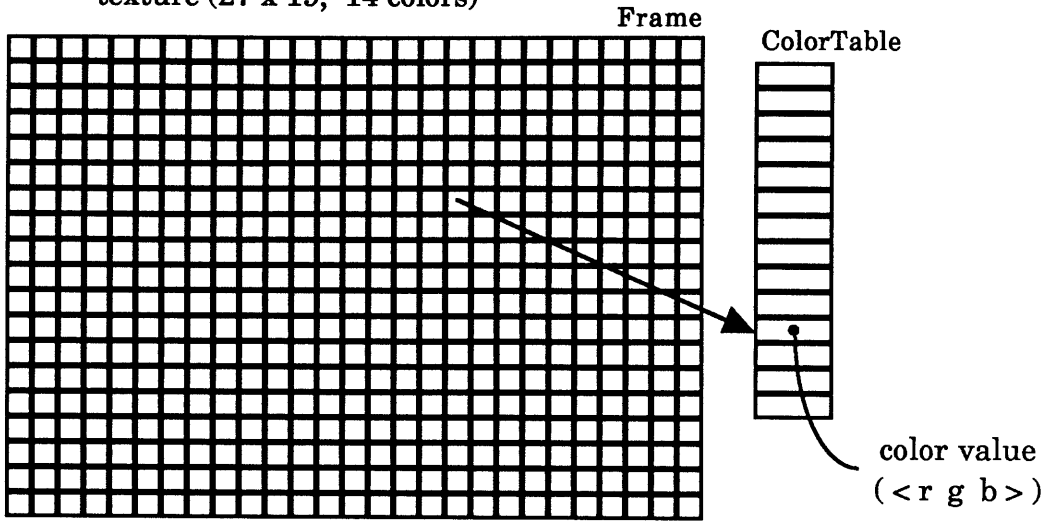
Figure 2.2 shows the steps involved in combining the silhouette properties.

intensity (In)

chromaticity (Ch)

$$\text{stain}\begin{cases}\text{redness (Rn)}\\ \text{greenness (Gn)}\\ \text{blueness (Bn)}\end{cases}$$

texture (27 x 19, 14 colors)



$$y = .30\,r + .59\,g + .11\,b$$

$$\text{red} = \frac{In}{3}\,(Ch \cdot (r - y) + y)\;(Rn + \frac{1}{Gn} + \frac{1}{Bn})$$

$$\text{green} = \frac{In}{3}\,(Ch \cdot (g - y) + y)\;(Gn + \frac{1}{Rn} + \frac{1}{Bn})$$

$$\text{blue} = \frac{In}{3}\,(Ch\,(b - y) + y)\;(Bn + \frac{1}{Rn} + \frac{1}{Gn})$$

Figure 2.1: Facial properties.

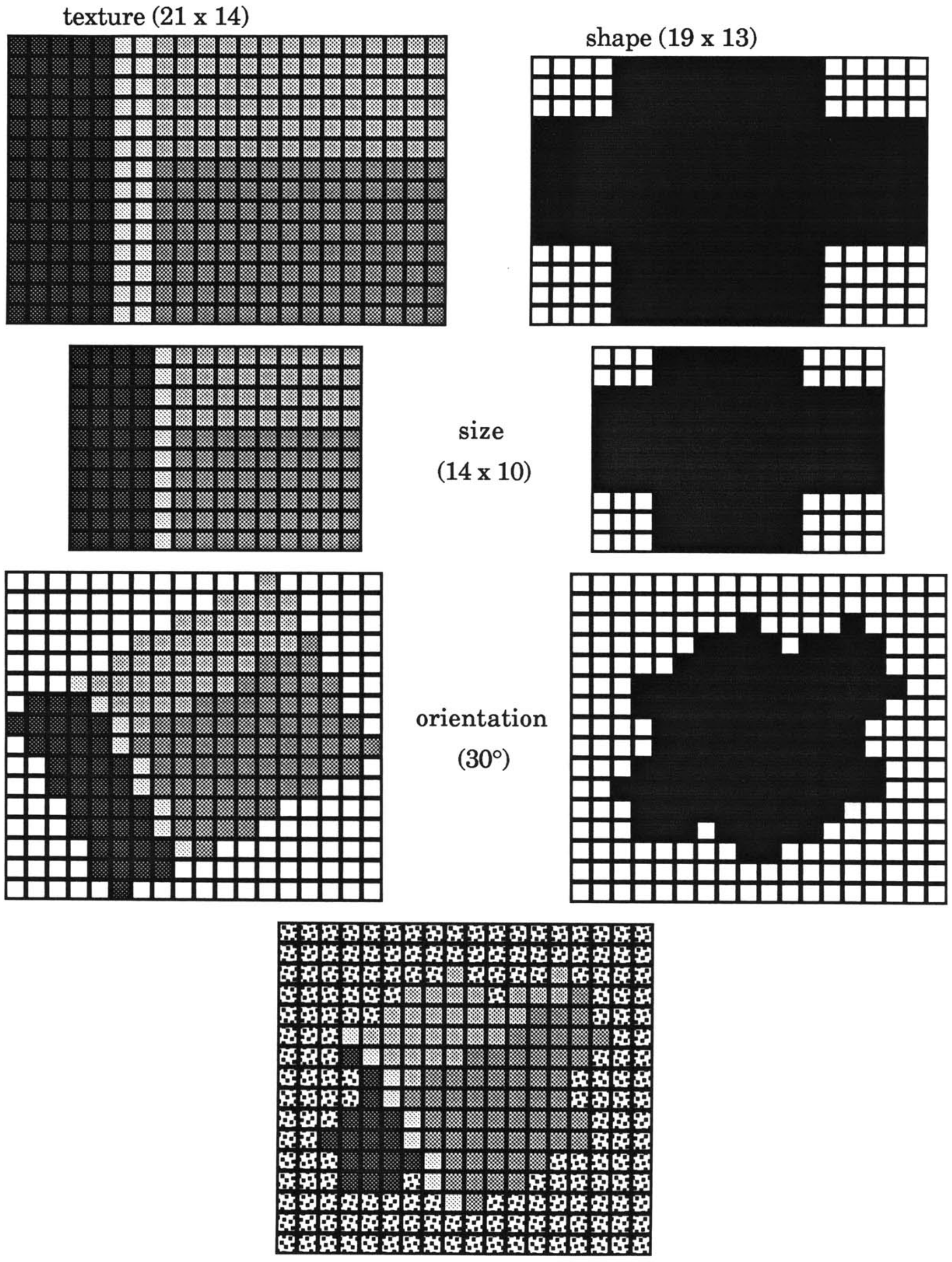


Figure 2.2: Silhouette properties.

### 2.1.2 Secondary Properties

Secondary properties define the dynamism of primary properties. These secondary properties are themselves static. The primary properties can be divided into those that use one, two, or three values, and those that use images or pseudo-images. Both types of properties use the path secondary property. Those of the first use paths to determine values, and those of the second use paths to index images. Properties that use value paths also have extents, offset values, and perhaps cyclic periods as secondary properties. Properties that use index paths have collections of images and rates of index increment as secondary properties.

A particular path may be defined in either of two ways. In the first way, the property is predestined and cyclic. In the second way, the property is synchronized to an event or variable outside of the data which forms the actual Gesture object.

**Predestined Paths** A predestined path is specified upon a Gesture's creation. Internal time variables mark the current position of the properties along their paths. The task which maintains the set of Gestures routinely updates each individual Gesture in some pattern. When each Gesture is updated, all of the time variables of its dynamic properties are increased accordingly. The new time values mark new positions along the predestined paths. Time variables over continuous or piece-wise continuous paths are used instead of a step-by-step script to allow flexibility. The update schedule of the set of Gestures may change over time, or the processor may be taxed in different ways at different times, causing unequal time increments. Due to these arbitrarily sized time increments, the exact points touched along the path for a given property may not match those touched in an earlier or a later cycle over the same path.

**Synchronized Paths** A synchronized path is tied to an external function, or perhaps to a particular external variable, upon a Gesture's creation. Although time variables are maintained for these paths as well, and the time values are passed to whatever external functions are called, the external functions are free to ignore these time values and calculate their return values in any way.

### Valued Properties

The secondary properties of position are offset position, path of motion, extent of motion, and, if applicable, period of motion. The offset position is an absolute location within the screen's coordinate system. The position of a Gesture might never actually equal the offset position, depending on how the path of motion is defined. The path of motion results from the interaction of two paths, namely the horizontal position path and the vertical position path. These two paths are defined independently and may be each either predetermined or synchronized. The extent of motion is a spatial scale factor. A larger extent means a wider or higher path. Of course, if the position of a Gesture is fixed, then the extent of motion is meaningless. The period of motion only applies to a predestined, non-stationary path of motion. It is the amount of time required to traverse the path. A shorter period implies faster motion. The secondary properties of position are illustrated in Figure 2.3.

Strictly speaking, both horizontal and vertical components of position have their own extent and period, but a predetermined path whose components scale independently in space or time may lose its two-dimensional identity. This identity may not always be important, but it may be a feature more easily recognized by a viewer than the component one-dimensional identities.

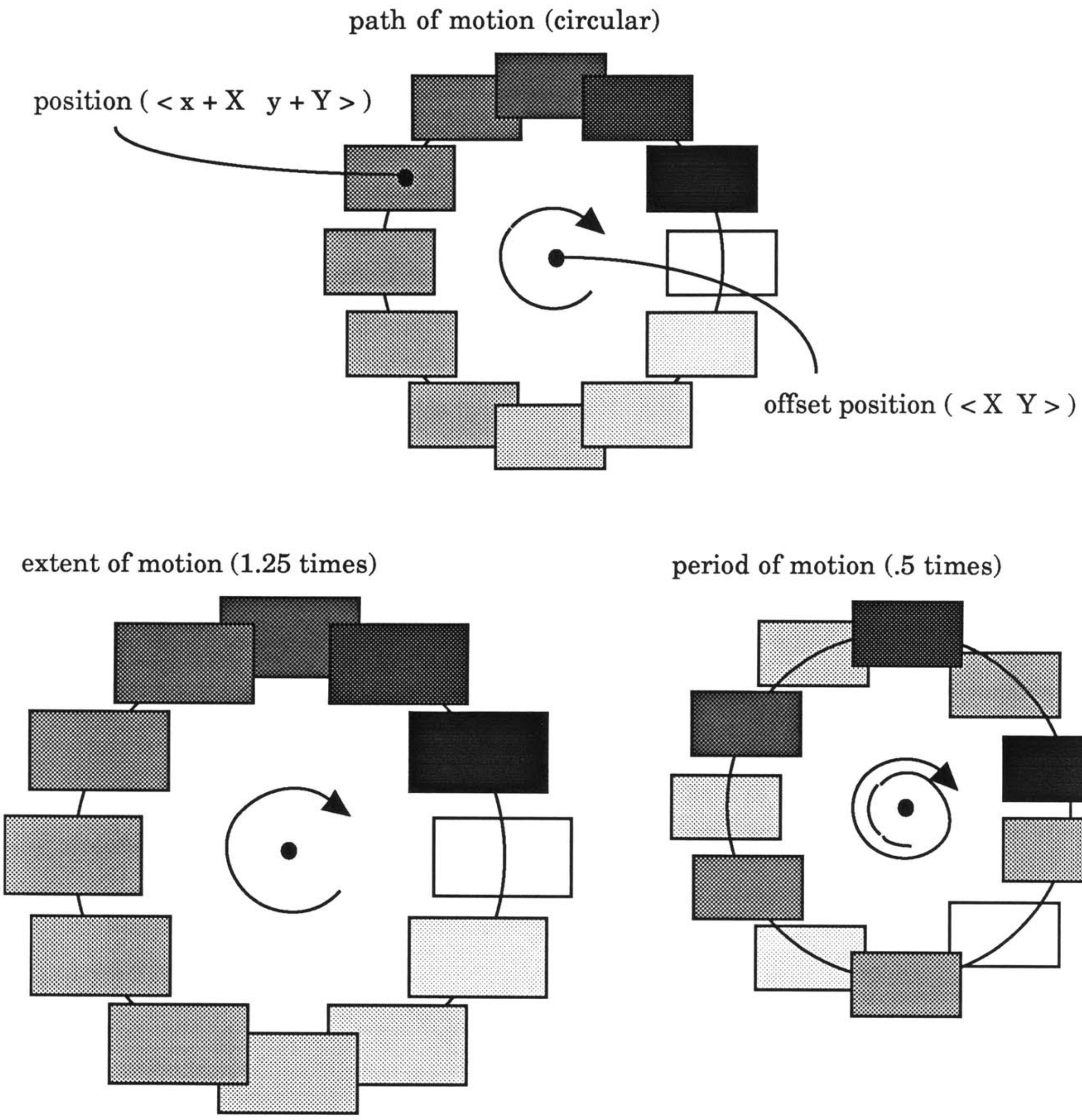

path of motion (circular)
position ( < x + X   y + Y > )
offset position ( < X   Y > )
extent of motion (1.25 times)
period of motion (.5 times)

Figure 2.3: Secondary properties of position.

Size has secondary properties analogous to those of position. In fact, for a Gestures with a stationary position, shape, and orientation, changes in position of the upper-righthand corner describes changes in size, but for a factor of two.

Orientation, intensity, and chromaticity are single valued properties and have a single extent, offset, and period each. Stain has three independent values, so it has three extents, periods, and offsets.

The range of orientation folds in upon itself. An orientation value of 2 pi radians is identical to zero radians. Extent for orientation is within an angle of rotation, which cannot exceed 2 pi radians.

The normal, unscaling values of intensity, chromaticity, and the three stains are all one. The extents of intensity and chromaticity are bands in the range zero to infinity. Zero intensity implies all of the colors become black. Zero chromaticity implies all of the colors become a shade of grey. The extent of each stain is some band in the range one to infinity. For high values of any of these properties, some non-zero color components may saturate to their full value or vanish at zero.

## Indexed Properties

A collection of textures is a fixed set of images. A collection of shapes is a fixed set of pseudo-images.

The index increment per second is not directly related to the true update rate of a Gesture. As mentioned above, Gestures may be updated with arbitrarily sized time increments. The number of updates per second may not be constant. The frame rate of images or masks is kept constant by skipping or repeating frames. This skipping or repeating is used to speed up or slow down the apparent frame rate with respect to the natural frame rate of a collection.

## 2.2 Implementation

This section describes the computing environment used for Gestures and presents a brief outline of the data types used in this project. The appendix at the end of this thesis offers a more detailed description of the data types.

### 2.2.1 Hardware

The work of this thesis was done on an IBM PS/2 Model 80 with an IBM 8514/A Display Adapter. The adapter was used in the advanced function mode with screen dimensions of 640 pixels wide and 480 pixels high. Off-screen memory round out the dimensions of the total frame buffer to 1024 pixels wide and 768 pixels high. The screen is updated at 60 Hertz, progressively scanned. Each pixel contains an 8 bit value which references a 256 entry color lookup table which is 24 bits deep, 8 bits each for red, green and blue. The PS/2 has a math co-processor and enough memory to form a virtual disk, also known as a RAM disk, with between 1 and 7 megabytes.

### 2.2.2 Images

In this project, an image is composed of a Frame and a ColorTable. A Frame is an array of color codes, and a ColorTable is a codebook for converting color codes into

24 bit color values. A pseudo-image differs from an image in that it does not have ColorTable for its Frame. The pseudo-image's codes are not used to define colors.

A data object named a Reel is used to collect the individual images or pseudo-images of a sequence together into a collection. The Frames and ColorTables of the image are copied from potentially random locations on a hard disk to consecutive locations on a virtual disk.

### 2.2.3 Dynamic values

A data object named a Track is used to determine the path of dynamic values and indices. When a Gesture is updated, all of the subsidiary Tracks are traced an amount equal to the time that has passed since the last Gesture update. The returned values of the Tracks become the new property values or indices. As described above, the returned values may be predetermined or synchronized to external events.

Predetermined paths can be straight lines, a series of spline segments, or any path described by an application provided function. StraightTracks, SplineTracks, and FunctionTracks trace the three types of predetermined paths, respectively. Any application provided function must take a generic data structure descriptor and a floating-point time argument.

External events are independent of any time variable that a Track may maintain. The term external is used to denote a part of the application which is outside of the data devoted solely to Gestures and Tracks. The simplest external event is the state of an application variable. Whenever a VariableTrack is traced any amount, the current value of the application variable is returned. Similarly, when a MouseTrack is traced any amount, the current state of the specific system mouse parameter is returned.

The most general external event is filtered through an application provided function. These functions are the same as those described above. The time argument may be ignored by the function itself, but the FunctionTrack maintains and passes the time value anyway.

## 2.3 Eloquent Scenery

The habitat of Gestures is called Eloquent Scenery. I chose this name to accent two qualities. First, the Gestures are intended for the background, or rather the periphery. They are not intended for direct use. Second, the Gestures can carry information. They are not just fancy 'icons' of background processes, although it would be easy to degenerate them into that.

Currently, Gestures are molded in isolation of one another and do not normally interact with one another. This is not an enforced restriction, but a convenience. The obvious way to have two Gestures interact is to link them via either VariableTracks or FunctionTracks. Although the interplay of a pair or, better yet, a constellation of Gestures would add dimensions to the expressive capability, it is not considered in this project. One reason is that Puck performance degrades as the number of Pucks is increased. Another reason is that allowing the possibility of interaction introduces a whole new set of visual properties that need to be specified and examined. Finally, in this project, the presentation of a number of distinct messages in parallel is a goal, so confusing the integrity of messages is potentially detrimental to this goal.

# Chapter 3

# Experiments on Gesture Properties

This chapter contains descriptions of two experiments on the implementation of Gestures presented in this thesis. The first experiment grades the degree of visual prominence for static Gesture property values in the presence of positional motion. The second experiments grades the degree of peripheral distraction caused by the dynamism of the various primary Gesture properties.

## 3.1 Experiment 1: Visual Prominence

As will be mentioned later, visual prominence can be used to order Gestures so that the user can quickly categorize the Gestures presented in parallel at any given moment.

This experiment relates the prominence of different visual properties to each other using the untutored, immediate impressions of some human subjects.

### 3.1.1 Method

A subject is presented with a series of trials. Each trial displays exactly three Gesture: one in the upper-lefthand corner (offset position of (140, 340)), one in the upper-righthand corner (offset position of (500, 340)), and one in the lower center (offset position of (320, 140)). From these three Gestures the subject chooses the most prominent one using whatever criteria feels natural. The subject then enters a choice by pressing one of the '7', '9', or '2' keys, depending on which key's position on the numeric keypad corresponds to position of the chosen Gesture. These keys are found on the upper-lefthand corner, upper-righthand corner, and lower center regions of the numeric keypad, respectively. Once a choice is made, the screen is cleared, and the next trial is presented.

Twenty-six Gestures were used in this experiment, one of which was considered the control Gesture. A trial consisted of the control Gesture plus two randomly chosen, different, and non-control Gestures. The offset positions of the Gestures was also randomly determined, so that the offset position of the control Gesture varied from trial to trial. Ignoring offset position, the twenty-five non-control Gestures yield six hundred possible trial pairs.

Figure 3.1 shows the Gesture property values used for this experiment. The control Gesture has the following property values: *pour* texture, *normal* intensity, *rich* chromaticity, *no* stain, *medium* size, *upright* orientation, *rectangular* shape, *circular* path of motion, *medium* extent of motion, and *medium* period of motion. The other twenty-five Gestures were defined by setting exactly one of these properties to one of its alternate values.

### 3.1.2 Results

Six subjects ran through between five and twelve trial sets, out of a pool of twenty-six, where each trial set consisted of sixty individual trials. Originally, I scheduled twelve trial sets for each subject, but due to technical problems and subject availability, some of the sets were not run. For this reason, I threw out all but the later sets from each subject that completed more than six sets, leaving no more than six sets of data. Only one subject had five sets. The sets remaining form a pool of twenty-one distinct sets, from which pool fourteen were seen by exactly two subjects.

Figures 3.2-3.5 together form a table with the composite results from the thirty-five trials set data. The letters A through Y correspond to the twenty-five non-control Gestures as marked by the row labels. The denominator at each location equals the number of times the row Gesture was considered alongside the column Gesture and the control Gesture. The numerator at each location in the grid equals the number of times the row Gesture was chosen by a subject as more prominent than the other two. The difference between the denominator and the sum of the numerators from a transposed pair equals the number of times the control Gesture was chosen over the two row/column Gestures. A box is placed around all fractions greater than one-half.

I use simply the sum of boxes across its row as the measure of relative prominence for a Gesture. Using this crude measure, the Gestures separate into the three arbitrarily delimited tiers shown in Table 3.1.

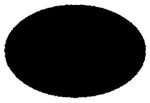
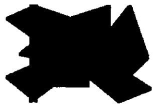
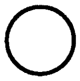
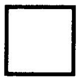
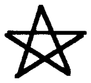
| | | | | | |
|---|---|---|---|---|---|
| texture | pour | icarus | roger | | |
| intensity (normalized) | normal 1.0 | dim 0.7 | bright 2.0 | | |
| chromaticity (normalized) | rich 1.0 | grey 0.0 | pale 0.3 | saturated 3.0 | |
| stain (normalized) | no 1.0 1.0 1.0 | red 2.0 1.0 1.0 | green 1.0 2.0 1.0 | blue 1.0 1.0 2.0 | |
| size (pixels) | medium 90 60 | small 60 40 | large 180 120 | wide 180 60 | tall 90 120 |
| orientation (degrees) | upright 0° | slanted 30° | upside-down 180° | | |
| shape | rectangular | oval  | jagged  | | |
| path of motion | circular  | square  | star  | | |
| extent of motion (pixels) | medium 50 | tight 25 | loose 100 | fixed 0 | |
| period of motion (seconds) | medium 5.0 | slow 7.5 | fast 2.5 | | |

Figure 3.1: Gesture property values for experiment 1.

| | | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| texture | | | | | | | | | | | | | |
| icarus | A | | 0/3 | $\boxed{4/5}$ | $\boxed{13/14}$ | $\boxed{6/8}$ | $\boxed{6/6}$ | $\boxed{2/3}$ | 1/4 | 2/5 | $\boxed{3/4}$ | $\boxed{5/6}$ | $\boxed{13/14}$ |
| roger | B | $\boxed{3/3}$ | | $\boxed{6/8}$ | $\boxed{7/7}$ | $\boxed{7/9}$ | $\boxed{6/7}$ | $\boxed{10/10}$ | $\boxed{5/5}$ | $\boxed{7/9}$ | $\boxed{4/5}$ | $\boxed{5/5}$ | $\boxed{8/8}$ |
| intensity | | | | | | | | | | | | | |
| bright | C | 1/5 | 2/8 | | $\boxed{7/7}$ | $\boxed{4/7}$ | $\boxed{5/7}$ | $\boxed{5/8}$ | 1/3 | 7/14 | 5/11 | $\boxed{2/3}$ | $\boxed{5/7}$ |
| dim | D | 0/14 | 0/7 | 0/7 | | 0/2 | 1/10 | 0/3 | 0/6 | 0/1 | 0/8 | 1/12 | 0/6 |
| chromaticity | | | | | | | | | | | | | |
| saturated | E | 1/8 | 2/9 | 3/7 | $\boxed{2/2}$ | | $\boxed{6/8}$ | $\boxed{7/8}$ | $\boxed{4/6}$ | 3/6 | 0/2 | $\boxed{10/10}$ | $\boxed{7/9}$ |
| pale | F | 0/6 | 1/7 | 2/7 | 5/10 | 0/8 | | $\boxed{1/1}$ | 2/15 | 0/6 | 1/5 | 1/9 | 2/8 |
| grey | G | 0/3 | 0/10 | 2/8 | $\boxed{2/3}$ | 1/8 | 0/1 | | 1/9 | 0/12 | 0/1 | $\boxed{7/11}$ | 2/6 |
| stain | | | | | | | | | | | | | |
| red | H | $\boxed{3/4}$ | 0/5 | $\boxed{2/3}$ | $\boxed{5/6}$ | 1/6 | $\boxed{13/15}$ | $\boxed{5/9}$ | | $\boxed{1/1}$ | 3/6 | $\boxed{4/6}$ | $\boxed{3/3}$ |
| green | I | $\boxed{3/5}$ | 2/9 | 7/14 | $\boxed{1/1}$ | 3/6 | $\boxed{6/6}$ | $\boxed{11/12}$ | 0/1 | | 4/11 | $\boxed{2/3}$ | $\boxed{4/5}$ |
| blue | J | 1/4 | 1/5 | $\boxed{6/11}$ | $\boxed{7/8}$ | 0/2 | $\boxed{4/5}$ | 0/1 | 2/6 | 5/11 | | $\boxed{6/7}$ | $\boxed{9/10}$ |
| shape | | | | | | | | | | | | | |
| oval | K | 1/6 | 0/5 | 1/3 | $\boxed{8/12}$ | 0/10 | $\boxed{5/9}$ | 4/11 | 1/6 | 1/3 | 1/7 | | 4/10 |
| jagged | L | 0/14 | 0/8 | 2/7 | $\boxed{4/6}$ | 1/9 | $\boxed{5/8}$ | 3/6 | 0/3 | 1/5 | 1/10 | 4/10 | |

Figure 3.2: Prominence results (part 1).

|  |  | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| orientation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| slanted | M | $\boxed{3/4}$ | 2/5 | 2/5 | $\boxed{7/7}$ | 3/9 | $\boxed{6/7}$ | $\boxed{6/9}$ | $\boxed{2/3}$ | 4/9 | 1/6 | 0/2 | 3/6 |
| upside-down | N | 0/4 | 0/5 | 0/4 | $\boxed{3/5}$ | 0/5 | 3/6 | 0/4 | 0/3 | 1/8 | 0/5 | 6/17 | 4/13 |
| size |  |  |  |  |  |  |  |  |  |  |  |  |  |
| large | O | 1/10 | 1/15 | 1/8 | $\boxed{6/7}$ | 3/9 | 0/3 | 1/5 | 0/6 | 1/7 | 0/7 | 4/9 | 4/9 |
| small | P | 1/9 | 0/4 | 0/3 | $\boxed{3/5}$ | 0/5 | $\boxed{3/4}$ | 1/3 | 0/0 | 1/7 | 0/4 | 3/11 | 2/9 |
| tall | Q | 0/13 | 0/5 | 0/5 | $\boxed{9/12}$ | 0/4 | 2/5 | 1/5 | 0/4 | 1/2 | 1/10 | 2/4 | 3/8 |
| wide | R | 1/11 | 0/2 | 2/8 | $\boxed{5/6}$ | 2/13 | 1/4 | 1/4 | 3/8 | 1/8 | 0/8 | $\boxed{3/4}$ | $\boxed{4/6}$ |
| path of motion |  |  |  |  |  |  |  |  |  |  |  |  |  |
| star | S | 2/9 | 1/12 | 0/7 | $\boxed{7/11}$ | 2/7 | 1/7 | 2/8 | 0/12 | 1/9 | 0/2 | 1/6 | 1/10 |
| square | T | 2/7 | 0/12 | 0/6 | 2/6 | 0/6 | 1/4 | 0/5 | 0/7 | 0/4 | 1/7 | 1/10 | 0/2 |
| fixed | U | 0/5 | 0/5 | 0/7 | 3/6 | 0/1 | 1/5 | 2/8 | 1/6 | 1/5 | 0/5 | 0/5 | 1/9 |
| extent of motion |  |  |  |  |  |  |  |  |  |  |  |  |  |
| loose | V | 0/8 | 0/2 | 0/7 | $\boxed{5/8}$ | 0/3 | 0/8 | $\boxed{5/8}$ | 2/15 | 1/4 | 2/11 | 3/7 | 4/14 |
| tight | W | 0/5 | 0/7 | 0/7 | 1/6 | 0/3 | 2/8 | 1/7 | 0/12 | 0/15 | 1/12 | 1/13 | 0/6 |
| period of motion |  |  |  |  |  |  |  |  |  |  |  |  |  |
| long | X | 0/4 | 0/11 | 0/5 | $\boxed{3/5}$ | 0/6 | 2/6 | 1/5 | 0/7 | 0/6 | 1/4 | 1/5 | 0/9 |
| short | Y | 1/10 | 0/9 | 0/3 | 3/6 | 3/10 | 1/5 | $\boxed{3/4}$ | 1/8 | 0/3 | 2/8 | 2/7 | 2/7 |

Figure 3.3: Prominence results (part 2).

| | | M | N | O | P | Q | R | S | T | U | V | W | X | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| texture | | | | | | | | | | | | | | |
| icarus | A | 1/4 | $\boxed{3/4}$ | $\boxed{9/10}$ | $\boxed{8/9}$ | $\boxed{13/13}$ | $\boxed{10/11}$ | $\boxed{6/9}$ | $\boxed{5/7}$ | $\boxed{5/5}$ | $\boxed{8/8}$ | $\boxed{5/5}$ | $\boxed{4/4}$ | $\boxed{9/10}$ |
| roger | B | $\boxed{3/5}$ | $\boxed{5/5}$ | $\boxed{13/15}$ | $\boxed{4/4}$ | $\boxed{5/5}$ | $\boxed{2/2}$ | $\boxed{11/12}$ | $\boxed{12/12}$ | $\boxed{5/5}$ | $\boxed{2/2}$ | $\boxed{7/7}$ | $\boxed{11/11}$ | $\boxed{9/9}$ |
| intensity | | | | | | | | | | | | | | |
| bright | C | $\boxed{3/5}$ | $\boxed{4/4}$ | $\boxed{7/8}$ | $\boxed{2/3}$ | $\boxed{5/5}$ | $\boxed{6/8}$ | $\boxed{7/7}$ | $\boxed{6/6}$ | $\boxed{7/7}$ | $\boxed{7/7}$ | $\boxed{7/7}$ | $\boxed{5/5}$ | $\boxed{3/3}$ |
| dim | D | 0/7 | 2/5 | 0/7 | 1/5 | 2/12 | 0/6 | 2/11 | 1/6 | 2/6 | 2/8 | 1/6 | 1/5 | 3/6 |
| chromaticity | | | | | | | | | | | | | | |
| saturated | E | $\boxed{6/9}$ | $\boxed{5/5}$ | $\boxed{6/9}$ | $\boxed{5/5}$ | $\boxed{4/4}$ | $\boxed{10/13}$ | $\boxed{5/7}$ | $\boxed{5/6}$ | $\boxed{1/1}$ | $\boxed{3/3}$ | $\boxed{3/3}$ | $\boxed{6/6}$ | $\boxed{7/10}$ |
| pale | F | 1/7 | 2/6 | $\boxed{2/3}$ | 0/4 | 2/5 | $\boxed{3/4}$ | $\boxed{6/7}$ | 1/4 | $\boxed{3/5}$ | $\boxed{7/8}$ | 4/8 | 3/6 | $\boxed{4/5}$ |
| grey | G | 1/9 | $\boxed{3/4}$ | $\boxed{4/5}$ | $\boxed{2/3}$ | $\boxed{4/5}$ | $\boxed{3/4}$ | $\boxed{5/8}$ | $\boxed{5/5}$ | $\boxed{5/8}$ | 3/8 | $\boxed{5/7}$ | $\boxed{3/5}$ | 1/4 |
| stain | | | | | | | | | | | | | | |
| red | H | 1/3 | $\boxed{3/3}$ | $\boxed{5/6}$ | 0/0 | $\boxed{4/4}$ | $\boxed{5/8}$ | $\boxed{11/12}$ | $\boxed{7/7}$ | $\boxed{5/6}$ | $\boxed{10/15}$ | $\boxed{11/12}$ | $\boxed{7/7}$ | $\boxed{5/8}$ |
| green | I | $\boxed{5/9}$ | $\boxed{7/8}$ | $\boxed{6/7}$ | $\boxed{5/7}$ | 1/2 | $\boxed{7/8}$ | $\boxed{8/9}$ | $\boxed{4/4}$ | $\boxed{4/5}$ | $\boxed{3/4}$ | $\boxed{14/15}$ | $\boxed{6/6}$ | $\boxed{3/3}$ |
| blue | J | $\boxed{5/6}$ | $\boxed{5/5}$ | $\boxed{7/7}$ | $\boxed{3/4}$ | $\boxed{8/10}$ | $\boxed{7/8}$ | 1/2 | $\boxed{6/7}$ | $\boxed{4/5}$ | $\boxed{8/11}$ | $\boxed{10/12}$ | $\boxed{3/4}$ | $\boxed{6/8}$ |
| shape | | | | | | | | | | | | | | |
| oval | K | $\boxed{2/2}$ | $\boxed{11/17}$ | $\boxed{5/9}$ | 4/11 | 2/4 | 1/4 | $\boxed{4/6}$ | $\boxed{7/10}$ | $\boxed{3/5}$ | $\boxed{4/7}$ | $\boxed{11/13}$ | $\boxed{3/5}$ | $\boxed{5/7}$ |
| jagged | L | 2/6 | $\boxed{8/13}$ | 4/9 | $\boxed{6/9}$ | 4/8 | 1/6 | $\boxed{7/10}$ | $\boxed{2/2}$ | $\boxed{7/9}$ | $\boxed{8/14}$ | $\boxed{6/6}$ | $\boxed{7/9}$ | $\boxed{5/7}$ |

Figure 3.4: Prominence results (part 3).

| | | M | N | O | P | Q | R | S | T | U | V | W | X | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| orientation | | | | | | | | | | | | | | |
| slanted | M | | $\boxed{8/10}$ | $\boxed{1/1}$ | $\boxed{1/1}$ | $\boxed{7/10}$ | $\boxed{2/3}$ | $\boxed{7/8}$ | $\boxed{4/5}$ | 0/0 | $\boxed{8/10}$ | $\boxed{4/4}$ | $\boxed{2/2}$ | $\boxed{6/8}$ |
| upside-down | N | 1/10 | | 1/6 | 3/8 | 2/8 | 1/5 | 4/10 | $\boxed{4/6}$ | 0/0 | 5/11 | $\boxed{10/10}$ | $\boxed{13/17}$ | $\boxed{5/7}$ |
| size | | | | | | | | | | | | | | |
| large | O | 0/1 | $\boxed{4/6}$ | | $\boxed{7/9}$ | 3/8 | $\boxed{3/5}$ | $\boxed{11/16}$ | $\boxed{4/4}$ | $\boxed{8/9}$ | 7/14 | $\boxed{9/10}$ | $\boxed{3/4}$ | $\boxed{6/9}$ |
| small | P | 0/1 | 3/8 | 2/9 | | 1/4 | $\boxed{4/7}$ | 4/9 | $\boxed{4/4}$ | $\boxed{3/5}$ | $\boxed{4/7}$ | 0/1 | $\boxed{8/10}$ | 5/10 |
| tall | Q | 3/10 | 3/8 | 3/8 | $\boxed{3/4}$ | | 0/5 | $\boxed{8/10}$ | $\boxed{11/13}$ | $\boxed{9/15}$ | $\boxed{6/10}$ | $\boxed{9/11}$ | $\boxed{6/6}$ | $\boxed{7/10}$ |
| wide | R | 1/3 | $\boxed{4/5}$ | 2/5 | 2/7 | $\boxed{3/5}$ | | $\boxed{5/7}$ | $\boxed{7/11}$ | $\boxed{7/11}$ | $\boxed{10/12}$ | 3/6 | $\boxed{6/8}$ | $\boxed{4/5}$ |
| path of motion | | | | | | | | | | | | | | |
| star | S | 0/8 | 2/10 | 5/16 | $\boxed{5/9}$ | 1/10 | 1/7 | | 2/8 | 2/7 | 4/12 | $\boxed{5/6}$ | $\boxed{2/2}$ | 4/11 |
| square | T | 0/5 | 1/6 | 0/4 | 0/4 | 1/13 | 4/11 | 2/8 | | 0/2 | 4/13 | 0/3 | 3/10 | 3/8 |
| fixed | U | 0/0 | 0/0 | 1/9 | 1/5 | 5/15 | 3/11 | 2/7 | 1/2 | | 2/11 | 1/2 | 4/10 | 1/10 |
| extent of motion | | | | | | | | | | | | | | |
| loose | V | 2/10 | $\boxed{6/11}$ | 7/14 | 3/7 | 3/10 | 2/12 | 5/12 | $\boxed{7/13}$ | $\boxed{8/11}$ | | $\boxed{4/6}$ | 2/5 | 3/8 |
| tight | W | 0/4 | 0/10 | 0/10 | $\boxed{1/1}$ | 1/11 | 2/6 | 1/6 | 1/3 | 0/2 | 2/6 | | 2/10 | 0/3 |
| period of motion | | | | | | | | | | | | | | |
| long | X | 0/2 | 0/17 | 0/4 | 1/10 | 0/6 | 0/8 | 0/2 | 2/10 | 4/10 | 2/5 | 4/10 | | 1/3 |
| short | Y | 2/8 | 2/7 | 2/9 | 3/10 | 3/10 | 0/5 | 4/11 | 4/8 | $\boxed{7/10}$ | 4/8 | $\boxed{3/3}$ | 1/3 | |

Figure 3.5: Prominence results (part 4).

### 3.1.3 Discussion

The body of data for this experiment is not large enough to allow a finer grading of the Gestures than these three tiers. Testing more subjects with more trials would fix a ranking for these particular Gestures, but that is not the point of this experiment. Rather, I use this experiment to provide a first approximation feel for the visual prominence of the various properties. To do this, I have isolated the individual properties. However, a full experiment must consider the interaction of the properties. For example, how does shape interfere with the prominence of orientation? Also, this experiment purposely avoids dynamism aside from positional motion.

Nevertheless, from this limited set of data I draw the following conclusions: (1) the facial properties contribute the most to visual prominence, (2) the edge properties contribute the second most to visual prominence, and (3) the secondary properties of position contribute the least to visual prominence.

The design of this experiment subtracts the effect of offset position from the results. Indeed, the arrangement of a pair at the top and one below in the center was chosen so that no Gesture was at a summit. While designing the experiment, I realized that the summit is a particularly attractive location. Also, it is reasonable to assume that any lone Gesture apart from a cluster of Gestures is compelling. For these reasons, I conclude that position can contribute strongly to visual prominence, too.

## 3.2 Experiment 2: Peripheral Distraction

Experiments studying the distraction caused by the dynamic visual properties of Gestures can contribute to two areas. First, understanding the effects of distraction can be used to help develop schemes which normally avoid these causes. In our case avoiding undue distraction allows the user to properly concentrate on the demands of the foreground task. Second, a measure will allow degrees of distraction to be accorded to messages that have corresponding degrees of importance. This experiment contributes to the second area. A more specific, longer-term experiment is needed to evaluate the effect of different dynamic properties on the user, in terms of stress and broken concentration, in different circumstances.

This experiment compares untutored, immediate impressions of distraction for a number of different dynamic Gesture behaviors.

### 3.2.1 Method

A subject is asked to read a chapter of text on one screen while grading a series of Gestures presented, one at a time, off to the side on a different screen. The subject grades each Gesture on its distraction using whatever criteria feels natural. The subject then enters the grade by pressing one of the 'j', 'k', or 'l', respectively corresponding to 'little distracting', 'somewhat distracting', and 'very distracting'. The subject can also scroll the text twelve line by hitting the space bar.

The first Gesture is presented eight seconds after the text appears. From then until the user finishes reading the text, a continuous stream of Gestures are presented, each cycling for eight seconds. The offset position of each Gesture is varied so that no two consecutive Gestures have the same offset position. The vertical position is alternately either 200 or 280, and the horizontal position is randomly either 200, 320, or 440. Thirty-three different Gestures were used in this experiment.

Figure 3.6 shows the Gesture property value ranges used for this experiment.

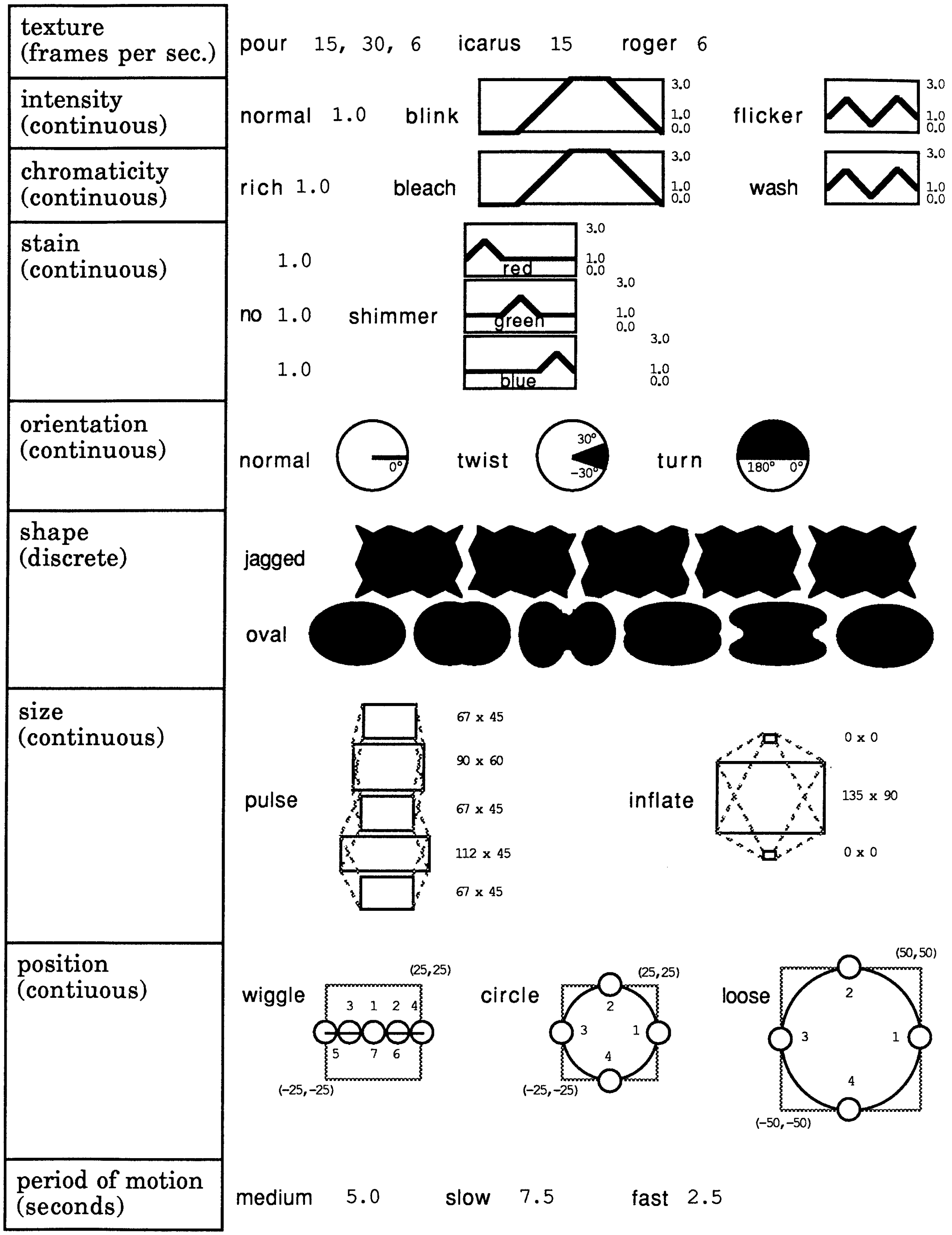


Figure 3.6: Gesture property values for experiment 2.

### 3.2.2 Results

The physical dimensions of the active text screen is approximately 9.5 inches wide and 7.5 inches high, and it is 80 characters wide and 24 characters high. The physical dimensions of the active graphics screen is approximately 10.5 inches wide and 7.5 inches high, and its center is 15 inches to the right and 2 inches down from the center of the text screen. Also, the graphics screen is angled approximately 30 degrees toward subject from the plane of the text screen. The subject was allowed to find a comfortable position from between 2 and 3 feet out, facing the text screen directly.

Four subjects graded Gestures while reading a 380 line chapter of text. Two of the subjects were male, two female. The particular chapter used was Part 1, Chapter 6 of Fyodor Dostoyevsky's *The Idiot.* I chose this text because it is one long rambling quote. It seem to me that the character's story was engaging, but required some concentration to follow the stop, starts, and digressions. The reading time, averaged across the four subjects, was approximately 16 minutes.

Figures 3.7 and 3.8 together form a table with the grades of the different Gestures given by the four subjects in different circumstances. The rows indicate the Gesture, and the columns indicate the subject and the circumstance. The 'near', 'mid', and 'far' columns correspond to the three possible horizontal offset positions. The 'ALL' columns sum the other three columns together The triple digit table entries present the composite grades. The left digit equals the number of times 'little distracting' was entered for that Gesture in that circumstance by that subject. Similarly, the middle and right digits correspond to votes for 'somewhat distracting' and 'very distracting', respectively. In the 'ALL' columns, boxed entries are dominated by 'very distracting' votes, and underlined entries are dominated by 'somewhat distracting' votes. Tied votes were rounded down.

| | Subject A | | | | Subject B | | | | Subject C | | | | Subject D | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | near | mid | far | ALL | near | mid | far | ALL | near | mid | far | ALL | near | mid | far | ALL | GRADE |
| **texture** | | | | | | | | | | | | | | | | | |
| pour (15 f.p.s.) | 100 | 000 | 100 | 200 | 100 | 000 | 301 | 401 | 000 | 000 | 000 | | 000 | 100 | 000 | 100 | • |
| pour (30 f.p.s.) | 000 | 200 | 210 | 410 | 000 | 110 | 020 | $\underline{130}$ | 000 | 000 | 000 | | 000 | 401 | 200 | 601 | • |
| pour (6 f.p.s.) | 100 | 010 | 000 | 110 | 000 | 000 | 000 | | 010 | 000 | 100 | 110 | 100 | 300 | 200 | 600 | • |
| icarus (15 f.p.s.) | 000 | 000 | 000 | | 300 | 200 | 000 | 500 | 100 | 001 | 100 | $\underline{201}$ | 200 | 200 | 100 | 500 | • |
| roger (6 f.p.s.) | 010 | 001 | 001 | $\boxed{012}$ | 001 | 000 | 000 | $\boxed{001}$ | 000 | 000 | 010 | $\underline{010}$ | 000 | 010 | 100 | 110 | ● |
| **intensity** | | | | | | | | | | | | | | | | | |
| blink | 000 | 000 | 000 | | 000 | 000 | 001 | $\boxed{001}$ | 001 | 000 | 000 | $\boxed{001}$ | 000 | 000 | 001 | $\boxed{001}$ | ● |
| slow blink | 000 | 000 | 000 | | 000 | 010 | 100 | 110 | 001 | 010 | 000 | $\underline{011}$ | 010 | 000 | 000 | $\underline{010}$ | ● |
| flicker | 012 | 001 | 000 | $\boxed{013}$ | 002 | 002 | 001 | $\boxed{005}$ | 000 | 012 | 011 | $\boxed{023}$ | 000 | 020 | 001 | $\underline{021}$ | ● |
| fast flicker | 001 | 010 | 000 | $\underline{011}$ | 002 | 001 | 000 | $\boxed{003}$ | 012 | 000 | 001 | $\boxed{013}$ | 001 | 002 | 010 | $\boxed{013}$ | ● |
| **chromaticity** | | | | | | | | | | | | | | | | | |
| bleach | 020 | 000 | 120 | $\underline{140}$ | 110 | 010 | 000 | $\underline{120}$ | 110 | 000 | 200 | 310 | 000 | 000 | 200 | 200 | • |
| slow bleach | 010 | 000 | 100 | 110 | 000 | 010 | 000 | $\underline{010}$ | 000 | 100 | 100 | 200 | 110 | 000 | 000 | 110 | • |
| wash | 100 | 100 | 200 | 400 | 100 | 210 | 000 | 310 | 000 | 200 | 000 | 200 | 100 | 000 | 000 | 100 | • |
| fast wash | 110 | 000 | 000 | 110 | 010 | 010 | 000 | $\underline{020}$ | 100 | 100 | 010 | 210 | 100 | 100 | 200 | 400 | • |
| **stain** | | | | | | | | | | | | | | | | | |
| shimmer | 020 | 100 | 000 | $\underline{120}$ | 001 | 010 | 100 | $\underline{111}$ | 100 | 100 | 010 | 210 | 000 | 000 | 000 | | ● |
| slow shimmer | 000 | 010 | 101 | $\underline{111}$ | 102 | 300 | 001 | $\underline{403}$ | 200 | 200 | 100 | 500 | 010 | 000 | 100 | 110 | • |
| fast shimmer | 010 | 100 | 010 | $\underline{120}$ | 110 | 101 | 100 | $\underline{311}$ | 100 | 100 | 000 | 200 | 000 | 000 | 000 | | • |

Figure 3.7: Distraction results for facial properties.

| | Subject A | | | | Subject B | | | | Subject C | | | | Subject D | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | near | mid | far | ALL | near | mid | far | ALL | near | mid | far | ALL | near | mid | far | ALL | GRADE |
| **shape** | | | | | | | | | | | | | | | | | |
| jagged series | 000 | 000 | 300 | 300 | 011 | 010 | 100 | $\underline{121}$ | 000 | 000 | 000 | | 100 | 100 | 000 | 200 | • |
| oval series | 100 | 000 | 110 | 210 | 001 | 000 | 000 | $\boxed{001}$ | 000 | 000 | 000 | | 000 | 110 | 100 | 210 | • |
| **orientation** | | | | | | | | | | | | | | | | | |
| turn | 000 | 001 | 011 | $\boxed{012}$ | 032 | 001 | 000 | $\underline{033}$ | 010 | 010 | 010 | $\underline{030}$ | 010 | 000 | 000 | $\underline{010}$ | ● |
| slow turn | 004 | 000 | 001 | $\boxed{005}$ | 002 | 010 | 000 | $\boxed{012}$ | 001 | 000 | 000 | $\boxed{001}$ | 010 | 010 | 000 | $\underline{020}$ | ⬤ |
| twist | 000 | 010 | 020 | $\underline{030}$ | 010 | 001 | 000 | $\underline{011}$ | 030 | 010 | 010 | $\underline{050}$ | 010 | 010 | 200 | 220 | ● |
| fast twist | 000 | 011 | 000 | $\underline{011}$ | 001 | 002 | 000 | $\boxed{003}$ | 010 | 020 | 010 | $\underline{040}$ | 020 | 030 | 000 | $\underline{050}$ | ● |
| **size** | | | | | | | | | | | | | | | | | |
| inflate | 000 | 000 | 000 | | 000 | 000 | 000 | | 010 | 003 | 021 | $\boxed{034}$ | 002 | 000 | 010 | $\boxed{012}$ | ⬤ |
| slow inflate | 000 | 000 | 000 | | 000 | 101 | 000 | $\underline{101}$ | 000 | 010 | 022 | $\underline{032}$ | 013 | 001 | 000 | $\boxed{014}$ | ● |
| pulse | 020 | 000 | 010 | $\underline{030}$ | 000 | 010 | 000 | $\underline{010}$ | 010 | 010 | 011 | $\underline{031}$ | 020 | 110 | 000 | $\underline{130}$ | ● |
| fast pulse | 001 | 001 | 000 | $\boxed{002}$ | 001 | 000 | 003 | $\boxed{004}$ | 012 | 003 | 010 | $\boxed{025}$ | 000 | 010 | 000 | $\underline{010}$ | ⬤ |
| **position** | | | | | | | | | | | | | | | | | |
| circle | 001 | 002 | 003 | $\boxed{006}$ | 010 | 000 | 000 | $\underline{010}$ | 002 | 010 | 000 | $\boxed{012}$ | 010 | 000 | 001 | $\underline{011}$ | ⬤ |
| fast circle | 000 | 002 | 000 | $\boxed{002}$ | 000 | 002 | 010 | $\boxed{012}$ | 001 | 002 | 000 | $\boxed{003}$ | 010 | 001 | 011 | $\underline{022}$ | ⬤ |
| loose circle | 000 | 000 | 000 | | 002 | 010 | 020 | $\underline{032}$ | 001 | 011 | 010 | $\underline{022}$ | 011 | 003 | 010 | $\boxed{024}$ | ● |
| slow loose circle | 001 | 000 | 000 | $\boxed{001}$ | 101 | 000 | 000 | $\underline{101}$ | 021 | 010 | 000 | $\underline{031}$ | 000 | 000 | 030 | $\underline{030}$ | ● |
| wiggle | 001 | 001 | 010 | $\boxed{012}$ | 000 | 000 | 001 | $\boxed{001}$ | 010 | 000 | 020 | $\underline{030}$ | 011 | 011 | 000 | $\underline{022}$ | ⬤ |
| fast wiggle | 002 | 011 | 001 | $\boxed{014}$ | 001 | 002 | 000 | $\boxed{003}$ | 000 | 001 | 002 | $\boxed{003}$ | 001 | 020 | 010 | $\underline{031}$ | ⬤ |
| slow wiggle | 000 | 000 | 020 | $\underline{020}$ | 000 | 000 | 000 | | 010 | 020 | 020 | $\underline{050}$ | 010 | 010 | 100 | $\underline{120}$ | ● |

Figure 3.8: Distraction results for silhouette and position properties.

I roughly average the results across the four subjects to determine the level of distraction for a Gesture. Table 3.2 shows how I split the three tiers.

### 3.2.3 Discussion

Once again, this body of data is not big enough to assign fine rankings. Using more subjects would only fix a ranking for these particular Gestures, rather than for the properties themselves. A much larger set of Gestures would be needed to test different secondary property values for the various primary properties and the interaction caused by dynamically changing more than one property at a time.

Changes in intensity, size, and position appear to be the most potentially distracting, while chromaticity and shape appear to be the least potentially distracting. The speed of change appears to potentially contribute a lot toward distraction. The extent of change appears to be somewhat strong, but its effect can be over slow changes. Not enough paths of change were used to allow a good analysis, but it seems that they would contribute to distraction mainly by providing local bursts of speed over sizable extents. The offset value or static component modifies the effect of the other secondary properties. For example, motions closer to the center are a little more distracting than motions farther away. Also, although this was not tested, intensity changes near black are probably less distracting than intensity changes near white.

I rank texture with a medium strength, but since the colors of a texture are not constrained, changes in texture could easily cause changes in intensity. In other words, changes of texture are potentially strong for distraction, but when using natural movie sequences, the changes need not be too distracting. The *roger* texture sequence used in this experiment is really a slide show of brightly colored frames, so it was fairly distracting. The *pour* sequence is a natural movie segment of a boy approaching a water faucet and pouring from it, and it was not as distracting.

| Upper Tier | | Middle Tier | | Lower Tier | |
|---|---|---|---|---|---|
| Gesture | # of boxes | Gesture | # of boxes | Gesture | # of boxes |
| *roger* | 24 | *grey* | 12 | *small* | 7 |
| *icarus* | 20 | *oval* | 12 | *pale* | 7 |
| *bright* | 19 | *jagged* | 11 | *loose* | 6 |
| *saturated* | 19 | *wide* | 11 | *upside-down* | 5 |
| *red* | 19 | *large* | 10 | *star* | 4 |
| *green* | 18 | *tall* | 9 | *short* | 3 |
| *blue* | 17 | | | *long* | 1 |
| *slanted* | 16 | | | *tight* | 1 |
| | | | | *fixed* | 0 |
| | | | | *square* | 0 |
| | | | | *dim* | 0 |

Table 3.1: The three tiers of Gesture prominence.

| Upper Tier | Middle Tier | Lower Tier |
|---|---|---|
| Gesture | Gesture | Gesture |
| *blink* | *roger* | *pour* (15 f.p.s.) |
| *flicker* | *slow blink* | *pour* (30 f.p.s.) |
| *fast flicker* | *shimmer* | *pour* (6 f.p.s.) |
| *slow turn* | *turn* | *icarus* |
| *inflate* | *twist* | *bleach* |
| *fast pulse* | *fast twist* | *slow bleach* |
| *circle* | *slow inflate* | *wash* |
| *fast circle* | *pulse* | *fast wash* |
| *wiggle* | *loose circle* | *slow shimmer* |
| *fast wiggle* | *slow loose circle* | *fast shimmer* |
| | *slow wiggle* | *jagged series* |
| | | *oval series* |

Table 3.2: The three tiers of Gesture distraction.

# Chapter 4

# Signs of the Computer Workstation

There are many current examples of indirect computer-to-user communication. To establish a background, this chapter categorizes some of these examples using two theories of signs. A sign is a unit of communication where a receiver perceives the physical expression of a sign and associates it with a particular content. The first system is from the definition of "six species of sign" given by Thomas A. Sebeok in *Contributions to the Doctrine of Signs* (1976). The second system is the "typology of modes of sign production" proposed by Umberto Eco in *A Theory of Semiotics* (1976).

While both Sebeok and Eco provides general definitions using wide ranges of examples, I will only consider the limited case of signs from visual displays here. Also, I will always assume that the user is present and attentive when the sign is displayed.

## 4.1 Six Species of Sign

The six species that Sebeok covers are *the signal*, *the symptom*, *the icon*, *the index*, *the symbol*, and *the name*. Sebeok chose these six because they "seem to occur most frequently in contemporary semiotics" (page 120).

As Sebeok explains, these categories really classify aspects of signs and not the signs themselves. A particular sign may have aspects that fall under two or more specific categories. For example, many common symbols are somewhat iconic: a stylized picture of a bicycle is both a symbol and an icon. Usually, one aspect of a sign dominates its character enough to suggest one classification over another. In this example, the bicycle symbol, while only iconic of a generic, unmotored, two-wheeled cycle, symbolizes all sizes and types of unmotored cycles, including unicylces and tricycles.

### 4.1.1 The Signal

A signal is a binary sign that is either perceived by the viewer or not. The nature of the expression may be completely divorced from the nature of the content. Neither the intensity of a signal, the duration of a signal, the temporal rate of a signal's presentation, the relation of a signal to other signals presented in parallel nor any other quality of a signal affect the connection of the expression to the content. If these other properties carry meaning, then it through a different type of sign.

A very common visual signal in many computer environments is the command prompt. The prompt signifies completion of the "foreground" task. The signal is especially useful if the user does not know exactly how long the foreground task will execute. To reinforce the idea that a sign may have different aspects, note that a command prompt may be a text string which spells the name of the machine which controls the terminal. In this case, the prompt is both a signal and a name.

Another example of a visual signal is the visual bell. A common form of the visual bell is a momentary inversion of the video signal across a single window or the entire screen. This bell is overloaded with meanings since it is usually the only way for a program or task to asynchronously signal the user visually outside of local text strings. One common meaning of the bell is "I do not understand what you typed." Another is "An unusual or special state has been entered, and you should look to the logical place for more descriptive information."

### 4.1.2 The Symptom

A symptom is a synchronous sign with a causal link between the content and the expression. When the content, such as some measured quantity, changes the expression, a more or less measurable quality, changes accordingly.

A good example of a computer symptom is response time. Response time makes a fairly nonspecific sign: it is hard to localize it in time or enumerate its components. Nevertheless, this sign is often perceived by the user and does carry meaning. Degrees of response time signify degrees of the computer's workload. Quick response time and slow response time are symptoms of unencumbered and encumbered systems, respectively. This information is especially useful when tasks run in the background, away from the user's immediate interaction.

Many types of intentional symptoms are built into programs, especially into programs which are under development. For instance, a program may print a piece of text when it reaches a certain state. The act of printing of this text merely signals the program's attainment of the state, but the frequency of printing symptomatically shows the degree of recurrence of the state. Also, the volume of text may be a symptom of the relative dominance of the particular state to other states.

### 4.1.3 The Icon

The expression of an iconic sign resembles topologically the signified content. While we are restricting expression here to visual media, mappings can be defined between nonvisual contents and visual expression. The displayed waveform of a spoken word is an icon of that word because there is a function to map features of the expression, the waveform, to the content, the utterance. The sub-classes of icons are images, diagrams, and metaphors. These three degrees of representation range from realistic through schematic to abstract, and really define a more or less continuous spectrum.

Obviously, since images and diagrams fall under the icon category, all of image processing and computer graphics have strong iconic components. Not only can static features be iconified, but also movements, sways, pulses, or any dynamic feature that characterizes an object or concept can be hinted or recreated. There is no reason that an icon must be static. For example, a simple dynamic icon is the hour hand on an analog clock face. The hand mimics the sun's, then the moon's, daily journey as viewed from earth. Of course, the iconic aspect of this example is overshadowed by the stronger indexic and symbolic aspects.

Metaphors are frequently used in computer programs to aid usage and development. Most window systems are built around metaphors for the shuffling and pigeon-holing techniques used in organizing physical desks. The whole interface is an icon for traditional desk usage. Even as window systems evolve to incorporate new schemes of presentation, they will almost definitely retain certain parts of this powerful metaphor.

### 4.1.4 The Index

An index is a sign whose expression is spatially or temporally contiguous to the content, or is itself an example of the content. While the icon uses similarity to connect an abstracted or concretized expression to its content, the index uses adjacency to connect the expression to its content at the same level of abstraction.

The most common index found in computer interfaces is the cursor. A text cursor, for instance, indicates where in a text stream the next character typed will be inserted. Similarly, a mouse's cursor is contiguous with the screen point the system holds as the target of the mouse's cumulative displacements and accelerations. In both of these cases, the signified locations are already visible, but they need to be set apart from other possible locations by being indexed.

In addition to continuously pointing to a continuously changing location, an index can persistently point to a location that was of particular interest in the past. A good example of this is found on certain stereo systems which have LED peak level indicators. On some of these displays, the LED which corresponds to the greatest level reached over some sample period stays lit even while the other LEDs follow the fluctuating level. The light of the index LED is contiguous in time to when it was lit to indicate a momentary signal peak. Similarly, in a computer interface where a selected menu item remains highlighted for some period after its selection, the expression of the index, the highlighting, is temporally contiguous with the content, the selecting.

### 4.1.5 The Symbol and the Name

Symbols and names have conventional links between the expressions and classes of content. The connection need not be based on similarity or contiguity. How specific the classes are distinguishes symbols from names. A name is essentially unique per content, and two items share the same name solely because of coincidence. A symbol collects contents that have properties in common, although the particular properties may not be easily enumerated.

A common example of a visual symbol is the so-called 'icon' that represents the file type on file finder programs. For instance, a directory file may be shown as a picture of a physical folder, a text file as a picture of a lined piece of paper, and an executable file as a picture of a video monitor or a computer. When a user scans a directory for a particular file, the pictures help the user classify kinds of files and ignore those kinds that do not match that of the desired file.

Alternatively, the 'icon' of a file may be unique, or essentially unique. In this case, the 'icon' is a name. Names and symbols may overlap somewhat. That is, two versions of a computer program may share the same name in the conventional sense, but that name is really only a symbol that represents either of two items that are very similar, but still distinct. A proper name would distinguish between the two versions.

## 4.2 Modes of Sign Production

The second conception of sign classification to be considered here is the one offered by Umberto Eco in *A Theory of Semiotics* (1976). Eco thinks the common language notion of signs as discrete elements discourages study of the changing interrelations between the various elements in both the expression and the content planes. Rather than viewing a sign as a whole with aspects belonging to different categories as Sebeok does, Eco holds that distinct sign-functions operate on the elements, its content and its expression, of the so-called sign under different circumstances.

> The notion of sign is untenable when confused with those of significant elementary *units* and *fixed* correlations; there are on the contrary 'signs' that result from the correlation of an imprecise expressive texture and convey a vast and unanalyzable portion of content; and there are expressive devices that convey different content according to different contexts, thus demonstrating...that sign-functions are the frequently transitory result of processual and circumstantially based stipulations. (page 216)

Since Eco does not consider the tie between expression and content to be fixed, signs are not viewed primarily in terms of some objective relationship between the two, but rather in terms of the receiver's labor to produce a connecting function. Either the receiver recognizes a logical connection between an expression and a content, the receiver accepts an expression as an example or a sample of a content, the receiver recalls the connection of a replicated expression to a content, or the receiver interprets the connection between an invented expression and a content. Eco titles these four circumstances *recognition*, *ostension*, *replica*, and *invention.*

### 4.2.1 Recognition

> Recognition occurs when a given object or event, produced by nature or human action (intentionally or unintentionally), and existing in a world of facts as a fact among facts, comes to be viewed by an addressee as the expression of a given content, either through a pre-existing and coded correlation or through the positing of a possible correlation by its addressee. (page 221)

Experience often reveals correlations between different events. That is, the probability of an event having happened or happening seems to increase or decrease in conjunction with the happening of a different event. Given enough confidence, a person will take the occurrence of the second event as a sign of the (past, present, or future) occurrence of the first event. The sign function is produced by the receiver recognizing a correlation, regardless of whether other people recognize the same connection or whether objective arguments seem to support or to contradict the connection. However, education from others and logical arguments may guide the receiver toward a specific recognition. Eco lists three types of recognition sign functions: imprints, symptoms, and clues.

Eco's imprint is similar to Sebeok's temporal index. Suppose a person is using a spread sheet without knowing exactly the rules that are used to calculate the various cells. Next, say the person notices that the value of cell A increases in some mutual relation to decreases in the value of cell B, and, generally, the independence of cell A from manipulations of cell C. However, one time while setting the value of C, the person sees the value of A increase. The person may infer that the value of B has decreased without checking. Of course, the value of B might not have decreased, but that is irrelevant with regard to whether the person thinks it has. The content of this sign function is the decreasing of the B value, and the expression is the imprint on the value of A.

As with Sebeok, the expression of a symptom is synchronously linked to changes in the content. Clues are asynchronously linked to changes in the content. That is, while the expression of a clue signifies a noteworthy change in the content, the expression can be perceived by the viewer at any time. Loosely speaking, clues differ from imprints in the same way that symbols differ from names. An imprint signifies a particular content, and a clue signifies a class of contents. Visual glitches are clues for incorrect or incomplete graphical operations, whether or not the specific operations can be identified.

### 4.2.2 Ostension

> Ostension occurs when a given object or event produced by nature or human action (intentionally or unintentionally and existing in a world of facts as a fact among facts) is 'picked up' by someone and *shown* as the expression of the class of which it is a member. (pages 224-225)

The content of ostension is the class as a whole, and the expression is a member or a part of a member. Examples, samples, and fictive samples fall under ostension.

A brief trailer or preview of a video recording is a sample. A tutorial that leads a user through the necessary steps of a particular operation is an example. It is a true example because the user must actually perform the steps, so the operation is actually executed, even if the results are disregarded. An automated tutorial that shows the mouse cursor moving to a menu bar, pulling down the menu, and selecting a particular menu item is a fictive sample. It is fictive because it is staged. It is a sample because a component of the operation, namely the interaction in this example, is not present.

### 4.2.3 Replica

A replica sign function is one that has been used previously to connect an expression with a content. How the original sign function was produced, whether through recognition, ostension, or invention, is irrelevant to how its replica is produced. Given a particular expression that was encountered previously, the receiver tries to associate it with one of the contents with which it was associated before. Of course, if no acceptable content is found, then a new sign function, if any, must be produced.

> ...[O]ne can replicate: (i) features of a given system that must be combined with features of the same system in order to compose a recognizable functive; (ii) features from a weakly structured repertoire, recognizable on the basis of perceptual mechanisms and correlated to their content by a large-scale overcoding, that must not necessarily be combined with other features; and (iii) features of a given system that must be added to a bundle or to a string of features from one or more other systems in order to compose a recognizable functive. (pages 237-238)

These three options correspond to combinational units, stylizations, and vectors, respectively. Combinational units form systems with definite rules of composition. Stylizations combine loosely, if at all. Vectors perform modifying functions, so they must be used with elements from other systems.

The types of systems which combinational units form may operate on any of three levels. The lowest possible level is of *figurae*. *Figurae* are units, like phonemes, which carry no meaning themselves, but from which meaningful units are constructed. The middle level is of lexical units like words. Individual lexical units carry meaning but little or no context. Context is created by the interaction of lexical units. The highest possible level of combinational units is of super-signs. A super-sign is a full statement with sufficient context to be considered a proposition alone.

Stylizations range from iconic symbols to literary or artistic genres. Typical window system 'icons' are stylizations. They are units, but there are no rules for combining them in a way to add meaning. At least this is true currently. The look-and-feel of a window system is also a stylization. Common buttons are put in the same corner of all the window for which they are applicable, and menu trees are traversed in the same manner despite the application.

Vectors are modifying qualities that have no meaning apart from other signs. Direction-specific pointers and intonational variations are examples of vectors. Often, when a spline curve is defined interactively velocity vectors are shown emanating from control points. The direction of the velocity is mimicked by the orientation of the line and the value of the velocity is mimicked by the length. The speed and acceleration of object motion are also vectors if they contribute to the meaning of the object.

To the extent that they involve sign functions, Eco includes programmed stimuli in this group also. The receiver may react to programmed stimuli without actually applying a sign function. However, if the receiver is aware of the stimuli and the reaction, the receiver can either invent a connection between the two or recall the connection. Only under the most crucial circumstances are programmed stimuli even considered in a user interface, such as, perhaps, insider a fighter airplane's cockpit.

Fictive samples are part way included with replicas. Using the previous example of an automated tutorial, the tutorial can be regenerated exactly as it is needed to help recall important points. The second and subsequent viewing of such a tutorial has a different quality than the first viewing. For one, the steps and their order are somewhat anticipated.

### 4.2.4 Invention

> We now have to define a semiotic mode of production in which something is mapped from something else which was not defined and analyzed before the act of mapping took place. We are witnessing a case in which a significant convention is posited at the very moment in which both the functives of the correlation are *invented.* (page 250)

Given an expression that (1) cannot be recognized via experience, (2) cannot be taken as an end in itself, as an example of a class, and (3) cannot be interpreted in a previously defined way, the receiver must invent or hypothesize a content and hypothesize which parts of the expression correspond to the the content. Eco places all sign functions that involve transformations within the domain of inventions. In particular, he lists graphs, projections, and congruences as requiring sign functions of invention, at least initially.

Visual graphs, of course, are very common. Presented with a new graph, the receiver must scan and coordinate the various pieces of parallel information that are presented and choose the salient points. The chosen salient points become the expression of the graph. Graphs involve topological transformations, meaning that, say, non-visual items are expressed visually. Over time a receiver may develop patterns of interpretation, in which case the sign functions are produced through recognition or replica, but the receiver can always choose to disregard these patterns in favor of a newly invented point of view.

Projections are incomplete, non-isomorphic, point-for-point mappings. In other words, specific features found in the expression correspond to a large, but not full, extent with specific features found in the content. A projection is an imperfect rendering. A congruence, on the other hand, is an essentially perfect rendering or a nearly complete, isomorphic, point-for-point mapping. Drawings and images are visual types of projection and congruence. A drawing may be more or less accurate, and images may have greater or lesser fidelity.

## 4.3 The Design of Signs

The types of signs mentioned above provide varying opportunities for the design of intentional, visual signs. Eco makes compelling arguments about the theoretical usefulness of reconfigureable sign functions rather than fixed signs, but for our purposes here we need to consider signs as fixable in space, time, and meaning. As such, the types he provides can still enrich the list given by Sebeok. I will mix and match the two lists of categories.

The remainder of this chapter examines the practical limits of expression, the range and nature of desired contents, and the viability of particular connections between the expressions and the contents.

**Signals and Programmed Stimuli** Almost any feature can be used to signal something. However, for the most impact, a signal should be very strong and unique with respect to its environment, especially if it is infrequent or unexpected. The range of possible contents for a signal is very large, since the expression has no relation to the form of the content. Unfortunately, while a signal may mean different things in different contexts, it is limited to one meaning in a particular context. With few useful means of expression and little opportunity for coding contents, signals must be used sparingly and for very specific purposes.

**Symptoms and Clues** The means of expression for a symptom should be dynamic, allowing it to synchronously follow the changes in the content. Clues are not synchronous, but they too can be expressed by a range. The content of either of these two can be a quantity of some sort. Given the limited number of possible dynamic features, the semantic distribution of these features must be done with care. Only the most relevant quantities can be represented in the various contexts.

**Icons as Transformations, and Examples and Samples** The most likely visual traits of icons, examples, and samples are images, motion, and behaviors. Naturally, the content of examples and samples would involve the same traits. The content of an icon might additionally involve topological transformations. The advantages of using these signs in a given context must be weighed against the restrictions put on these traits for being used for other semantic roles.

**Indices and Imprints** The expression of indices and imprints should be remarkable with respect to their surroundings, but, unlike signals, they do not appear from nowhere unexpected. The content of these would, most effectively, be something subtle or non-descript. The usefulness of an index or an imprint depends on how important the specific content is and how hard it is to identify that specific content normally.

**Symbols and Names as Combinational Units** Combinational units are best expressed by features that can form distinguishable sets. The range of content for symbols and names is unlimited. Any object or idea can be symbolized. Any object or idea can be named. In order to approach this range of meaning, the sets formed from the chosen features are combined in either serial or parallel collections, just as phonemes are used to specify spoken words.

**Vectors** As with symptoms, the expression of vectors vary over a spectrum. The content of vectors is defined in terms of other signs. That is, vectors are modifiers or qualifiers. Vectors can be used, in particular, to heighten the expressiveness of a combinational system. This role is important when the accompanying statements are necessarily brief due to time or complexity.

# Chapter 5

# Can Eloquent Scenery Communicate Effectively?

In this chapter I will evaluate the effectiveness of using Gestures to provide information according to the model provided by Jay Doblin in "A structure for nontextual communications." Specifically, his middle section, titled "The model of message flow," presents a network of message statements, shown here in Figure 5.1, which organizes and focuses his discussion of the properties of effective communication.

Doblin derives his network by listing statements which must be met independently for communication to succeed, rating the interdependence between pairs of these statements, and, finally, graphing the structure as a whole. When laid out, the network's structure features two types of groupings, one circumferential and the other radial. The circumferential groups form the three levels: technical, semantic, and effectiveness. I will consider each of these levels in turn. The radial groups point toward three different goals of communication: to inform, to persuade, and to stimulate.

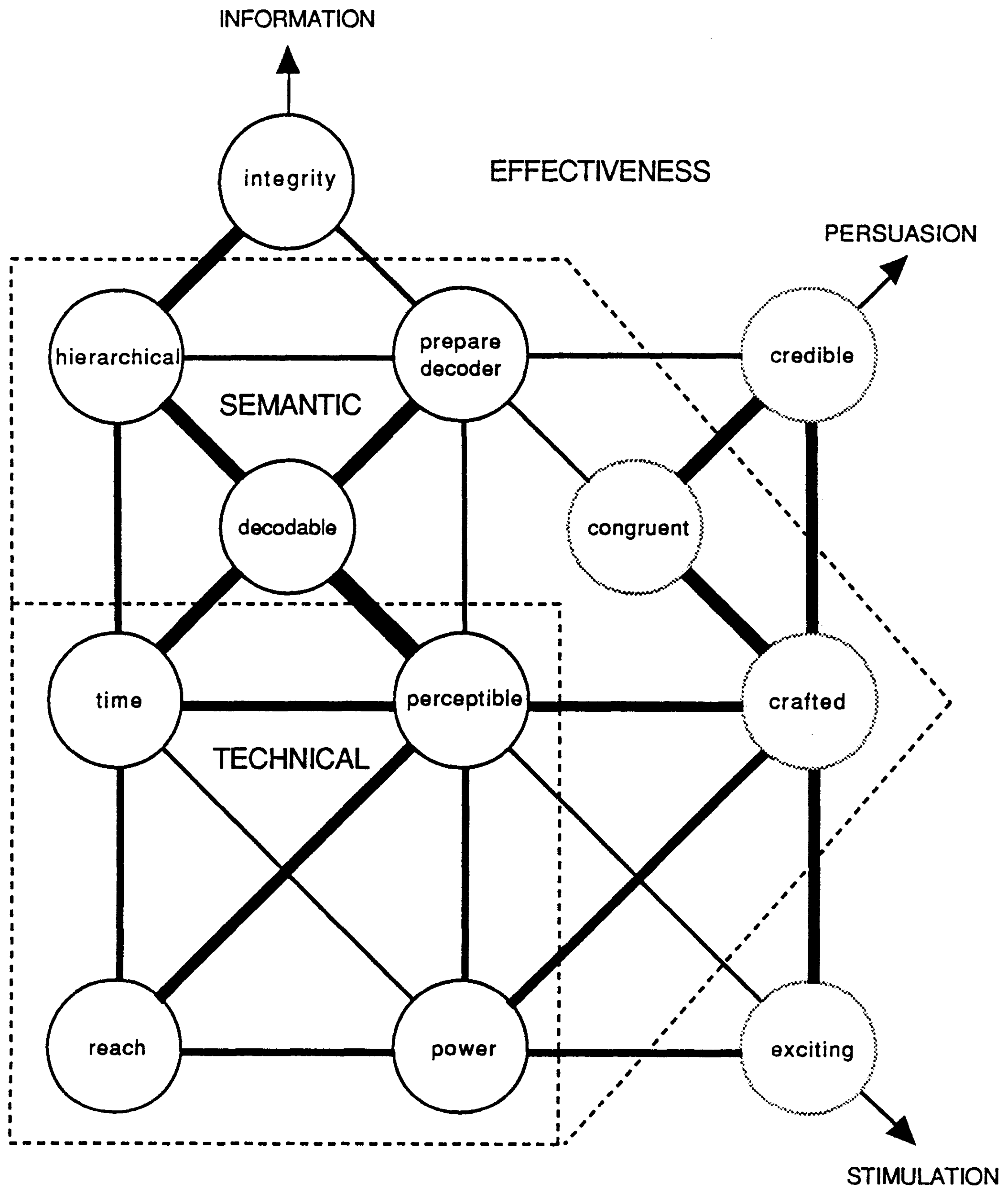


Figure 5.1: Network of network statements

Of these three goals, only the goal of informing is directly important in this paper. However, if the technical layer of Eloquent Scenery is strong enough to provide information, then it has a good foundation for other, similar projects whose tasks are either advertisement or entertainment.

## 5.1 Technical Concerns

Doblin places four message statements within the technical level of his network model. They are "a message must reach the decoders," "a message must be powerful," "a message must have sufficient time," and "a message must be perceptible."

### 5.1.1 A message must reach the decoders

For a message to have a chance to reach the decoder, it must be expressed on a medium available, or potentially available, to the decoder. For a visual message, the expression should be completely or partially in the decoder's normal field of view. A person's field of view is split into a central or foveal region and a peripheral region. Although it is not completely necessary, people tend to center their primary visual task, paying scant conscious attention to sights in the outer field. The visual system does register, however, activity in the periphery, and a person will respond to that activity under certain conditions. For example, many people can navigate a path with steps and turns while reading. The pace of reading may be less than optimal, but the details of the walking chore often do not conflict with the thoughts and ideas generated for and from the reading.

The domain of Eloquent Scenery is just in the peripheral vision. That is, in this project, Gestures will not be used for either, direct communication or conversations. Gestures only need to reach the user's peripheral vision to be 'heard'. With a properly managed peripheral field, the subtlety of Gesture properties can exploit the range of human visual sensitivity.

In typical computer user environments, direct communication and dialogue takes place in windows or terminals. By limiting the user to one visible window at a time, the clutter of a messy 'desktop' can be avoided, allowing the Eloquent Scenery to manage itself and only itself. Gestures reside either in the screen area outside of the window on large, high resolution displays, or on separate displays placed alongside the main one. The system, currently lacking a way to detect the user, assumes the presence of the user. More sophisticated systems could both detect the presence and activity of the immediate user and track a specific user throughout the extent of a communications network.

### 5.1.2 A message must be perceptible

The expression of a message must be perceived for it to be appropriately and correctly decoded. The one exception is the zero message where the absence of expression implies a specific meaning. For instance, if all possible problem conditions are detectable and properly coded into messages, then the absence of messages implies efficient and safe operation. Obviously, in a case like this, a non-zero message should not be so small or so faint that the user fails to notice it and assumes the zero message. Two non-zero messages with different meanings must also be distinguishable from each other.

All of the properties of Gestures have blind spots and points of confusion. Size is especially sensitive because a Gesture that is too small has so few pixels that none of the other properties can be adequately detected. However, since properties are dynamic, a Gesture that momentarily shrinks to nothing is sometimes acceptable. Other properties can be interpolated from when they were last visible.

Care must be taken with other properties as well. The natural change of colors in a texture sequence may incorrectly appear as changes in any of intensity, chromaticity, or stain, or hide actual changes in any of these. Changes of shape can visually overlap changes in size if they amount to a zooming rectangle. If the natural orientation of a texture is not obvious, then the effective range of the orientation property is cut in half or in quarter.

### 5.1.3 A message must be powerful

A message must have enough power to attract the decoder's attention, or, rather, should have an amount of such power appropriate to its importance. A definition of importance is necessary. Importance is both a factor relative to other, essentially concurrent messages and a measure absolute within an environment. For example, of two messages presented roughly in parallel, one may be more awaited by the user, but neither may be truly important, requiring immediate attention. Furthermore, absolute importance cannot be handled with relative distinctions. If two messages are important for external reasons, then they must be heard in turn and not at the expense of each other.

The user's primary task, which is beyond the control and sight of Eloquent Scenery, is considered absolutely important here. Only very important messages should distract the user from that task. Prominence is a different matter. Distractiveness is defined in this project to be a quality of a Gesture that draws visual focus to the object unless it is consciously ignored by the viewer. Prominence is defined to be a quality of a Gesture that heightens the viewer's visual interest once the viewer consciously focuses on the field of Gestures.

The Gestures that occupy Eloquent Scenery have a number of properties, and each of these properties has a static offset and a dynamic adjustment. All of these properties and their components contribute to the overall power of each individual Gesture. For example, intense (bright) colors offer more power than subdued colors, and an object pulsating about some average size has more power than an object remaining at that average size. An initial attempt to investigate the relative strengths of these properties and components of properties was presented in Chapter 3.

### 5.1.4 A message must have sufficient time

A message must be available for at least the amount of time it takes the decoder to receive it. Otherwise, the message is lost. The decoder may need additional time to actually decode the message, working on intermediate representations, so new messages cannot be presented too quickly afterward. For relatively stable communication, limits of decoder reception speed or retention memory are less important: if the decoder needs a second look, the message is still there, within some arbitrary or necessary message life-time.

Eloquent Scenery is designed to be quasi-stable: a Gesture will remain as long as it is pertinent and not interfering with other messages or the user's quiet. However, having Gestures that must be repeatedly looked at by the user conflicts with the idea of a non-imposing, peripheral scheme of communication. Gestures cannot be so obscure that they require prolonged study. Also, some visual properties are more prominent than others and are noticed immediately by the viewer. A sort of visual hierarchy can be formed with striking, quickly perceived features at the top and subtle, slowly perceived features at the bottom. The use of such a hierarchy will be discussed below.

## 5.2 Semantic Concerns

Doblin places five statements within the semantic level of his network model. They are "a message must be decodable," "a message must have proper hierarchy," "a message must have a prepared decoder," "a message must be properly crafted," and "a message must have congruity." Of these, only the first three are considered by Doblin to directly affect the goal of informing, but all affect it in some way.

### 5.2.1 A message must be decodable

The components of message expression must have clear and definite relationships to the the components of message meaning. Some expressional components may be formational parameters while others perform modifying functions. Formational parameters are structural components like phonemes which carry no meaning independently but which, when combined in pairs or multiples, form entities that can be defined with particular meanings by convention. Modifiers adjust or enhance the conventional meanings of the word-like entities. Eco's work described in Chapter 4 calls formational parameters *figurae* and modifiers vectors. To use an appropriate example, in American Sign Language for the Deaf the formational parameters are hand configuration, place of articulation, and movement. Some modifiers of this language are reduplication, speed, elongation, and various tempo changes. This example will be explored in greater detail in the next chapter.

The individual properties of Gestures can be rated on their respective performance to represent quantitative, ordinal, and nominal information. A quantitative value ranges over a more or less continuous scale. An ordinal value comes from a discrete set of ordered values. A nominal value comes from a discrete set of arbitrary values. Since formation parameters are discrete and arbitrary components, they are best fit by properties that can represent nominal information well. On the other hand, quantitative properties can perform the task of modifying more easily. An initial categorization of Gesture properties is presented in the next chapter as well.

### 5.2.2 A message must have proper hierarchy

The hierarchy of a message's expression should reflect the hierarchy of the message's meaning. In particular, the prominent features of the expression should correspond to the significant components of the meaning. This allows the decoder to scan the message for essentials and to begin to form or recall a context for the full message. Of course, the decoder may choose to ignore the message upon realizing its context, for whatever reasons. The form of the message can aid, or hinder, both the ease of its interpretation and the speed of any rejection-acceptance decision.

The proposed form of Gestures has three layers. The layers are "How important is this message?", "In what way is this message interesting?", and "What are the details of the message?" Each subsequent layer has an increasing degree of complexity. Specifically, importance is a linear scale with only the upper end needing non-trivial expression. Interest is multi-dimensional, but only a point in this space is needed to identify the primary focus. The specifics of a message are as rich as possible or necessary, using whatever properties are appropriate. How particular properties of individual Gestures should be used to code these three layers will also be discussed in the next chapter.

### 5.2.3 A message must have a prepared decoder

The decoder must know both the relationships between expression components and meaning components and the hierarchy in which both types of components are organized. There are two extremes to defining the relationships and hierarchies. The one uses universal definitions, and the other uses personalized definitions. Universal definitions are both decoder and task independent. Once taught the vocabulary or how to look up definitions, the decoder can use the language in any situation. For the most part, the English language has universal definitions of words, and the American Sign Language has universal definitions of hand and arm gestures. On the other hand, personalized definitions are limited to a single decoder or to a small number of decoders. Diary codes use personal shorthand and are designed so that only the creator can understand them. Espionage codes are often mission dependent and only those involved in that mission have the necessary code keys.

Eloquent Scenery is designed to be neither a universal language nor a personal code, but rather a task specific tool. That is, for a given task, the Gestures should be largely user-independent. This allows the group of users involved in the specific task to educate each other on the specialized Gesture vocabulary for that task. However, this language choice also forces users to become re-educated when they use the system for new and different tasks. A number of task specific Gesture definitions are proposed in the next chapter.

### 5.2.4 A message must be properly crafted

Doblin places craftmanship within the semantic level, but it has a technical component as well. A well crafted message has, as Doblin says, authority. However, it also often provides a certain amount of uniformity and, perhaps, comfort so that the decoder need not be distracted by the details of expression. The authority is mainly a psychological factor, while the uniformity and comfort may address some psychophysical factors. For example, a spokesperson with a squeaky voice may seem amateurish, but it may also be truly annoying. Poor penmanship actually may impede reading performance.

Eloquent Scenery does not need to project authority. It has the authority of a tool, a helpful device. It does need to be concise and should not be unnecessarily distracting. This need is related to the requirement of hierarchy mentioned above. However, instead of ordering the types of message, crafting gives the different types of message the necessary and only the necessary amount of power. These issues are discussed somewhat in the next chapter.

### 5.2.5 A message must have congruity

This statement refers to congruity between message form and message content. Doblin mentions the fact that many types of expression have mood connotations, and these connotations especially affect messages designed to persuade. Mood is less important for information presentation, but there are still opportunities for congruity. In particular, iconic similarities can be used to aid interpretation by providing mnemonics. However, unless iconicity is used consistently, the congruity will become gratuitous. The natural development of American Sign Language appears to be away from iconicity toward arbitrary, structured association.

No mandatory congruities are proposed for Gestures. If a quantity's diminishing is more significant than its growing, then it may be represented by a quantitative property increasing rather than decreasing. Likewise, values of nominal properties, such as shape, may be chosen because of their relationship to other possible values rather than because of their similarity to some quality of the content.

## 5.3 Effectiveness Concerns

Doblin places three statements within the effectiveness level of his network model. They are "a message must have integrity," "a message must be credible," and "a message must be exciting." These statements accompany, respectively, the goals of informing, persuading, and stimulating.

### 5.3.1 A message must have integrity

The effectiveness of information is tied to its integrity. A message that says or implies something with one of its parts while saying or implying the contrary with a different part causes confusion for the decoder. The confusion not only nullifies whatever information is in the particular message, it erodes the decoders confidence in other, similarly presented messages. However, messages should not inspire more confidence than they truly deserve. While messages should be as integral as possible, they should not hide inconsistencies or discrepancies for this end.

The goal of Eloquent Scenery is a system which informs the user on circumstances that it perceives through whatever mechanisms. If two mechanisms interpret the same target circumstance differently, there is an ambiguity in interpretation. However, this ambiguity is predicted by the combination of the rules of interpretation and the actual circumstance, thereby illustrating the limitations of these rules. The system would be censoring examples of its own limitations if it were to weed these ambiguities out. If the rules of one message component say a condition is safe while those of another recognize a danger, and there are no means of resolving the conflict, then the conflicting message should be presented, possibly even highlighted. This allows the user to seek more detailed information from whatever external resources are available.

### 5.3.2 A message must be credible

A message that is designed to persuade needs to be credible in the eyes of the decoder. The decoder must be willing to dismiss previous or alternate messages in favor of the present one. Even pieces of information must compete like this to a lesser extent. Like integrity, credibility inspires confidence in the decoder. Again, this confidence should not be fabricated because of the risk of deceiving the decoder.

The credibility of parts of Eloquent Scenery needs to develop over time. The user will trust those Gestures that reliably represent areas of the environment well. Some Gestures will only serve their purposes marginally well, therefore the user will want to discount, if not completely ignore, unless they are eventually redefined. For this reason, the definitions of individual Gestures should be allowed to evolve over time.

### 5.3.3 A message must be exciting

Excitement of some form is necessary to arouse or stimulate the decoder. Stimulation for stimulation sake need not be hierarchical, coherent, or understandable. But stimulation is rarely a sole goal. Even popular entertainment often has additional cultural purposes either toward unity or toward dissent. Excitement can be used to keep the decoder's interest or to accent particularly important messages. With a predominant goal of informing, excitement can be consciously directed to individual points.

Eloquent Scenery uses visual excitement to signify importance. An urgent message could be made exciting by vibrating its Gesture quickly, for instance. In general, however, Eloquent Scenery is in the background and should not be too provocative.

# Chapter 6

# Signs from Gestures

In this chapter, I propose some ways of using Gestures to convey information. The first section specifies which visual properties of Gestures are suited to performing different tasks independently. The second section explores the notion of using composites of the various properties to provide a richer set of messages, which set approaches a rudimentary language. The final section suggests some specific allocations of properties in different user contexts.

## 6.1 Gesture Properties as Message Qualities

As outlined in the last chapter, we have three message qualities that need to be carried by the visual properties of Gestures. One quality announces importance to an inattentive user. Another quality demonstrates relevance quickly to a half-way attentive user. The final quality provides the actual information. This section rate the various Gesture properties according to how well they contribute to this final quality by providing different types of information. For the section, I restrict the types of information to some combination of quantitative, ordinal, and nominal values.

Figure 6.1 catalogues the strength of various Gesture properties with regard to these qualities. The first two columns show the results of the experiments described in Chapter 3. The other three columns are based on the ranking provided by Jock MacKinlay in "Automating the Design of Graphical Presentations of Relational Information." My grades differ from MacKinlay's because of the different graphical assumption. Although I attempt to justify my grades, they are not grounded in a firm perceptual theory. In this way, I am following MacKinlay's argument:

> ...[T]here does not yet exist an empirically verified theory of human perceptual capabilities that can be used to prove theorems about the effectiveness of graphical languages. Therefore, one must conjecture a theory of effectiveness that is both intuitively motivated and consistent with current empirically verified knowledge about human perceptual capabilities. (page 124)

For quantitative, ordinal, and nominal information, MacKinlay ranked perceptual tasks according to their ability to effectively present data. What MacKinlay call perceptual tasks, I have been calling visual properties. This is simply a shift from the viewer's point of view, so to speak, to the computer's representational point of view.

I cannot use MacKinlay's rankings exactly, however, because they involved assumptions different from the ones in effect here. In particular, MacKinlay was concerned solely with static graph displays with multiple data points, therefore he assumed both ruled axes and perceptual interaction between concurrent message units. Eloquent Scenery uses a nondescript, neutral background and has completely independent Ges-

| | | | | | |
|---|---|---|---|---|---|
| texture | **STRONG** | MEDIUM | weak | **STRONG** | **STRONG** |
| intensity | **STRONG** | **STRONG** | **STRONG** | weak | weak |
| chromaticity | **STRONG** | weak | MEDIUM | weak | weak |
| stain | **STRONG** | MEDIUM | MEDIUM | MEDIUM | MEDIUM |
| size | MEDIUM | **STRONG** | **STRONG** | MEDIUM | MEDIUM |
| orientation | MEDIUM | MEDIUM | **STRONG** | **STRONG** | MEDIUM |
| shape | MEDIUM | weak | weak | MEDIUM | **STRONG** |
| offset position | **STRONG** | MEDIUM | MEDIUM | **STRONG** | **STRONG** |
| path of motion | weak | weak | weak | MEDIUM | **STRONG** |
| extent of motion | weak | MEDIUM | MEDIUM | weak | weak |
| speed of motion | weak | **STRONG** | **STRONG** | weak | weak |
| | relevance (visual prominence) | importance (peripheral attraction) | quantitative information | ordinal information | nominal information |

Figure 6.1: "Skills" of Gesture properties.

tures. Also, rather than ranking the Gesture properties, I merely grade them relatively as *strong*, *medium*, and *weak*.

### 6.1.1 Quantitative Information

Quantitative information involves a continuous value. Necessarily, the value will be limited by the bounds of the medium used to convey it, in our case a Gesture property. MacKinlay's ranking for quantitative information, in order, is position, length, angle, slope, area, volume, density, color saturation, and color hue.

Position for Gestures is handicapped for two reasons: (i) there are no ruled axes, and (ii) the excursionary motion of the Gesture would hide slight differences in offset position. Of course, Gesture position could degenerate to the typical graph sense, but all of the secondary motion properties, path, extent, and period of motion, would vanish. Short of that, position has, at best, medium strength for presenting quantitative information.

MacKinlay's length and area correspond to Gesture size, and slope corresponds to Gesture orientation. Angle and volume do not directly translate. Density, color saturation, and color hue roughly match Gesture intensity, chromaticity, and stain, respectively.

Speed of motion in considered strong here because it is perceived fairly quickly, while extent of motion is only medium strength because it is perceived less quickly and is harder to quantify. Texture, shape, and path of motion are all weak because they cannot represent continuous information well without artifice, such as displaying a floating point number!

### 6.1.2 Ordinal Information

Ordinal information involves discrete members of an ordered set. For example, the day of the week or month of the year is ordinal information. Position, density, color saturation, color hue, texture, connection, containment, length, angle, slope, area, and volume is the ranking given by MacKinlay.

Gesture position is considered strong here because displacements in units of the extent of motion or larger is a clear distinguishing factor. Intensity, chromaticity, and stain are downgraded for Gestures because of the interaction of these properties with the color-rich and unconstrained texture. MacKinlay used the word texture in restricted sense of a repeating, abstract pattern. This interaction reduces the effective range of these properties. For quantitative information the reduction simply restricts the content range, but for ordinal information it destroys opportunities to define distinct plateaus.

Connection and containment are beyond the current project.

Orientation is particularly stronger here than for MacKinlay because of a presumption of natural orientation clues within the Gesture's texture itself. MacKinlay removes shape from consideration for ordinal information because he was dealing with small objects, but the number of sides or protrusions can easily be used to distinguish good sized Gestures.

### 6.1.3 Nominal Information

Nominal information involves discrete members of an unstructured set. That is, members of the set need not bear any relation to other members of the set. MacKinlay's order of perceptual tasks for presenting nominal information is position, color hue, texture, connection, containment, density, color saturation, shape, length, angle, slope, area, and volume.

Once again, the face properties, besides texture, are reduced in strength. The rest essentially follow MacKinlay's order.

## 6.2 Visual-Gestural Languages

In this section, I compare the properties of hand gestures made by deaf people with the defined properties of the Gesture visual objects. Deaf sign language is a visual-gestural language that coordinates the interaction various properties. This coordination creates a potential set of messages greater in number than one which is created by considering the properties in isolation.

### 6.2.1 American Sign Language for the Deaf

The source of my information on American Sign Language is the book *The Signs of Language* by Edward Klima and Ursula Bellugi. One of the main points of this book is that ASL is a full, independent language.

> In this primary communication system [American Sign Language], developed and used by deaf people in the United States, we have found, as linguists have found in spoken languages throughout the world, a form with its own highly articulated means for expressing and relating concepts, and with an underlying network of regularities connecting visual

> form with meaning. ASL is clearly a separate language, distinct from the spoken English of its surrounding community. (page 2)

I do not wish to cover all of the properties of ASL, but rather to briefly show how hand gesture properties are used as combinational units and vectors (See section 4.2.3). In particular, American Sign Language has combinational units or formational parameters at the lowest level where the units themselves carry no meaning, but only their interaction. These units are among those called *figurae* by Eco and are similar in function to phonemes of spoken languages. Also, the signs ASL undergo distinct morphological processes which alter the signs' meanings or their grammatical functions. The form of these processes fall under Eco's vector category, since they affect the meaning of other signs but are themselves meaningless apart from the signs they affect. For example, temporal aspects that denote 'for a long time' or 'over and over' have no real meaning apart from qualifying an event mentioned in an accompanying statement.

### Formational Parameters of American Sign Language

In a spoken language, the *figurae* are phonemes, and series of them form morphological entities. In ASL, there are three *figurae* or formational parameters, and the interaction, largely concurrent, of members of the three parameters form lexical entities. The three parameters are hand configuration, place of articulation, and movement.

> Handshapes are...differentiated by the spatial configurations of the hand, resulting from extension, contraction, contact, or divergence of the fingers and thumb; the digits may be arranged in a variety of ways and may be held so as to form a vast array of static configurations. (pages 43 and 45)

William Stokoe's *Dictionary of American Sign Language* list 19 classes of primes for hand configuration, and each of these classes has a small number of subprimes which vary on a finer scale. Distinct, minor formational parameters which are sometimes categorized under hand configuration are contacting region, orientation, and hand arrangement.

> The primes of place of articulation are defined with respect to particular locations and areas on and around the body within a delimited region we call the signing space. (pages 50-51)

Stokoe listed 12 different primes: five about the head, four along the arm, one on the neck, one on the trunk, and one in the neutral space in front of the torso. Klima and Bellugi claim that those in the neutral space can be divided into those on the horizontal plane, those on the vertical, or frontal, plane, and those on the sagittal plane, also known as the plane of bilateral symmetry.

> If the movements of signs are compared as global wholes, the differing shapes, tempos, directions, oscillations, and dynamics of the motions appear extremely rich and varied; nevertheless, the movement parameter of signs can be described in terms of distinct movement components (in the DASL Stokoe proposed 24 different movement primes) which can occur singly, in sequence, or simultaneously within single monomorphemic signs. (page 54)

There are alternative theories for the analysis of movement, but, loosely, there are a relatively fixed number of common types of movement. These types can be grouped in a smaller number of classes, such as hand-internal movements, wrist movements, directional movements, circular movements, and interactive movements. Hand-internal movements involve wiggling, bending, opening, or closing fingers. Wrist movements consist of supinating, pronating, twisting, nodding, or rotating actions of the wrist. Directional movements are straight traces of the hands in one of the three orthogonal planes, horizontal, frontal, and sagittal, in the neutral space. Circular movements are also within one of the three planes. Interactive movements involve the two hands or a hand and the body.

## Formational Components of Morphological Processes

Chapter 12 of the Klima and Bellugi book is called "The Structured Use of Space and Movement: Morphological Processes." In that chapter, they analyze the way in which signs are built up to provide richer information succinctly in American Sign Language. Rather than sequentially appending gestural components, the movements of signs in ASL are modified by some combination of suppressing, exaggerating, and adding features.

There are two branches of traditional morphology: derivational and inflectional. Derivational morphology deals with forming new words out of old words. Inflectional morphology deals with adding grammatical markers to words to indicate tense, person, number, gender, case, and aspect. For example, past and present tenses place events in time and person index the speaker, the listener, or a third person as the subject.

Inflections for signs particular interesting for this thesis are temporal aspects. These aspects are indications of temporal qualities for events or states such as onset, duration, frequency, recurrence, permanence, or intensity. Figure 6.2 shows the features of modulations for temporal aspect as presented by Klima and Bellugi. The aspects of meaning are grouped into pairs. While the pairs are related semantically, the first involves a state and the second involves a change of state. I only present the function of these features here to give a flavor of the types of sign modification possible.

Taken together, all of the morphological processes described by Klima and Bellugi can be described along eleven spatial and temporal dimensions, where each dimension has only few values, often only two values. The eleven dimensions enumerated by Klima and Bellugi are planar locus, geometric pattern, direction, manner, rate, tension, evenness, size, contouring, cyclicity, and doubling of hands. The manner, rate, tension, evenness, size, and cyclicity dimensions correspond respectively to the *end-marked*, *fast*, *tense*, *even*, *elongated*, and *reduplicated* features found in Figure 6.2.

The values of planar locus are the horizontal plane and the frontal plane. Geometric pattern refers to the overall tendency of movements in a sign, whether toward a point or through an arc, circle, or line. Also, the overall tendency of movements can be directed sideways, downward, or upward.

Manner refers to onset or offset of movement through cycles, whether continuous, end-marked by pauses or holds, or restrained with a noticeable recoil or a check. In the course of cycling, the muscles of the signer can be tense or lax. The rate of movement can be fast or slow, and it can be even or uneven. Also, the movement through the individual cycles can be elongated or abbreviated. The shape or contour of movement through the individual cycles can be straight, circular, or elliptical. This shape is different from the geometric pattern mentioned above, because it refers to local hand movements, while the other refers to overall arm movements.

Cyclicity and the doubling of hands interact with these above dimensions. Movement may be cycled through once, a fixed small number of times, or a nonspecifically large number of times. Also, the movements may involve a lone hand or both hands either in phase or in opposition.

| | |
|---|---|
| *Reduplicated:* | presence vs. absence of cyclic reduplication. |
| *Even:* | evenness vs. unevenness of tempo thoughout a cycle. |
| *Tense:* | presence vs. absence of extra tenseness in hand and/or arm muscles. |
| *End-Marked:* | presence vs. absence of stops or holds at the ends of cycles. |
| *Fast:* | increased vs. decreased rate of movement. |
| *Elongated:* | elongated vs. shortened size of movement. |

| | Reduplicated | Even | Tense | End-Marked | Fast | Elongated |
|---|---|---|---|---|---|---|
| Predispositional<br>'be characteristically sick' | + | + | - | - | + | + |
| Susceptative / Frequentative<br>'easily get sick often' | + | + | - | + | + | + |
| Continuative<br>'be sick for a long time' | + | - | + | - | - | + |
| Iterative<br>'keep on getting sick again and again' | + | - | + | + | - | + |
| Protractive*<br>'be sick uninterruptedly' | - | | + | | | - |
| Incessant<br>'seem to get sick incessantly' | + | - | + | + | + | - |
| Intensive<br>'be very sick' | - | + | + | + | + | + |
| Resultative<br>'get (fully) sick' | - | - | + | + | - | + |

* Protractive aspect is made as a long, tense hold in place. Thus some features do not apply.

Figure 6.2: Features for aspect inflection.

### 6.2.2 An Analogous Language for Eloquent Scenery

Using the above description of a actual, practical gestural language, an analogously structured language using the Gestures of Eloquent Scenery can be devised. Whether this analogous language definition is itself practical, let alone optimal, is a large topic, needing much more research over a longer term.

There is an important difference between how American Sign Language works and how Eloquent Scenery is designed to work. While gestures in ASL are morphological and lexical units, the Gestures in Eloquent Scenery are complete propositions, or what Eco calls super-signs. Naturally, a large number of super-signs are needed to approach the expressive range of a small number of lexical signs that can be mixed and matched within a grammar.

#### Formational Parameters

Corresponding to hand configuration, place of articulation, and movement for American Sign Language are shape, position, and path of motion for Gestures. The very flexible texture property can be added as a formational parameter, too. Not coincidentally, these four properties are the strongest at presenting nominal information.

**Shape** The only constraint on shape is the general one that it preserve a majority of the face. Of course, skinny shapes can be used, but just not along with any relevant textural property values. Probably, the best way to handle shapes is to select a relatively large number of readily distinguishable shapes and use only them as primes. Introducing new shapes in an ad hoc way might increase the occurrence of misidentifications. Shapes should also be considered with regard to their behavior during rotation. For instance, a circular shape may remove the perception of the orientation property. Natural orientation clues may not always be available in the texture.

**Position** Under the assumption of a nondescript background, there are no landmarks or points of reference against which position values can be defined, except for the outer boundary. However, Eloquent Scenery could be redesigned place a background with distinctive topological features behind the Gestures. The effect of this redesign on the anticipated powers of distraction and prominence by Gesture properties would have to be weighed. Alternatively, regularly spaced locations, either along an invisible two dimensional grid or within unmarked concentric rings, could be used to define primes.

**Path of Motion** Movement is restricted to two dimensions for Eloquent Scenery, but, due to the absence of mechanical restrictions, a greater variety of path shapes can be devised. Also, being a super-sign instead of one of a series of lexical signs, a Gesture can spend a longer time tracing its path. Obviously, the path of motion cannot become too complicated or the user will be confused, but since the motion continually cycles, the user need not comprehend in one pass.

**Texture** The texture property can easily be restricted to a fixed, but large, set of primes that can be learned by each user over time. On the other hand, the texture property can be left free-form, where the images contribute iconically to the Gesture's meaning. A compromise between the two may be possible, too. That is, primes could be built around the composition of generic objects within images and the evolution of that composition over a sequence, while the particular objects in the compositions are left open for innovation. For example, one prime might be the motion of an object from the left to the right, and another might be the evolution from one object to five objects.

## Formational Components of Morphological Processes

Many of the dimensions of patterning for morphological processes mentioned for signs cannot be transplanted to Eloquent Scenery, especially not with how Gestures are implemented in this thesis.

Currently, the lifetime, hence the number of cycles, of a Gesture is undetermined, depending on the course of events. In this situation, cyclicity is meaningless. Also, my assumption of Gesture independence disallows the interaction of Gestures in concert. For the two dimensional environment of Eloquent Scenery, different plane cannot be used and the direction of overall movement is impoverished. The tension of muscles cannot be considered unless it is simulated by tiny, random jiggling.

Manner, evenness, contouring, and geometric patterning can play a role in the meaning of Gestures, but they must be built into the paths of motion. The easiest matches are with rate of movement and size of movement. Changes in period of motion directly affect the rate of movement, and changes in extent of motion can be used to elongate or compress movement.

Two additional Gesture properties that can help modify meaning are orientation and size. For anatomical reasons, hand orientation and hand size are incorporated into hand configuration in American Sign Language. Here, however, these properties are free to act independently. Both orientation and size can be fairly finely graded to increase the number of possible combinations of features.

Since Gestures are presented in parallel in a way that signs of sign language are not, there is still the need of Gestures to be ordered according to their relevance. Rather than adding to meaning, intensity, chromaticity, and stain might better serve the function of signifying relevance.

## Language Evolution

The components of a Gesture language considered here are viewed in relation to a working gestural language, American Sign Language. However, ASL evolved more or less naturally from mimetic representation. The structure was not completely planned in advance, but largely discovered afterward. Rather than fixing the function of the various Gesture properties, different arrangements can be tried and evaluated. One of the results of such an evaluation may be that Gestures are too constrained and that new properties need to be developed.

## 6.3 Gesture Signs for Different Contexts

In this section, I speculate the role of Gestures as signs in five different contexts and how Gestures can be used to help fill these roles. The five user contexts considered are concentration, vigilance, diagnosis and repair, search, and curiosity.

### 6.3.1 Concentration

A concentrating user wants infrequent signals. There are many situations where the user may want or need total mental concentration directed toward the main, foreground task. Visual objects constantly shifting about, blinking, or just appearing and disappearing in the periphery will break this concentration. However, there may very well be a small number of justifiable reasons for the system to intrude on the user, particularly emergencies. The best way to indicate a small number of possible, but infrequent message contents is through signals.

The only issue in forming signals with Gestures in this context is the power of the various visual properties to distract. Movement, dynamic sizing, and luminance flickering were the strongest candidates for distraction from the second experiment in Chapter 3. Sizing and flickering can also settle down to fixed value gracefully. The strong secondary properties of motion vanish when position is fixed.

### 6.3.2 Vigilance

A vigilant user wants signals and symptoms. A user who is responsible for the monitoring of various systems or who is interested in the status of various systems needs to know the occurrence of particular events. However, he or she does not always want to be completely surprised by these events. Signals can signify the events, and properly defined symptoms can be used to help anticipate them. Clues are not useful here since they are retrospective in nature.

The signals can be handled as above. In addition to a means of expression, symptoms need a means of continuity. Either the Gesture should remain visible during its lifetime, or it should have identifiable features, i.e., a name. The identity can be provided simply by a static texture, a static shape, or a home position. The dynamism of the rest of the properties can be used for the expression of quantitative information, such as variable values, and ordinal information, such as discrete state progressions. Of course with such a large choice of synchronous properties, the actual set of properties used by the Gesture itself can denote the symptom's identity.

### 6.3.3 Diagnosis and Repair

A trouble-shooting user want symptoms, clues, and samples. A user looking for problem areas in the hope of remedying them needs an accurate representation of the present state, a usable history of past events, and a means of predicting future behavior. Symptoms, as synchronous signs, can only really represent the present state. The important tell-tale signs of previous states can be preserved as clues. Samples are helpful both for recreating previous steps and for simulating future consequences.

Care must be taken to separate symptoms from clues from samples because they involve different time frames. One would want past clues or imaginary future samples interpreted as present, active symptoms. A property like stain which is not particular strong in presenting any type of information could be effectively used to separate time frames.

### 6.3.4 Search

A searching user wants symbols, names, and stylizations. A user whose task is to find a specific item or set of items primarily needs to be able to distinguish and categorize individual items. Since symbols and names naturally form classes, they are the natural choices to represent items. A symbol is already a type of stylization, but broader types of stylization can be used, too. For instance, the clustering of items according to their characteristics is a stylization. Also, the use of more visually prominent features for more likely candidates is a helpful stylization.

In this case, nominal information and relevance are the stressed message qualities. Texture, as a still or as a movie, shape, and path of motion can be used for nominal information. Position can be more effectively used for relevance. Any of intensity, chromaticity, stain, size, and orientation can also be used to show relevance.

### 6.3.5 Curiosity

An inquisitive user wants combinational units, and vectors. The most information hungry type of user is one trying to satisfy simple curiosity. Brief, concise, and descriptive statements are the most helpful. A possible way to approach general descriptions is to form a general purpose language such as the one hinted at in the previous section. Combinational units are used to form statements and vectors can be used to modify or qualify these statements.

A user who uses Gestures out of curiosity will naturally want to ask questions and give orders. This thesis has ignored direct communication from the user to the Eloquent Scenery. I did this to concentrate on the issues of information presentation. The next chapter presents some topics about interacting with Gestures for future consideration.

# Chapter 7

# Beyond Eloquent Scenery: Conversational Scenery

There are two main considerations for a future attempt to expand Eloquent Scenery into Conversational Scenery, that is, to have Gesture objects that take orders, answer queries, request information, and promise to notify of future events.

The first consideration is how does the user talks to the Gestures directly. Currently, Gestures are controlled by an application program. Of course any application can define its own user interface to the Gestures, but a set of conventions, possibly managed by a single task, may ease the application design process and reduce the amount of extra user education.

The second consideration is of the Gestures themselves: how should they be redesigned to respond to loosely defined external events. Currently, a particular property can be tied to an external event via a function or a variable. But with the present design, there is no simple way for a Gesture as a whole to transform in response to a conjunction of disparate events. For example, an excited, distracting Gesture may stand down or become quiet if the user moves the mouse to the vicinity of the Gesture and clicks a button. In order to give a sense of the domain of conversational moves, I present a brief overview of the theory of speech acts put forth by John R. Searle in *Speech Acts* and *Expression and Meaning.*

## 7.1 Speaking to the Scenery

Pinpoint mouse selections and exact click-and-drag movements are suited to docile windows with buttons, scroll bars and pop-up menus. However, one does not need to interpret the mouse movements so precisely. Instead, the motion of the mouse can be decomposed into the secondary properties of position we have used to describe Gesture position, namely path of motion, extent of motion, and speed of motion.

With these secondary properties in hand, mouse gestures can be viewed as a limited projection of full, three dimensional gestural signs. In the previous chapter, I described the features of American Sign Language. To review these features: The formational parameters of signs in ASL are hand configuration, place of articulation, and movement. The formational components of morphological processes are planar locus, geometric pattern, direction, manner, rate, tension, evenness, size, contouring, cyclicity, and doubling of hands.

With a single, standard mouse place of articulation can be defined by the rough location of the mouse cursor when a mouse button is pressed. The chord of buttons pressed can correspond to hand configuration, and the motion of the mouse with buttons pressed can correspond to movement. Obviously, neither of these last two substitutions are as rich as the original, but the basic idea is the same.

Some of the dimensions of patterning movement do not project to mouse manipulations. In particular, planar locus and tension cannot be used at all. The doubling of hands could only be used on a two mouse system. The rest of the dimensions can be used essentially as for full gestures, understanding that the physical demands of moving a mouse are a little different than those of moving hands freely.

Despite these limitation, a complete gestural language for mice can be developed.

## 7.2 Speech Acts

One can look at language use as a series of acts. Thus, a conversation between two people or entities can be described as one or more speech acts by the first, followed by one or more by the second, followed by more by the first, and so on. However, an 'act' can be defined in a number of different ways.

A locutionary act can be described without regard for the intentions of either the speaker or the hearer. It is described in terms of articulation, syntax, and, to some extent, semantics alone. An illocutionary act can be described in terms of the representation of beliefs, desires, or intentions by the speaker, regardless of the particular utterances used. The speaker may express, request, describe, promise, or declare in the same way using completely different utterances. Finally, a perlocutionary act can be described in terms of the motives, rather than the moves, of the speaker with regard to the beliefs, behavior, or intentions of the hearer. These descriptions are above the particular set of illocutionary moves used by the speaker, and may only be correct with respect to a subset of possible hearers. That is, different hearers may respond in different ways to illocutionary and locutionary acts used to form a particular perlocutionary act.

For the purposes of a communicative but essentially unintelligent user interface, only locutionary and illocutionary descriptions are really relevant. Except in dangerous circumstances or in response to security restrictions, the computer is completely obedient to straight, well-formed orders and requests. In the other direction, it is somewhat inaccurate to ascribe motives to a computer, at least at the present time. Perhaps advances in artificial intelligence will eventually change this.

Up to now, this thesis has been directed toward defining locutionary acts, so that Gestures can convey information to the user. However, presenting information is only one type of illocutionary acts. Searle provides a classification of five types of illocutionary acts. The five types are assertives, directives, commissives, expressives, and declarations.

**Assertives** Of the five types, assertives, or statements of belief, cover the realm of presenting information. The other types have different points or purpose. The major component of an assertion is a description of the belief, but it also may contain a degree of certainty or sincerity.

**Directive** The purpose of a directive is to express a desire by the speaker for the performance of some action in the future by the hearer. A directive may take the form of a request, of an order, or of something in between. Searle appears to consider all questions directives, since they are requests for the hearer to answer. However, some questions may be instead considered assertions tinged with doubt. In any event, the major components of a directive are a description of what is to be done including a rough time scale, an indication that the speaker wants the hearer to do it, and some impression of the strength of the speaker's desire.

**Commissive** The purpose of a commissive is to commit the speaker to a future action. A commissive may range from a promise to a threat. The important components of a commissive are a description of the action including a time scale, an indication that the hearer intends to perform the action, and a set of conditions or a level of guarantee.

**Expressive** The purpose of a expressive is to voice a psychological state with respect to a property of the speaker or the hearer. An expressive can range from a thanking to an apology to a condolence. The important components of an expressive are a description of the property and its relationship to the speaker or hearer, an identification of the psychological state, and a level of sincerity.

**Declaration** The purpose of a declaration is to perform an action through words. Declarations are used to name, to resign, to fire, to excommunicate. The only major component of a declaration is a description of the proposition. Since a declaration is an immediate, all or nothing action, sincerity is not an issue. Either the proposition takes effect or not.

Any of these illocutionary acts may be false or invalid, but does not effect the form of the act. An assertion may be false or a lie, a request may be ignored, a promise may be broken, an expression of sympathy may be insincere, and a declaration may be without foundation or force. Nevertheless, the assertion, request, promise, expression, or declaration was made.

Eloquent Scenery, as it is designed now, can assert facts. To the extent that a computer can be said to make declarations, Eloquent Scenery can make declarations. Since computers lack psychological states in the traditional sense, their using expressives would be a bit of a charade. However, a computer's requesting information and promising, in a loose sense, to accomplish tasks can be useful component of a user interface, and Eloquent Scenery cannot handle these as it is currently designed. By restricting the use of Gestures to this one-way communication, I have ignored interactive issues.

In some sense, requests and promises need to be acknowledged by the hearer. The speaker needs to know that the hearer understood a request because there is an assumption that the speaker wants the hearer to perform the specified action. A request can be made blindly, but the requester may never know if an ignored request was misunderstood, missed completely, or dismissed. For a similar reason, the acknowledge of promises by the hearer can be useful. There is an assumption that the hearer of a promise has an interest, either positive or negative, in the speaker's accomplishment of the action. Otherwise, the promise has no value. The maker of an unacknowledged promise may never know the desire of the hearer with regard to the promised action.

For Eloquent Scenery to be expanded to Conversational Scenery, Gestures need to redesigned so that they can wait quietly for a response from the user up to a certain time, and then either press their point by intruding on the user's workspace, or recognize that they may be bothering the busy user and disappear for the time being. Some user interfaces present an unignoreable dialogue box when the system wants a piece of information regardless of how important that information is at a given moment. If the user does not provide the specific information in some of these systems, then the computer assumes some default or zero answer rather than postponing the question to some less critical time. A background system like Eloquent Scenery need not be so thoughtless.

# Chapter 8

# Conclusion

This thesis proposes a system of background communication for workstations. Dynamic visual objects called Gestures are used as signs by the computer to reach an otherwise occupied user.

The primary visual properties of these Gestures are position, texture, intensity, chromaticity, stain, orientation, shape, and size. All but the texture and shape properties are defined with respect to the varying of one, two, or three numeric values. The numeric values are varied either along fixed paths with a fixed extent and a fixed period, or in synchronization with some application variable or function. The texture and shape properties are defined with respect to fixed collection or images or masks that can be cyclically indexed at some fixed rate.

The properties have different strengths at different tasks. Intensity, size, position, and speed of motion can are the strongest for signifying importance through peripheral distraction. Texture, intensity, chromaticity, and position offset are the strongest for showing relevance or for ordering through visual prominence. Intensity, size, orientation, and speed of motion are the strongest for presenting quantitative information. Texture, orientation, and offset position are the strongest for presenting ordinal information. Texture, shape, path of motion, and offset position are the strongest for presenting nominal information.

These different skills can be used together to populate the peripheral region of the user with a hierarchy of informative, possibly intrusive, Gestures, with Eloquent Scenery.

A language can be developed for Gestures structurally analogous to the sign language of deaf people. That is, values of the component properties can be used to composite a large set of entities that can be linked with specific meanings. However, because of the restrictions placed on the domain of Eloquent Scenery, individual Gestures would better form complete propositions rather than the lexical items of sign languages. The specific restrictions that lead to this distinction are that Gestures continually cycle so that the user can evaluate them at leisure, and that Gestures be brief so that the user is not taxed unduly.

Future extensions can be made to Eloquent Scenery can be made so that the user can interact directly with Gestures. One possible system would allow the user to speak to the Eloquent Scenery in mouse gestures.

Long term experiments must be developed to determine if Eloquent Scenery is viable in a real work situation and if it can evolve a useful vocabulary.

# Appendix A

# Data Type Descriptions

## A.1 The Frame

The graphics package used for this project is home-made and is called Frames. The name reflects the primary data object of the package: the Frame. A Frame is a closed rectangular coordinate system with units of pixels by pixels. Frames differ from many rectangular memory arrays in that not all of the pixels in a Frame necessarily have memory associated with them, and the memory that is associated with its pixels may come from different pools of memory and may even change location over time.

Obviously, with current technology having every pixel associated with a random memory location would have seriously high space and time costs. Storing a four byte address per a four byte pixel value creates a doubled space cost which is expensive, but, perhaps, within reason. Randomly accessing via an address per pixel, however, is unacceptably expensive in time for Frame-wide operations such as 'blits', block transfers of pixel information, and 'fills', writes of the same pixel value to every pixel in a Frame. Both main memory access and file access are optimized for the forward sequential case, thus favoring the chunking of pixels in rows, or columns, but only rows in this package. On many graphics cards, including the IBM 8514/A card which was used in this project, display memory transfers and loads are optimized for two dimensional rectangular blocks where a fixed pitch increments the column address by one for a given row address. Useful pitch sizes are powers of two, such as 1024, because bit shifts can be used instead of multiplies for addressing. The optimal chunking of pixels is in the largest rectangular blocks possible.

### A.1.1 Fabric

In the Frames package the rectangular blocks of memory which belong to Frames are called fabric. There are four types of fabric: display, memory, file, and abstract. Appropriately, display fabric uses display memory, memory fabric uses main memory, and file fabric uses system files. Abstract fabric has a single pixel value which logically covers the fabric and which is set by writes and fills and read by reads and blits. Basic Frames, such as DisplayFrames, have only one piece of display fabric, but special Frames called PatchworkFrames can have multiple pieces of multiple types of fabric. The pieces of fabric in a PatchworkFrame form a disjoint set all within the bounds of the Frame's coordinate system.

### A.1.2 The Master Frame

Fabric, as such, does not exist outside of Frames, and the existence of each bolt of fabric is tied to the existence of its master Frame. Master Frames are the only Frames created independent of other Frames and therefore require independently allocated memory or the ability to successfully allocate sufficient amounts of memory. If the creation of a master Frame allocates pixel memory itself, that memory is freed when the master is deleted. Whether or not the pixel memory is implicitly allocated, the fabric associated with that memory, along with whichever non-master Frames still use that fabric, is invalid when the master is deleted. Master Frames must be handled with some care. For this reason, masters cannot be used as destinations for any fabric manipulation functions.

### A.1.3 Fabric manipulation

The major fabric manipulation functions are RedefineFrame, DressFrameWithFrame, StripFrameOfFrame, and PatchFrameWithFrame.

**Redefining** The redefinition function provides a Frame with a new origin and new extents with respect to its old coordinate space. The new Frame can extend past the old boundaries. Whatever parts of the old Frame are outside the new one are sheared of their fabric, and whatever parts of the new are outside the old are initially unclothed. Frames retain no knowledge of past fabric. A Frame completely covered with fabric that is shrunk and then re-enlarged to its former size and relative coordinate system will now only have fabric in the shadow of its intermediate state.

**Dressing** The dressing function can clothe parts of a Frame currently without fabric. A source Frame is positioned relative to the destination Frame and whatever fabric covers unclothed parts of the destination are from then on shared by the two Frames. No pixel information is changed during this operation, just the logical data structures.

**Stripping** The stripping function removes specific pieces of fabric from a Frame. This time, a reference Frame is positioned relative to the target Frame and whatever fabric of the target matches overlapping fabric from the reference is sheared off the target and forgotten. Again, no pixel information is changed within the fabrics themselves. A Boolean passed to the function determines whether the fabrics have to match. A true value strips fabric that matches exactly the piece that overlaps, while a false value strips any piece of fabric that overlaps.

**Patching** The patching function replaces pieces of fabric already belonging to a Frame. As in the dressing operations, a source Frame is positioned relative to a destination Frame. However, the opposite condition causes the fabric to be shared. Namely, only when fabric covers fabric is the source fabric picked up by the destination Frame. Since Frames possess no memory of past fabric, fabric that is patched on and then stripped off leaves a hole. A Boolean determines whether there is a change in pixel information. No pixel information is changed if the Boolean is false. If the Boolean is true, the pixel information in the superseded fabric is blitted to the superseding fabric. This second option allows a simple paging out of, say, a DisplayFrame to a file by patching the entire Frame with an uninitialized FileFrame and retaining the pixel information. Any subsequent operations that reference the same Frame still see the same pixel values, but now they are in a system file.

In all three of the dressing, stripping, and patching operations, a Frame is turned into a PatchworkFrame or a basic Frame depending on whether or not there are multiple pieces of fabric inside the Frame. Before explaining how PatchworkFrames are used in this project, let me mention a potential use of them in general. Very often the shape of pixel information is not rectangular, or is not orthogonal to the coordinate space. In such cases, rather than using a full bounding rectangle, which is sometimes needlessly wasteful of space, a PatchworkFrame could be created by successive dressing operations that would tile the needed regions with valid memory while leaving the other regions unclothed or clothed with cheap abstract fabric.

## A.2 The Puck

I originally wrote the Puck data type using a proto-type Frame package without PatchworkFrames. I have since rewritten it more simply using them. This section describes the implementation of Pucks.

A Puck is merely a sophisticated cursor. There were five particular design constraints for Pucks. First, the color richness of a Puck's face may be either slight, using few colors, or great, using many colors, or somewhere in between. That is, the number of colors used to paint a Puck is not necessarily fixed across the set of Pucks. Second, not only might multiple Pucks exist and move at the same time, they must be able to move over and under each other according to their two-and-a-half dimensional layering. Third, a Puck may be fairly large, although still smaller than typical windows. A reasonable size for a Puck is 120 pixels by 90 pixels or roughly 1/28th the area of a 640 x 480 screen. Fourth, the pixel values used to paint a Puck's face may change frequently over time, and if the pixel information are indices into a color lookup table or color code table, then the color code table may also change frequently over time. Fifth and finally, the shapes of a Puck may be other than rectangular. A Puck is expected to have a face, and cannot degenerate into a narrow line, but the face may be, for instance, oval or ragged.

All of these constraints present difficulties for the implementation. Also, there are a number of necessary trade-offs between them given current technology. I will consider each in turn.

### A.2.1 Rich in Color

Since a Puck may be rich in colors, it must use full pixels and not some subset of bit-planes. A simple way to make a completely smooth monochrome cursor, or one which has only a few colors, is to allocate one or a few bit-planes for the cursor's exclusive use. For example, on an 8-bit framebuffer one bit of each pixel may represent whether or not the cursor covers that particular pixel. Of course, this means that only 7 of those 8 bits are available for color information. The number of colors for images is cut from 256 to 128, and the cursor has only one color. The loss of one-eighth of one's available screen memory may be tolerable, but the loss of one-half of one's colors is not.

The goal of motion over an arbitrary background is to preserve the covered regions for when they might reappear. However, the covered regions can be saved in different ways. Either the actual pixel information can be stored away or whatever information necessary for recreation of the pixel information can be remembered. The choice between the two depends on the relative volatility of the moving object to the covered region. If the object moves very often compared to changes in the background and the background must be partially recreated on every object move, the rate of background activity is artificially raised to that of the object. On the other hand, if the background is likely to change more often than the object is moved, as is the case for static window systems, and some off-screen corner, perhaps a majority, of it must be kept up-to-date for the moves to work properly, then the background activity will be made more sluggish by always preparing for something that happens only once in a while. In this project, a Puck can be expected to move constantly, therefore saving the background's raw pixel information is the more convenient. Nevertheless, an efficient method of updating that off-screen memory must be provided.

### A.2.2 Multiple Pucks

Being able to update off-screen parts of a Puck's background is especially important when there are other Pucks that may be moving as often as it. PatchworkFrames can be used to manage the fracturing of the effective background seen by a Puck that moves under other Pucks.

When the top-most Puck is drawn or transferred its destination is a sub-Frame of the screen Frame. The pixel information from the destination Frame is saved to the Puck's save Frame and the Puck's own pixel information is blitted to the destination. The fabric patching operation allows the lower Pucks' procedure to be as simple. Before the transfers from and to the destination, the destination Frame is patched successively with all of the save buffers from the Pucks in higher levels, starting with the top-most's and going down. The patches are offset by the relative distances of the higher Pucks to the target Puck. If some of the higher Pucks do in fact overlap the target Puck, their patches will take hold. Otherwise, their patches will be ignored.

### A.2.3 Large Pucks

So far, a Puck at any layer can save what it will cover and transfer itself onto the screen properly. With this ability for a given object, apparent motion can be achieved by restoring what had just been covered, saving what is to be covered, and then drawing the object. Each of these three steps is a full transfer. While these operations are all that are needed to simulate motion, timing considerations still affect the quality of that simulation. Naturally, as the object's area becomes greater, the time required for three full blits becomes more significant. In addition to this slowness itself, an annoying visual effect becomes exaggerated as the size of these blits grows. Namely, due to the order of these steps, every pixel within the bounds of the object spends more time as the background value than as the object value. As the video signal is continually and rapidly being generated in parallel with all of these Frame operations, at a certain point the generated video starts to show more of the background and less of the object. The end result is that the object becomes specter-like or semi-transparent when it moves. If the object moves often, as it does in this project, then the object is mostly a specter.

One way around this visual illusion is double-buffering. Double-buffering uses two full screen buffers which can be alternately designated as the source for video generation. The idea is to display whichever of the two is not undergoing revision. This technique works nicely if you have the extra memory and the switching capability. The IBM 8514/A card has neither. However, it would still be worthwhile to reduce the time devoted to the pixel transfers.

Assuming motion small relative to object extent, restoring only the edges around the object helps speed up the move's pixel update and lessen the specter effect in the absence of double-buffering. For example, if an object is to move right 10 pixels, restore the left 10 columns, shift the save buffer left 10 pixels, save the 10 columns to the right, draw the object in its new position. If one uses two save buffers, blit the still to be covered portion from the old save buffer to the new, save the 10 columns to the right of the object on the screen, restore the left 10 columns, and then draw the object. When the object is 100 pixels wide, for instance, 210% of its area is transferred instead of 300%. Of course, the real time saved depends on the overhead associated with the extra fragmented blits. While the edges will still be specters, the large majority of the object area will only have object values during the entire course of the procedure. Double-buffering, if feasible, will smooth out the remaining unwanted visual effects.

Using Frames, there are still only a relatively short list of steps to perform a move operation. (1) From the Frame of the active range (with objects on lower layers drawn and the fabric from the save buffers of objects on higher layers patched) define the old and new destination Frames at their appropriate positions (and with their appropriate sizes, for they need not be congruent). (2) Positioning the new destination in the old's coordinate space, strip from the old destination Frame whatever fabric overlaps. (3) Blit the save buffer to the stripped Frame (which for small displacements causes writes only the the edges). (4) Using the new width and height, define a sub-Frame of the save buffer Frame with an origin at the same place as the new destination's origin in the old's coordinate space. (5) Dress the newly defined Frame with the fabric from the new destination Frame (with no offset). (6) Blit this dressed Frame to the save buffer Frame. (For small movements, this operation shifts the majority of pixel information in the save buffer around and saves the new edges. Steps (4) and (5) are not the only way to accomplish this function. For example, the source Frame could be defined within an appropriately patched range Frame, but this involves more internal bookkeeping work.) (7) On the IBM 8514/A card, if the shape of the object is non-rectangular or is rotated, prefabricate the source texture Frame by blitting the save buffer to it using a sticky pixels (explained below). (8) If the set of the object's colors is changing with the new texture, wait for vertical retrace and then load the color set into the display as quickly as possible. (9) Finally, blit the prefabricated Frame to the new destination.

### A.2.4 Non-rectangular Shapes

Allowing a Puck to have a non-orthogonal or non-rectangular shape is the next design constraint. This also bends the multiple Pucks constraint, but because of a time slow-down, not because of an unavailability of resources. Drawing some Puck's non-rectangular face is fairly straight-forward. If the graphics card has a transparency blit, then the portions of the source Frame outside of the face can be filled with a transparency value and the blit performed. If the graphics card has a masked blit, then a mask Frame can be provided for the blit to the destination Frame. The IBM 8514/A graphics card has neither a transparency blit nor a masked blit, but does have sticky pixel blits. If a destination pixel value is within a certain range, then it can be overwritten; otherwise, it cannot be. Rather than painting parts of the screen with these sticky pixels and blitting, the regions of the source Frame outside of the face are painted with them and the destination blitted back to the source using the special blit. After this, the complete source Frame can then be blitted to the destination Frame seamlessly.

The main problem is allowing movement under a non-rectangular shape. Up to now, the use of PatchworkFrames has accomplished lower layer movement. Patchwork-Frames, however, have only rectangular pieces of fabric and have too much overhead to permit reasonably the use of a lot of small, pixel-thin scraps of fabric. In the absence of pixel-specific masks, the higher level Pucks, or at least their overlapping corners, need to be redrawn. In this project, this final step is ignored due to the following conditions: 1) the Pucks in use are nearly rectangular, with small "transparent" regions, 2) all of the Pucks are already updated as frequently as possible, and 3) although there must be accounting for overlap, the Pucks are mostly spaced out.

### A.2.5 Dynamic Textures and Colors

As hinted above, blitting pixel information onto a destination Frame is the method of drawing a Puck's face in this project. For performance reasons, a Frame in display memory is the best source for this blit. The pixel information to be used as the Puck's face is prefabricated in an off-screen source Frame. Although this prefabrication step may add to the overall time required to perform a move, it helps remove unwanted visual effects, especially when the pixel information and the colors used to realize them are constantly changing.

Changing the texture, the pixel information, of a Puck's face without changing the set of colors used by it is fairly trivial for images. One simply blits the new pixel information to the source Frame. If the texture changes rapidly, then there is no need for the texture information to persist off-screen. That is, by the time the source is needed to perform a move operation, the source itself needs to be updated. Different Pucks can share a common source buffer.

Changing the colors used by a Puck's face introduces a couple of problems. First, mismatches between the pixel information found on the screen and the color code table used to translate those pixels into a video signal must be avoided. If the colors are simply undergoing some transformation, like getting more intense, less saturated, or redder, then the discontinuity might not be noticeable. However, if the colors are changing in a completely arbitrary way, then the result will be garbage similar to static. The ideal circumstance is to change both the Puck face and the color code table within the time it takes the video generation to get from the end of the old position, through the vertical retrace, to the start of the new position. Unfortunately, most displays do not offer that precision of control. Some, like the IBM 8514/A, do offer a pause until next vertical retrace function. The time slot to work in is cut, on average, in half. This technique works except when the Puck moves near the top of the screen and the destination is fractured.

The Second and more devastating problem of a non-constant color set is that the the colors cannot be shared by other parts of the display. The Puck's colors must be kept separate from those of other Pucks and the background. This is very unfortunate because it works directly against the rich in color constraint. For example, given 256 colors for a display, only three Pucks with 80 colors each can be shown, with only 16 colors remaining for the background. Obviously, it also works against the multiple Pucks constraint.

## A.3 The Reel

In this project, the texture of a Puck's face can change over time, but only in repeating cycles. This concession is made for two reasons. First, without adequate compression the data bandwidth needed to provide the illusion of movies is greater than the hard-disk access rate on the IBM PS/2 system used. However, storing all of the pixel information needed in random-access memory yields a sufficient speed for a relatively large Puck, say 192 pixels by 128 pixels, or a small number of smaller Pucks. Main memory size is constrained under the version of the DOS operating system used, so a virtual disk is used instead.

The type of communication desired in this project provides the second reason to use only short, cyclic movies. Namely, a fusion of stable and momentary communication is wanted. Since the messages will be delivered in the user's peripheral vision, only being noticed occasionally, parts of the message's expression must be repeated cyclically to present the context and identity of the message. Other qualities of the message, those classified as modifiers, can coincide with immediate changes to some computer state independent of the stable parts. These synchronous parts are momentary in nature, gone after a moment, and, therefore, cannot be too complex. A short, repeating texture sequence can be viewed carefully once at any time during the message's existence and does not require constant attention.

The data type I designed to collect together the individual Frames in a texture sequence, along with whatever color sets are appropriate to each Frame, is called the Reel. A Reel is created with four parameters: the Frame width, the Frame height, the number of colors used in each Frame, and the number of Frames in the sequence. A more complex implementation of the type might allow any or all of these four to vary over time. The only one that can be varied for a particular Reel is the number of Frames, although it is not really designed to change often. Although width and height are constant for the Frames found in a particular Reel, the Gesture type allows dynamic resizing of the actual Puck's face by performing the appropriate rescaling of the Frames.

The use of a Reel is very simple. After allocating the Reel by passing the four parameters, the Frames and the color sets can be passed independently in any order. Internally, a large FileFrame is created where the collection of Frames are stacked above each other. Although it is not implemented now, it would be possible to use a PatchworkFrame to conceptually glue together separate files, if one file could not be created large enough. In parallel to the large FileFrame a large ColorTable file is created. Individual Frames or Frame collections are added to the Reel. This causes the pixel information found in each Frame to be blitted to the big Frame. Similarly, a color set is added to the Reel by writing the color information to the appropriate position or consecutive series of positions.

The Frames and color sets are then accessed one by one as they are needed. The Reel type only manages the necessary memory bookkeeping. A Track is used to control the actual ordering of Frame access from the Reel.

## A.4 The Track

Tracks are designed to describe and control any varying parameter value, such as the indices into a Reel, position, size, intensity, etc. All Tracks have periods within which a local time variable cycles. Using a particular time value, the Track calculates a value and returns it. Even though the time value cycles, the values returned by a particular Track at two different instances using the same time value are not necessarily equal. Certain types of Tracks are context-sensitive: changed by value or events external to the Track itself.

There are six types of Track implemented: StationaryTracks, VariableTracks, FunctionTracks, MouseTracks, StraightTracks, SplineTracks. This list is definitely not exhaustive of all possible conceptual classes of Tracks, but, if needed, the FunctionTrack can simulate any desired performance. A short description of each Track type follows.

**The StationaryTrack** A StationaryTrack has one value associated with it. Every time it is asked for a value, a StationaryTrack returns its value. The value is a RealNumber (double float), as are all of the Tracks' return values. Naturally, a StationaryTrack's period is irrelevant and, therefore, arbitrarily defined to be zero. Since its value is constant, a StationaryTrack is trivially context-free.

**The VariableTrack** A VariableTrack has an address associated with it. Every time it is asked for a value, the Track dereferences through the address a value of a specified type. The allowable types are a Integer8 (signed char), Whole8 (unsigned char), Integer16 (signed short int), Whole16 (unsigned short int), Integer32 (signed long int), Whole32 (unsigned long int), Real32 (single float), Real64 (double float), Boolean (unsigned char), and FixedPoint (signed long: 16 bit integer, 16 bit fraction). The value read is cast to a Real, scaled by multiplying it with another Real, and, finally, offset by adding the product to a third Real, and then it is returned. Tracks were designed with generality and uniformity in mind, not speed. In the current project, they do not happen to cause bottle-necks. As with StationaryTracks, the period of a VariableTrack is irrelevant. A VariableTrack is context-sensitive in that any other, totally disconnected part of the program may have access to the same address and change the variable's value. Of course, this is the idea.

**The FunctionTrack** A FunctionTrack has an address to a function associated with it. The pointed-to function takes two arguments, an address and a Real, and returns a value of a specified type. The types are same as mentioned in the last paragraph. When the Track is asked for a value, it calls the function with a specified address, specified at the Track's creation, and the current time value. The value returned by the function is cast, scaled, and offset, and then it is returned. The period of a FunctionTrack constrains the time value as described above. That is, the time value is revalued modulo the period before it is passed to the function. A time offset can also be provided to determine the phase of time cycle, i.e., at what time value does the Track start. A FunctionTrack is context-sensitive, too.

**The MouseTrack** A MouseTrack is a special-purpose FunctionTrack. Instead of having a function address, it has a mouse feature associated with it. When asked for a value, a MouseTrack gets the current state of the mouse and reads the desired feature. The features of a mouse are the x and y positions in screen coordinates, and left, right, and middle button positions in terms of the bi-state: 0 for released, 1 for pressed. Again, the value is cast, scaled, and offset, and then it is returned. Since only the current state of the mouse is used, no internal time value is kept by the Track. By definition, a MouseTrack is context-sensitive.

**The StraightTrack** A StraightTrack follows the straight line between two Real end-points. When asked for a value, a StraightTrack calculates the intermediate value based on the time value. There are two cases of this calculation, depending on whether or not bouncing has been chosen. Without bounce, the Track covers the length of the line in one time period and discontinuously jumps to the initial end-point upon lapping. With bounce, the Track covers the length of the line in one-half of a time period and reverses direction for the second half. The value is not scaled or offset before it is returned, since these can be done implicitly by choosing appropriate end-points. The phase of the strokes or the oscillations is fixed by a time-offset. Since a given time value will always imply the same value, a StraightTrack is context-free.

**The SplineTrack** A SplineTrack follows a piecewise continuous cubic spline. The pieces of spline are specified in a Path data structure. A Path is a set of points, where each point has a value, a velocity, and a time span to the next point associated with it. A given path may be defined and stored having any number of dimensions, but only one dimension is used to define a single SplineTrack. When asked for a value, a SplineTrack determines which piece of the Path coincides with the current time value, converts the time value to the spline-local time variable, and calculates the value by adding together the terms for the four, 0 through 3, powers of the time variable. The value is scaled and offset, and then it is returned. The value scale factor is the ratio of the desired value range to the Path's defined point range. A time offset determines the phase of the SplineTrack. As with the StraightTrack, the SplineTrack predictably returns the same value for the same time value and is, therefore, context-free.

# Appendix B

# Acknowledgments

First, I wish to thank those people who have let me do fun projects. Walter Bender, thanks for being my graduate advisor, allowing me to play with all the wonderful garden toys, and helping me define this thesis project. Don Hatfield, thanks for being my friend and boss and initiating many lines of thought provoking chats. Muriel Cooper and Ron MacNeil, thanks for being my undergraduate research advisors and allowing my interest in workstation graphics to develop.

I also wish to thank some of my closest friends. Thanks to Tom Polapink, the craftsman; Mark Russo, the sabino, Linda Autino and Mike McKenna and Maureen McKenna Monteleone and Doug Monticiollo, friends from before; Glen and Fran Conlon, Mr. and Mrs. Cool; Janet *the* Cahn, the old woman of media; Bill *bubu* Butera, the old man of media; Hubert *sphere* Delany and Eric *schwa* Pivnik, housemates from hell; Roger Biasca, my Toscanini's buddy; Barbara *barb* Moore, my film partner; Dimitry Rtischev, the mad russian; Paul *plin* Linhardt, the pink man; Joe *foof* Stampleman, the rapmaster; and the wild and wooly Wachman brothers Joel and

Josh. I especially want to thank Pat *pasquale jr.* Romano and Ann Birne, my family during the thesis crunch. I will always thank my mother, my father, and my brothers John, Wayne, Brian, Tim, and Larry. My best friend Frank Radics needs no thanks, but he has mine anyway, as does his family.

I thank my colleagues from I.B.M.: Alberto, Eric Anderson, Doug Church, Sandy Crowthers, James Fleming, Charles Forsythe, Fritz Giesin, Roger Hagen, Beth Harvey, Tareq Hoque, Bruce MacKinnon, Song Min, Peter Monta, Tim Murphy, Andy Pierce, Irene Shen, Cathryn Szekely, and Ralph Vinceguerra.

I thank my colleagues from the Garden: Kris Bartol, Alex Benenson, Mike *vmb* Bove, Heidi Burgiel, Xiang Chen, Pascal *lacsap* Chesnais, Jim Davis, Uri Feldman, Scott Fullam, Deirdre *dee* Hall, Henry Holtzman, Kevin Landel, Gene Lee, Hakon *howcome* Lie, Lin Liu, Debby Hindus, Sue Kyoung Lee, Sanjay Manandhar, Patrick *paddy* McLean, Chee Kong Mok, Harry Papadopoulos, Natalio *che* Pincever, Abha Singh, Blaze Stancampiano, Laura *teo* Teodosio, John *wad* Watlington, and Chi Wong.

I thank my general colleagues from the Media-Industrial Complex: Vicky Bipart, Hal Birkeland, Vanessa Boris, Tim Browne, Mario Bourgoin, Dave *dead* Chen, Joe Chung, Stuart Cody, Glorianna Davenport, Paul Dworkin, Laureen Fletcher, Gillian Galloway, Bernd Girod, Michael Hawley, Anh Viet Ho, Bill Jarrold, Mary Lou Jepsen, Michael *wave* Johnson, Alan Lasky, Ellen Liao, Andy Lippman, Ben Lowengard, John Maeda, Elaine McCarthy, Janette Noss, Linda Peterson, Patrick Purcell, Mark Sausville, Chris *geek* Schmandt, Ricki Goldman Segall, Megan Smith, Dave Sturman, Jenifer Tidwell, John *jh* Underkoffler, Corinne Wayshak, and Clea Waite.

I have tried make these lists as complete as possible, so I can remember them all. If I left anyone off these (admittedly rather impersonal) lists, then it is probably because I hate them, although it could be because I never felt I really met them or because of an honest mistake.

# Bibliography

[1] Shi-Kuo Chang, Tadao Ichikawa, and Panos A. Ligomenides, editors. *Visual Languages.* Plenum Press, New York, New York, 1986. TA1632.v56.

[2] Jay Doblin. A structure for nontextual communications. In *Processing of Visible Language 2*, pages 89–111, New York, New York, 1980. Plenum Press.

[3] Donis A. Dondis. *A Primer of Visual Literacy.* MIT Press, Cambridge, Massachusetts, 1973.

[4] Umberto Eco. *A Theory of Semiotics.* Indiana University Press, Bloomington, Indiana, 1976. P99.E3.

[5] Frederik L. Engel. *Visual Conspicuity as an External Determinant of Eye Movements and Selective Attention.* Pergamon Press, Oxford, England, 1976. QP491.E5.

[6] Patrick Griffiths. The communicative functions of children's single-word speech. In *Children's Single-Word Speech*, pages 87–112, New York, New York, 1985. John Wiley and Sons. P118.C475.

[7] Mary Ritchie Key. *Paralanguage and Kinesics (Nonverbal Communication).* The Scarecrow Press, Inc., Metuchen, New Jersey, 1975. BF637.C45.K48.

[8] Edward Klima and Ursula Bellugi. *The Signs of Language.* Harvard University Press, Cambridge, Massachusetts, 1979.

[9] Paul A. Kolers, Merald E. Wrolstad, and Herman Bouma, editors. *Processing of Visible Language 2.* Plenum Press, New York, New York, 1980.

[10] Jock MacKinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, pages 110–141, April 1986.

[11] Neville Moray. *Attention: Selective Processes in Vision and Hearing.* Hutchinson Educational, Ltd., London, England, 1969. BF321.M67.

[12] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, London, England, 1969. P105.S439.

[13] John R. Searle. *Expression and Meaning: Studies in the Theory of Speech Acts.* Cambridge University Press, London, England, 1979. P95.S4.

[14] Thomas A. Sebeok. *Contributions to the Doctrine of Signs.* Indiana University Press, Bloomington, Indiana, 1976. P99.S32.

[15] John W. Senders, Dennis F. Fisher, and Richard A. Monty, editors. *Eye Movements and the Higher Psychological Functions.* John Wiley and Sons, New York, New York, 1978. QP477.5.E93.