37

AN INTELLIGENT SKETCHPAD:
A Gestural Language for Denoting Temporal Relations in Dynamic Design

by Karen Donoghue

Bachelor of Computer Science, Tufts University
Medford, Massachusetts, (1987).

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning in Partial Fulfillment of the Requirements for the Degree of

Master of Science

at the Massachusetts Institute of Technology
June, 1993

Author

Karen Donoghue
Program in Media Arts and Sciences
May 7, 1993

Certified by

Muriel R. Cooper
Professor of Visual Studies
Thesis Supervisor

Accepted by

Stephen A. Benton, Chairman
Departmental Committee on Graduate Students

AN INTELLIGENT SKETCHPAD:
A Gestural Language for Denoting Temporal Behaviors in Dynamic Design

by Karen Donoghue

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning on May 7, 1993 in partial fulfillment of the requirements for the Degree of Master of Science.

Abstract

Sketching offers an intuitive and rapid form of data compression, one in which visual ideas can be quickly expressed. Electronic sketching now allows us to capture subtleties of human hand motion articulated during the *act* of sketching: changes in speed, pressure, and tilt of the stylus are reflected by the input device. Computer systems should be capable of understanding the syntactic, semantic, and temporal implications of sketched input derived from the *gestures* used to create these sketches. The electronic medium allows for sketching on a "page" which is no longer two-dimensional but multi-dimensional and has the added dimension of time. This thesis postulates a new convention which allows us to interpret sketched marks made upon this page and to tell objects on it how to behave over time, spatially, in relation to each other and to their larger context. Their implications are then manifest in the visual appearance of the objects on this page and in changes to the behaviors of these objects over time.

This thesis presents a research system referred to as an Intelligent Sketching Substrate, an electronic page which interprets sketch marks made upon it in terms of the way in which the marks are drawn. The system defines a gestural sketching language enabling a designer to express the visual appearance, spatial location, and temporal behavior of an object in a dynamic electronic environment by interpreting characteristics of sketch marks. Such characteristics include articulative, geometric, spatial, and temporal qualities expressed during the creation of these sketch marks. The prototype system is demonstrated for dynamic graphic design and animation. A designer can rapidly sketch a layout for an interactive page, using pressure and tilt to indicate the visual weight and translucency of text and images on the page and how these objects should change their appearance and behavior over time. Through the articulation of sketch marks designated to indicate motion (walking, running, etc), an animator can also use this system to produce an easily-modifiable animated scene of a 2D two-legged figure.

Applications for this type of work include interactive document layout, user interface rapid-prototyping, and sketch-based storyboarding for animation.

Thesis supervisor: Muriel R. Cooper, Professor of Visual Studies

This work was supported by Paws, Inc., and Hewlett Packard.

AN INTELLIGENT SKETCHPAD:

A Gestural Language for Denoting Temporal Behaviors in Dynamic Design

by Karen Donoghue

The following people have served as readers for this thesis.

William Buxton
Professor of Computer Science
Faculty of Computer Science
University of Toronto, Toronto, Canada

Patrick Purcell
Professor of Human-Computer Interaction
Faculty of Informatics
University of Ulster, Londonderry, Northern Ireland

Acknowledgments

Many people have contributed to this work. I would like to thank Professor Muriel Cooper for encouraging me and providing a computationally intensive environment in which to work on the ideas in this thesis. Thanks for all the sketches and layouts, too...

I would also like to thank:

Henry Lieberman, Ron MacNeil, David Small and Carlton Sparrell for feedback on this thesis.

Professor William Buxton from the University of Toronto for being a reader and for doing inspiring gesture research.

Professor Patrick Purcell of the University of Ulster for being a reader and for appreciating the blurred line between art and computer science.

Tim Bird of Paws, Inc. for support and for incredibly invaluable feedback and ideas on the topics covered in this thesis, encouragement and insight, especially with regards to the Intelligent Pencil Test system.

Candace Obert for her contributions to the software and hardware to support this thesis. Chip Goetzinger for contributions to the animation portion of this work. Thanks for being such great UROPs!

Thanks to my colleagues in the Visible Language Workshop, without whom this experience would not have been anywhere near as exhilarating: Amy Freeman, Maia Engeli, Jeffrey Ventrella, Craig Kanarick, Suguru Ishizaki, Louie Weitzman, Alan Turransky, B.C. Krishna, Tim Kukulski. Thank you Bob Sabiston for providing the BadWindows environment upon which all of this work is built and for encouragement.

I would especially like to thank Linda Peterson for sanity checks throughout this last two years.

Thank you Ben Davis of the MIT Center for Educational Computing Initiatives for discussions about art and visual computing.

Thank you Ken Knowlton for **inspiring** me.

Thank you Steve Levine for encouraging aggressive thought about pen-based interfaces.

Thank you Alex Harui and Michael Schirpke for teaching me about electronic pens.

And thanks to my friends Brenda, Rose, Maureen, and Jessika for support and encouragement...

I would especially like to thank my mother Albina and my father Robert for support and encouragement...

Thank you Mark Lucente. Your lucid and insightful thoughts and comments about everything have never ceased to amaze me.

Table of Contents

1. Introduction and Motivation

Sketching offers an intuitive form of data compression, one in which visual ideas can be quickly expressed. Paper and pencil allow for a rapid, simple and inexpensive means for communicating information which might otherwise take a long time to express in words. This is especially true with the communication of information which changes over time and whose spatial and temporal behaviors can be diagrammatically expressed. An example of this is football strategy notation, in which the spatial and temporal aspects of a football play can be described through a sketch (Figure 1.1). The marks left on the page are visual remnants of the gestures used to make those marks. Electronic sketching, however, now allows us to capture the *act* of sketching: speed, pressure, and tilt of the stylus are reflected by the input device. We now have an accurate record of the sketching process, similar to what one might get from watching someone sketch and describe the process simultaneously. As stylus-based computer input devices become better able to express the subtleties of human hand motion, a computer system should be able to watch and interpret the user's sketching process. The system should understand the syntactic, semantic, and temporal implications of sketched input derived from the *gestures* used to create the sketches.

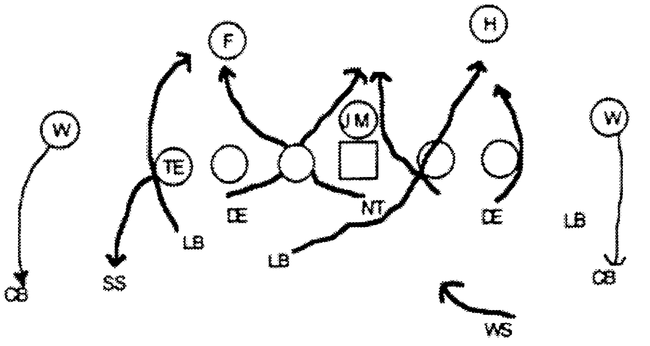


Figure 1.1: Example of football strategy sketch showing spatial and temporal aspects of a football play (From The Pragmatics of Haptic Input - CHI '90 Tutorial, W. Buxton)

Unlike old-fashioned paper and pencil, the electronic medium allows for a new type of "paper" which is no longer simply two dimensional but multidimensional. Since it contains the added dimension of time, it functions more as a "substrate" upon which marks are made and interpreted. There is a need for a new convention[†] which allows us to interpret these marks, and how to tell objects on this "page" how to behave over time, spatially, in relation to each other and to their larger context. For example, a sketched arrow can be considered to imply motion. Its physical appearance reflects only a few aspects of how it was created. Drawn quickly with a pencil and with very little pressure, this arrow appears to be very thin and light grey. It might be interpreted based on how it appears to mean simply: *move this object over there.* In the electronic medium, we know much more about how the arrow was made. We know the velocity and acceleration of the hand which made the mark, the subtle variations in pressure and tilt, and relationships between the arrow and other objects on the page. We now have a means for capturing *the design process.* If we define a language to interpret these marks according to criteria which are useful to us, we can exploit these subtle features and enable a faster, more expressive and more efficient means for translating information to the computer which would otherwise require tedious explicit specifications, mathematical formulas, and in the worst case, computer programming. Gesture, as interpreted through this language, offers a means for describing how objects should move and behave over time.

There are many conventional gestural languages. American Sign Language and the gestural conventions used by music conductors are two such systems. A richly expressive computer input device can reflect the activities of the user to a high degree of accuracy: consequently, computer systems should be able to interpret this information in useful ways. This thesis exploits the information communication by the articulation of sketches to better understand the design process, and uses this information in domains in which codified notational languages have existed previously only on paper, or not at all. A computer system which understands that the designer uses swift, short, light strokes when doing rough sketches for a page layout and uses firmer, more slow and steady strokes while drawing that same page layout closer to its final visual form. The understanding of this expressive input signal will enable personalized "design assistants" or "design agents" which are capable of mimicking the designer's sketching style or can fill in more detail when presented with only a sketchy outline.

After we understand the content of what is sketched, we can begin to build models which map behavior characteristics of what is articulated to what is understood by the

[†] This notational convention could be coupled to cultural conventions, such as the static notational marks used by graphic designers and proofreaders.

computer. We can do this for static drawings but we can also begin to prototype environments in which the designer must envision temporal events as part of the design process.

One domain in which temporal design is required is animation. As a first step in visualizing the motion of a character over time, animators typically create a pencil test on paper. In this pencil test, he sketches out a scene, creating a sequence of frames to indicate how a character should move through the scene. A computer system which allows the animator to use a pencil, a tool with which animators are already familiar, and recognizes temporal features of these annotations would allow an animator to quickly sketch a character in rough form, recognize it from a library of canonical characters, and add temporal annotations to key frames to indicate motion of the character over time. These temporal annotations are then interpreted and applied to the character to show motion or dynamic behavior over the course of a scene.

The proliferation of electronic newspapers and interactive books also provides a domain which requires the control of temporal behaviors of objects on multidimensional "pages". Methodologies employed by graphic designers provide a model for the way in which these new electronic documents will be designed. These methods include the use of rapidly-made thumbnail sketches and prototypes, in which concrete physical characteristics (such as font type and weight) are abstracted away to enable higher-level design issues to be addressed. Temporal behavior of the objects on these pages, such as the way an article "fades" as it's content becomes obsolete, can be thought of as another form of animation. These behaviors can be "roughed out" very rapidly using information implicit in the way the thumbnail sketches are made.

Sketch-based computer interfaces will be more useful as portable and hand-held computers become ubiquitous, requiring interfaces for which a mouse or keyboard is impractical. The need to communicate remotely using small portable wireless computers will validate the use of pen-based interfaces beyond mere substitution for keyboard and for handwriting in boxes. An architect visiting a building site might need to enter sketches into his computer system which denote dynamic behavior of traffic though an office park, a choreographer in a dance studio might annotate a virtual musical staff with applicable dance notations, and a cartographer might annotate a map showing some of the migrational behavior of animals of a particular geographic region over time. These interfaces need to provide the user with an electronic workspace which understands the contextual and dynamic implications of the marks made upon it, and should also reflect the expressiveness with which the marks are made. This interface should also exploit the knowledge that the user already possesses about the handwritten marks, as in the way that proofreaders' marks are understood both by editors and graphic designers.

Described in this thesis is a means for defining a gestural language to enable designers to express spatial and temporal behaviors of sketched objects, reflecting the fact that the user is communicating more information into the computer than by just writing on paper. Sketches are interpreted in terms of their geometric, spatial, temporal and gestural qualities (such as pressure and tilt) and used to control the dynamic behaviors of objects in a dynamic environment (Figure 1.2) This thesis demonstrates that traditional techniques employed in graphic design and animation can be used to control the behaviors of objects over time. Additionally, the notational conventions proposed in this thesis form the basis on which additional conventions can be built. As part of our graphic design prototype, our convention allows for an arrow to control the motion of an object over time. As these conventions become learned, they can be extended both to accomodate the more sophisitcated skills of the user, and to add notations which indicate higher-level control of objects, such as constraints among or grouping of objects.

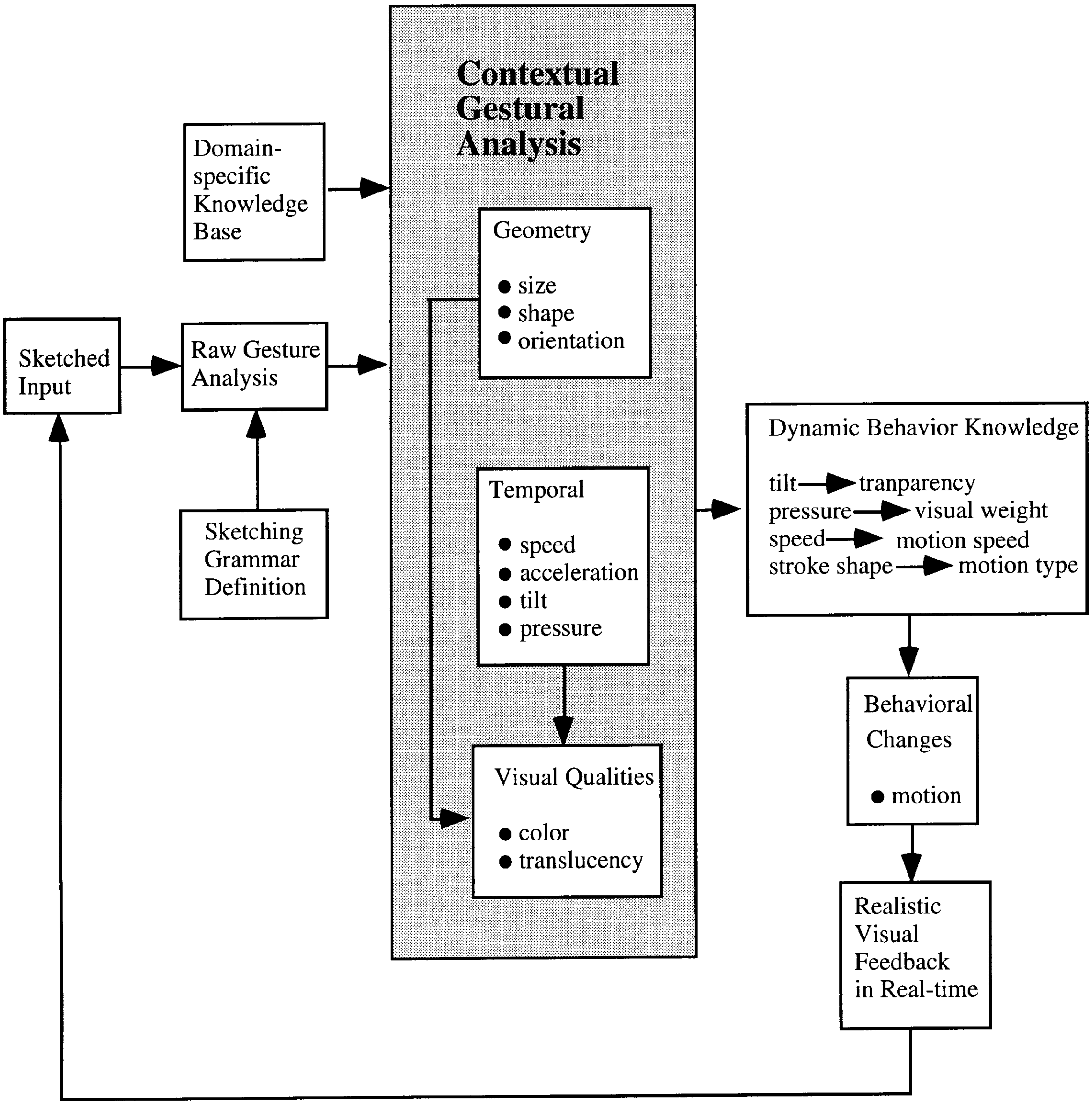


Figure 1.2: The components of the Sketching Substrate prototype system enable sketched input to be interpreted and mapped into motion control mechanisms through a sketching grammar . This sketching grammar considers the articulation of the sketched input relative to its contextual grammar.

The first domain in which we explore the use of a gestural sketching language is graphic design. Typically designers begin to visualize ideas by means of thumbnail sketches (Figure 1.3). Pencil and paper offer the most rapid and simple means of producing the first stage of design. At this point in the design process, the sketches represent loosely-defined ideas or "visual markers" which can stand for high-level representations of the page components. This abstraction away from detail allows the designer to focus on higher-level information in the design. Lower-level details such as typeface and size, though forming the foundation of the high-level of abstraction, are unnecessary at this stage of the design process.

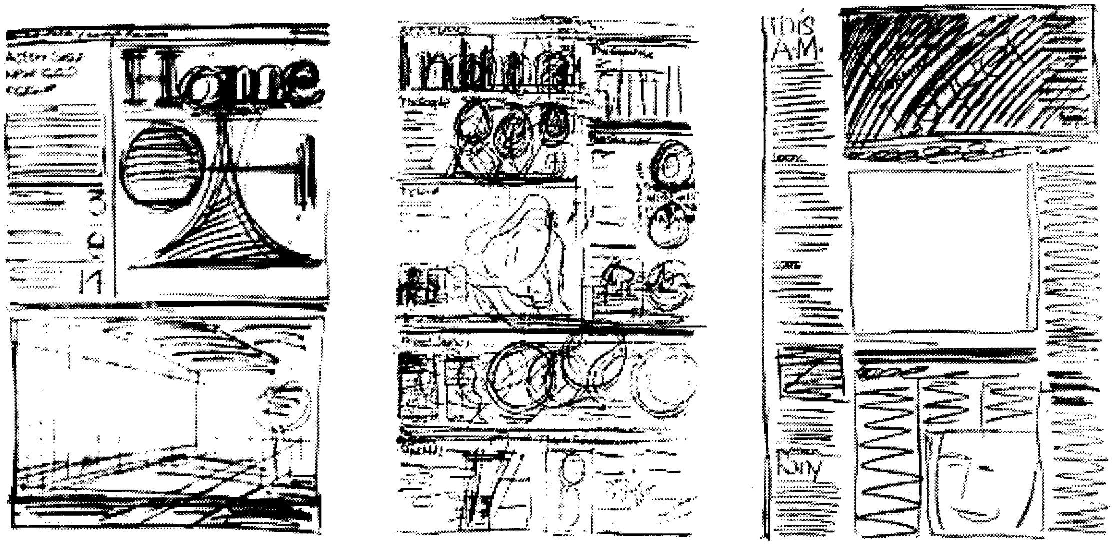


Figure 1.3: Example thumbnail sketches for page layout showing images and text. (From *Newspaper Design for the Times*, by L. Silverstein)

This provides an intuitive and fast means of capturing the design process and design solutions to visual communication problems. Instead of having to use a system which requires the user to choose tools for creating shapes and typefaces, the system, acting as an "interface agent", can interpret the visual qualities which the designer expresses through the sketches themselves. The sketches represent both geometric objects (such as images and pieces of type) and visual qualities of these elements such as weight, style, and point size which are respectively coupled to pressure, sketch type, and sketch size (Figure 1.4).

Having created a "page" of elements in this electronic environment, the system described in this thesis extends our static gestural sketching grammar to enable designers to employ a "rough sketch" methodology to visualize dynamic behaviors of these elements. By using sketching gestures, the system enables the designer to specify behaviors over time, specifying how the elements on the page should behave dynamically and qualitatively (Figure 1.5). In this way, we can communicate that a title should fade as it slides off of the page without having to use mathematical equations or programming. The change in visual appearance can be “roughed out” very rapidly using information implicit in the way the sketches are made. If the designer indicates that the behavior of the type object "Design" in Figure 1.5 should move to the right of the page while decreasing its visual weight or

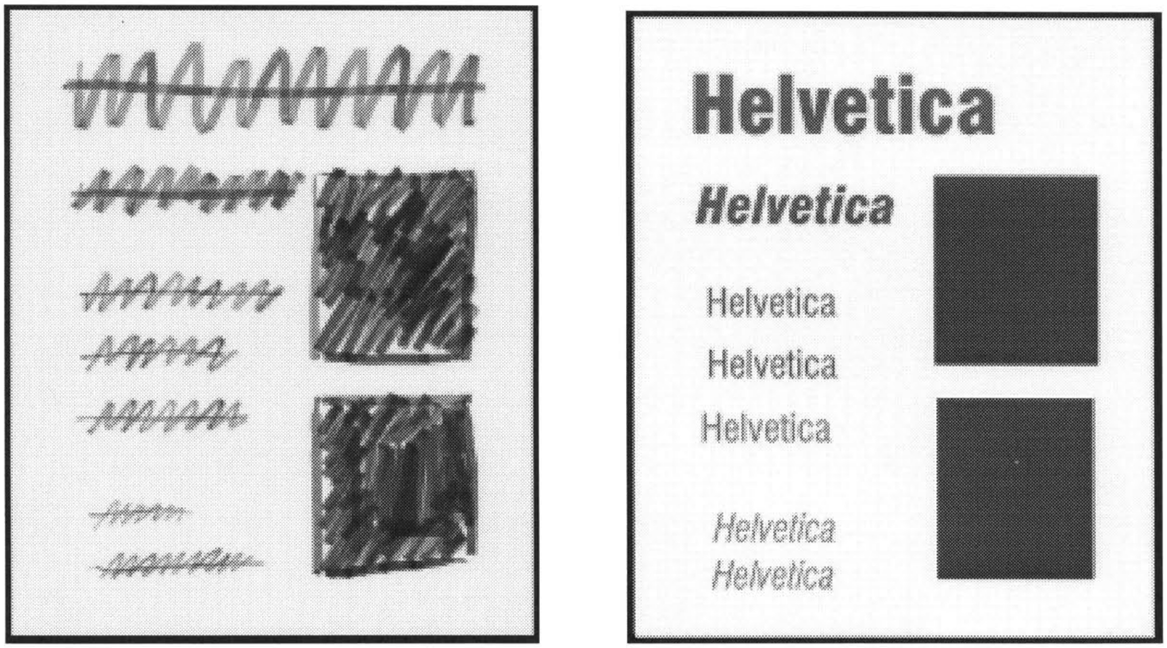


Figure 1.4: An example of a page layout sketch interpreted by the system. On the left is a thumbnail sketch made by the user with the stylus showing representations for text and image elements. The right page shows how the system interpreted these marks and replaced them with canonical forms of different typeface, weight and size based on the features expressed in the sketch such as pressure, tilt, and color.

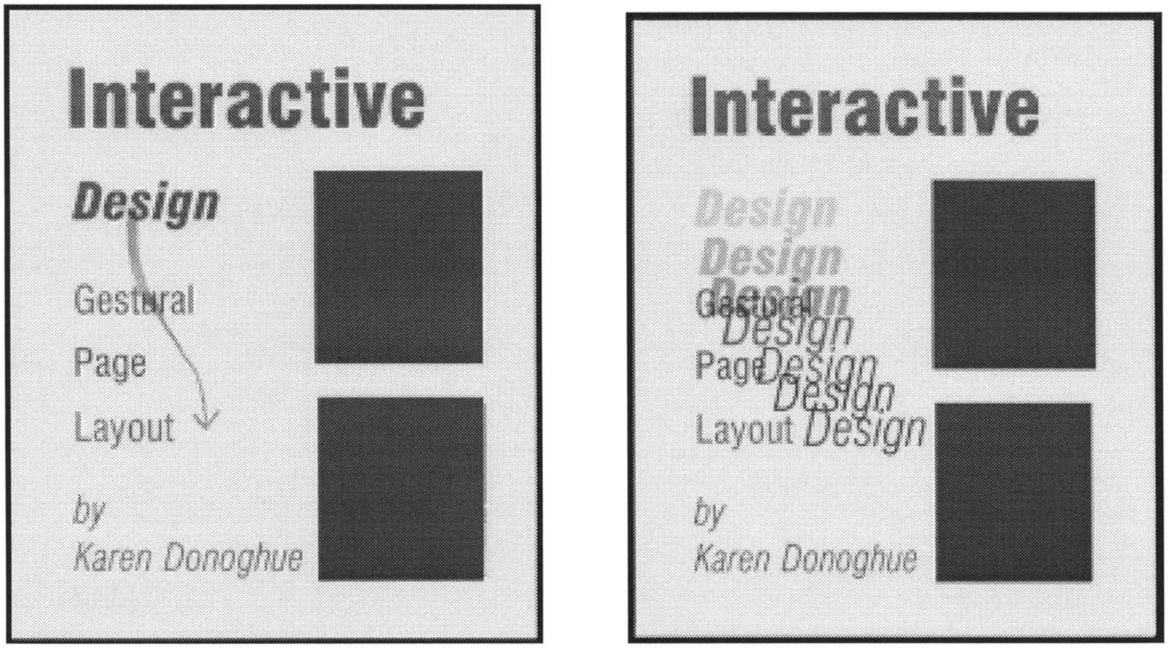


Figure 1.5: An example of interpretation of a sketching gesture to control dynamic behavior of a type object on the page. We have replaced the type forms from Figure 1.4 with actual text to illustrate a page layout closer to final form. On the left is the dynamic annotation sketch made by the user with the stylus indicating how the type element "Design" should move diagonally down and to the right of its original position, decreasing its visual weight and increasing its translucency as it moves. The control of transparency is determined by the amount by which the user tilts the stylus while drawing the arrow. The page on the right shows the interpretation of these sketch mark by the system and its effect on the layout.

"boldness", the system uses the articulation of the sketched arrow to indicate that behavior over time an associate it with the object description.

We could, with more knowledge about content, use this system to prototype interfaces which include dynamic components or temporal information. One example would be the first stage in designing an electronic newspaper or magazine. We could create a few rapid sketches (Figure 1.6), and after replacing the sketches with canonical type and

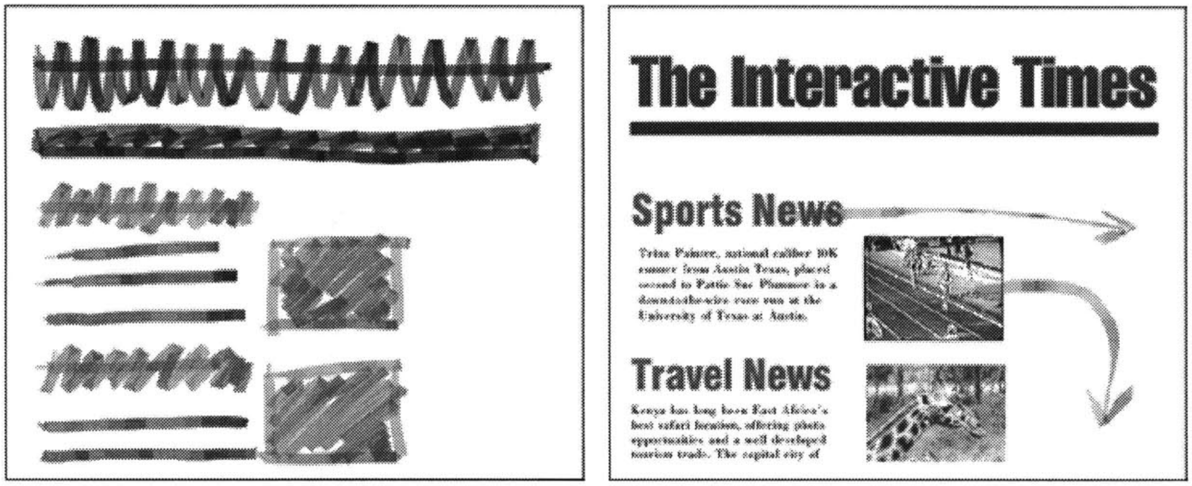


Figure 1.6: An example of how this system might be used to prototype the interface of an interactive newspaper or magazine. Sketched annotations at left are interpreted as canonical forms at right (we have added actual content for clarification) and sketched marks indicate how title title "Sports News" and the video clip associated with it should play out over time.

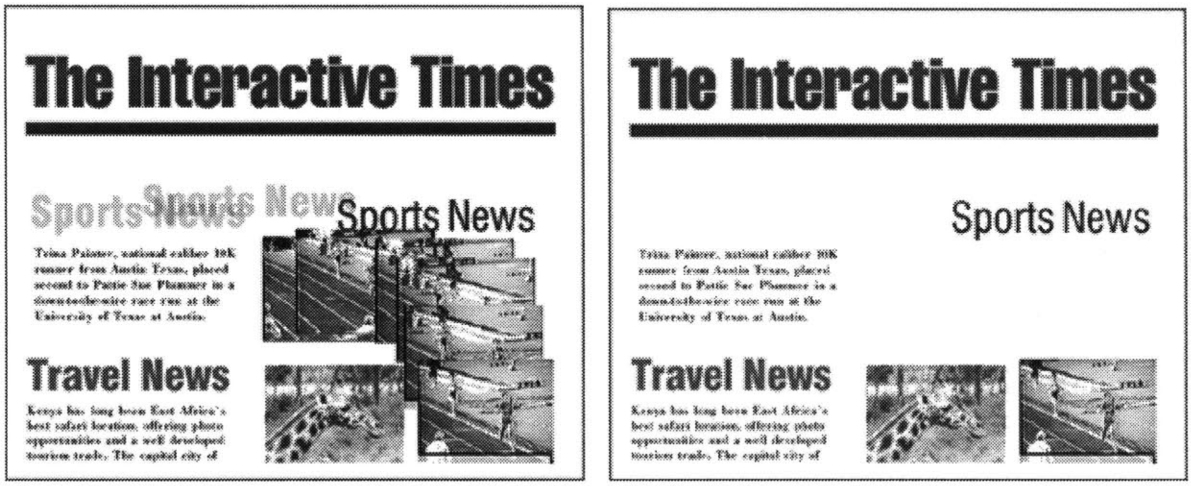


Figure 1.7: From the example shown in Figure 1.6, at left we see how the temporal annotations have resulted in the video clip being played out over time while following the path indicated by the sketch. Also note that the type element "Sports News" has changed its location and boldness based on how the sketched arrow was drawn in Figure 1.6.

image elements, indicate how we want these elements to behave over time on the page. Figure 1.7 shows how these annotations can be used to control the way in which the elements behave over time. Through sketching we can very quickly experiment with many design variations without investing alot of time and effort in creating the prototype.

A second domain in which we demonstrate our gestural sketch language is animation. Animation is a temporal event in which an object behaves according to a physical (or other) law and is represented by the renderer, which can be a human or machine. Viewing frames or "small temporal multiples" allows us to see static snapshots of this temporal behavior. Animators first start to visualize the dynamics of a scene by creating rough storyboards and pencil tests. The system described in this thesis attempts to enable for rapid and intuitive means for gesturally indicating how a character should move through a scene. Our goal is to enable the user to "rough out" a motion sequence through a sketch. By employing a sketching grammar which interprets geometric aspects of the sketched input as well as the temporal aspects of how the object should move, we control the motion of a two-legged figure in a 2D scene through the use of a sketching grammar which maps the user's input parameters expressed in the sketching act into useful control mechanisms (Figure 1.8). The model for interpreting sketching gestures is based on empirical data, derived from sketching samples[†] (which form the basis of the separate grammars for graphic design and animation) coupled with an inferencing mechanism. To prove this model, we have chosen a subset of four basic motion types. These motions include walking, running, jumping, and shuffling. The articulation of the gesture controls the rate at which the character moves, as well as the type of motion which is visually manifest as a result of the user's input sketch.

[†] These sketching samples included all sketching notations defined by the two demonstrations discussed in this thesis, captured using a software tool for recording speed, acceleration, tilt (X and Y), pressure over time. The samples were collected from the author and from two members of the development team.

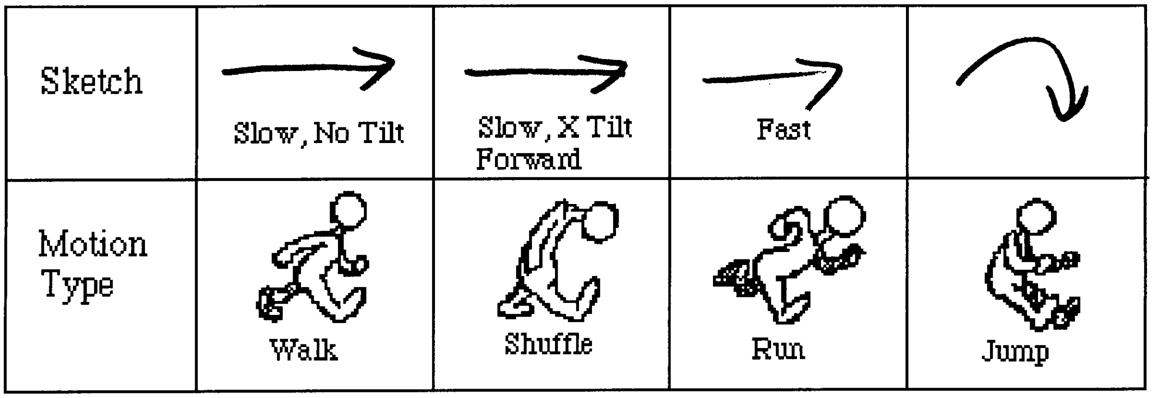


Figure 1.8: Relationship of sketching gestures to motion types in the control of a two-legged figure. Both the geometric aspects of the sketched marks as well as the articulation of the marks determines the type of motion performed by the character.

This thesis explores issues of expressivity in computer-human interfacing. Specifically, it seeks to answer the question: how much can a user communicate through sketching? Moreover, by knowing *how* these sketches were created, does this information provide useful and intuitive means for a method of capturing processes previously available only through pencil and paper? Is the fact that the paper is "listening" to the user's sketches and interpreting how they were made of any real use to us, and can we build a useful model to interpret, for example, the processes employed in the design of a page layout or in the early sketches of a blueprint for a house?

These issues begin to touch on deep issues of process and product, and are the impetus for the author's interest in gesture. Since computer input devices now can reflect the user's articulations to a high degree of accuracy, one could extrapolate from where the technology exists today to ask profound questions: if Van Gogh and Seurat had each used a computer system whose "brush" captured all of the nuances in tilt and pressure, velocity and acceleration and any other measureable or derivable quality, would this information result in a meaningful record of their processes, or would only the visual remnant, the painting, be the most useful product? Would this record of their artistic processes enable them to protect their work by patenting their articulations resulting from painting, since this information would represent the implementation of an idea, which defines the basis for patents? Would there be a way to interpret this information so that we could tell the processes of these two artists apart?

And what about the design processes employed by graphic designers? If we have a means for capturing rapid early sketches made by the designer, can we build an acceptable and useful model, perhaps a "designer's assistant" which can interpret sketchy designs into more canonical forms, perhaps even integrating knowledge of content in later design iterations? If animators had paper which understood the meaning of an arrow drawn quickly versus an arrow drawn slowly, would they begin to incorporate this type of electronic sketching tool into their everyday design processes? Would it enable them to visualize animated scenes more quickly, or would it hinder their creative process?

This thesis begins to explore these issues by presenting several prototype answers which show that for some design processes, especially those involving dynamics, sketching can impart rich, useful and important information which can be interpreted in both static and dynamic ways.

This thesis is arranged as follows. Chapter 2 provides background information and reviews related work by other researchers and other notational methods employed in different domains. Chapter 3 describes the approach employed in the interpretation of sketching gestures in a demonstration from the domain of Dynamic Graphic Design, and from Animation. Chapter 4 discusses the development of the software tools and lower-level routines necessary to implement the system. Chapter 5 presents results of the system and user-definable parameters. Chapter 6 discusses future work and applications. Chapter 7 concludes with a summary of the thesis. Chapter 8 contains appendices which give detailed algorithms. Chapter 9 provides a bibliography.

2. Background and Related Work

2.1 Principles of Dynamic Design

Dynamics is the study of the physics of bodies in motion, undergoing change over time. Graphic design deals with communication of information, with regard to its aesthetic and persuasive qualities. The use of the computer as a tool for simulating motion allows for a new type of design to support dynamics of objects which are sometimes represented metaphorically but many times have no analog in the real world. Hence, the term dynamic design encompasses the methodologies employed in the creation, presentation, and manipulation of information which changes over time. This chapter describes the wide variety of sources which inspires the work described in this thesis. These sources include methods for communication by notational and gestural means, research work in gesture recognition, visual languages, and animation, as well as commercially available systems which enable dynamic design.

There are methodologies which exist in seemingly disparate domains which we can integrate in the electronic environment to enable a new methodology for dealing with dynamic information. Animators have a richly developed means for communicating visual information over time [20], both in notational means (in the drawing of a walking cycle, for example) and in tools and techniques to allow for rapid experimentation with these notations. Pencil tests and thumbnails are tools which serve as motion prototypes: they can be quickly altered to change the motion characteristics of a character in a scene. Graphic designers use thumbnail sketches and prototypes to develop and test an idea in successive stages [37, 41]. Part of the strength of these methodologies is that they can result in prototypes which can be created very rapidly, at different level of detail, and can be "thrown away" if they prove to be unsuccessful during any stage of the design process. This allows for the visualization of ideas without the investment of a large amount of time and effort of full-scale production.

New computer input devices allow us to integrate the expressive and functional aspects of these rapid prototyping methods in an environment in which dynamic behavior can be controlled. With more expressive input devices we can begin to build tools which enable these rapid prototyping methodologies to become faster, more efficient, and perhaps better able to depict the final visual form more robustly. This thesis attempts to integrate gestural input with methodologies for rapid prototyping employed in the domains of animation and graphic design.

2.2 Traditional Dynamic Design Methodologies

2.2.1 Animation

Animation storyboarding offers example methodologies to control behaviors of objects over time. Typically, an animator creates thumbnail sketches by hand to quickly and efficiently summarize a scene, showing aspects such as motion [32]. These can be quickly assembled into a flipbook or recorded onto video and played back as a sequence at varying speeds so that the animator can get a "feel" for the motion throughout the scene.

To facilitate this process, there have been several computer systems which allow for the specification of motion by direct placement of the object, both in 2D and 3D. These include several commercial products such as Macromind Director[34] and Adobe Premiere[35] which require that each object's temporal behavior must be explicitly defined at specific points in time. Transitional behaviors of objects are controlled either by direct manipulation or by choosing "standard" behaviors such as wipes and fades from a menu. Symbolics has produced an animation system called S-Dynamics [36] which allows for automatic "inbetweening" of objects and tools for replication of sequences of time.

Several research efforts have addressed the problem of communicating specifications about how objects should behave over time. In a seminal paper on picture-driven animation, Baecker demonstrated a system for sketching characters and controlling their frame-by-frame motion by specifying a path drawn with a stylus [33]. The stylus sampled the user's notations at a rate of 25 points per second, and acceleration denoted in the articulation of the sketched path was reflected in the motion of the character. This system demonstrated a new approach to the specification of picture dynamics, one which exploited the use of a direct manipulation interface. It is from this research that I have derived much of the inspiration for attempting to use sketching as a means of controlling aspects of moving objects. Other work has dealt with the specification of more complicated kinematics, for the purpose of retaining realistic-looking motion. Sabiston [29] describes a novel method for extracting 3D motion out of 2D sketches by specifying a 3D stick figure which is "fitted" to the 2D sketch, allowing for maintenance of intrinsically 3D visual information, such as foreshortening. Librande [30] describes a means for using the Radial

Basis Function methodology described by Poggio and Brunelli [44] to intelligently generate inbetween frames from multidimensional key frame examples, with a notational sketch interface for recognizing emotional qualities of facial sketches designed by Donoghue [31].

2.2.2 Interactive Multimedia Design

Designers of interactive systems use several methods for indicating how objects should behave over time. One means is the use of storyboarding techniques to position objects explicity over time using "small multiple" [39, 40] views which show changes in position and other aspects of appearance laid out over time. Another means is to encode knowledge about an object and a set of rules which describe its behavior over some time period [33]. Yet another method is to explicity code behaviors over time, either as constrained by a set of dynamic equations [50] or constrained by behaviors of other objects [51]. The latter of these methods results in dynamic behaviors which tie the motion of one object to another.

Most multimedia environments require the designer to organize and temporally arrange input streams explicitly. To design an interactive magazine or newspaper [47, 48], a designer must define each page explicitly, each of the interactions on the page, its links to other pages, and the links to related visual objects. To create a very rough “demonstrable” prototype typically requires much effort and sometimes requires programming. A means for doing intelligent sketching annotation provides the designer with a way to do the preliminary design more quickly. It enables him to get a rough idea of the page dynamics by sketching out the layout for a page, its constituent components (title, paragraphs, images, etc) and the behaviors for these objects by using certain classes of annotations which are understood by the system to imply temporal behavior.

Rapid prototyping is an acceptable design methodology employed in the design of computer human interfaces [41]. Macromind Director [34] is a commonly used tool for creating these prototypes, but which still requires a scripting effort often in concert with programming.

2.3 Sketch-Based Interfaces

Sketch-based computer interfaces allow the user to interact with the system through a stylus and tablet. Since Sutherland [1] demonstrated the interactive Sketchpad system in

1963, there has been some work on the rendering of the expressive qualities of sketched input (known as "inking"), as transducers have improved to allow for faster data rates and more parameters describing user actions. Work in the area of expressive visual feedback for sketching include Bleser et al [2], who demonstrated a pressure and tilt-sensitive model of a charcoal sketch system in which tilt was used to alter the momentary "footprint" of the brush as it was moved over time. Donoghue and Knowlton [3] demonstrated a system which allowed for real-time pressure-sensitive kanji brushstrokes, the momentary footprint of the brush derived by making small, incremental changes to the shape of the brush based on the current state of the brush. This method bypassed the need for a lookup table of brush shapes and prevented unnatural "quantizing" of brush strokes. Ware [4] demonstrated a six degree of freedom bat brush as an input to a painting program. Commercial paint programs are now available [5, 6] which make use of pressure-sensitive input devices [52, 53] to give the user very realistic feedback. These systems allow for interpretation of the input information in terms of geometry which manifests itself strictly as visual feedback in the way the marks appear on the screen, giving the user a realistic reflection of what is being expressed through the input device. These input devices typically provide three degrees of freedom: X, Y, and a Z axis, in which force perpendicular to the writing surface is sometimes referred to as pressure. There is no interpretation of the contextual or temporal implication of the marks made by the user through the input device.

The input device described in this thesis [54] goes beyond what has previously been available in commercial paint programs, since it does provide interpretation of the marks which are made. We also have a more responsive input device which reflects, in addition to X, Y, and pressure, the tilt in X and Y expressed by the user during the sketching act.

2.4 Gestural and Line Drive Interfaces

Computer interfaces have incorporated several different methodologies for interpreting the gestures of the user in some useful way. Most of the work in gesture recognition converts continuous body motion into useful "chunks" of information. These "chunks" are used as input into a language which is of some use to the system. Typically these chunks are considered as static "tokens" in the language. The information implicit in the *articulation* of the gesture (such as the speed of the motion) offers a means for further exploiting gesture in interfaces, especially as transducers become better able to reflect body motion. Gestural interfaces in which some visual remnant (such as a line) of the gesture is produced are referred to as "line drive" interfaces.

Prior work in the area of gestural based interfaces has included recognition of handwriting [11, 12], proofreader's symbols [8, 14], and shorthand notation. Handwriting recognition is typically done as two-dimensional recognition, though higher dimensional recognition methods incorporating tilt and pressure have been investigated by Buxton et al. [7]. Buxton et al [10] also uses gestural annotation as a means for transcribing music notation. In this demonstration, the user points to a specific position on the staff and uses shorthand notation to enter a musical note. The gesture, used to indicate the musical note, indicates the type of note, such as a sixteenth note. Hardock [14] and Coleman [8] implemented proofreading systems, in which proofreader's notations were recognized by the computer and applied to text. Rubine [13] has demonstrated an example-based gesture system for defining and parsing single-stroke gestures, using a mouse. Other research has been done in the area of sketch recognition as a means for inputting flowchart information, mathematical equations [27], and circuit diagram information [26]. There have been several important demonstrations of machine recognition of freehand sketches. Herot [16] demonstrated the HUNCH system, an interactive sketch system which considered speed and sequence of the sketch act as a means of garnering information about the intent of the user. Kurtenbach and Buxton [17] demonstrated a gestural notation system for creating and manipulating simple geometric objects (squares, circles, and triangles) using shorthand and proofreaders'-type gestures, allowing the user to specify the command move, copy, and delete through hand-drawn symbols. This demonstration indicated the importance of not only what type of symbol is drawn, but where it was drawn, and how it related spatially to other drawn objects.

Zimmerman and Lanier et al [9] describe a three-dimensional approach to gesture interpretation using a data-glove, in which specific hand positions and their transitional motions are interpreted as controlling the viewpoint of the user over time. Systems have also been attempted which recognize static and motion gestures for American Sign Language (ASL) [22]. Fels and Hinton [18] demonstrated the GloveTalk system which recognizes a subset of ASL and generates continuous speech with control of stress and speed through gesture. Computer vision systems have also been demonstrated which recognize gestures from a live video source. The GEST system [19] recognizes hand gestures, which are used as control means in for a flight simulator and a graphics editor. Krueger et al demonstrated Videoplace [20], which does real-time processing of a video image of the user to recognize hand and finger gestures to manipulate simple geometric objects. Machover [45] employs custom-designed sensing technology by Gershenfeld [46] together with a cello to map gesture to musical result. Parameters used to derive gestural information include bow pressure, position and placement as well as finger and wrist placement.

The recognition of a continuous control signal, as in the articulation of a gesture, is the larger goal of this work. This involves extracting and interpreting aspects of the *way* in which a gesture is made and using these qualities in useful and expressive ways. One example which uses the interpretation of a continuous signal is the CANDY system [21], which allows disabled users to drive computer-based applications using the gestures produced by the user's best physical motion. The CANDY system employs a custom-tailored mapping from the analog control signal of a gesture to an appropriate control of an input device. The system allows a user who, for example, has only the ability to move his wrist towards or away from his chest to generate a control signal which can be translated and interpreted by the system for use in controlling a computer interface.

Many of these systems include gesture recognizers which need to be taught about the gestures which they will parse, requiring specific code which is difficult to generalize to other types of systems. The system described in this thesis is a generalized gestural "line drive" interface and will not require the user to teach the system about gestures, but will employ accepted notational conventions, if available, to provide intutive, real-time interpretation of sketched strokes.

2.5 Page Layout and Diagram Understanding

In designing the layout for a page, a designer first creates a thumbnail sketch showing title, main image, caption, and several paragraphs using the rough-sketch notation that graphic designers use to sketch static thumbnail layouts (Figure 2.1). By using more pressure and slower, more forceful strokes to create the image, the designer can gesturally indicate relationships of the image to the entire sketch (more importance, more visual weight, more focus).

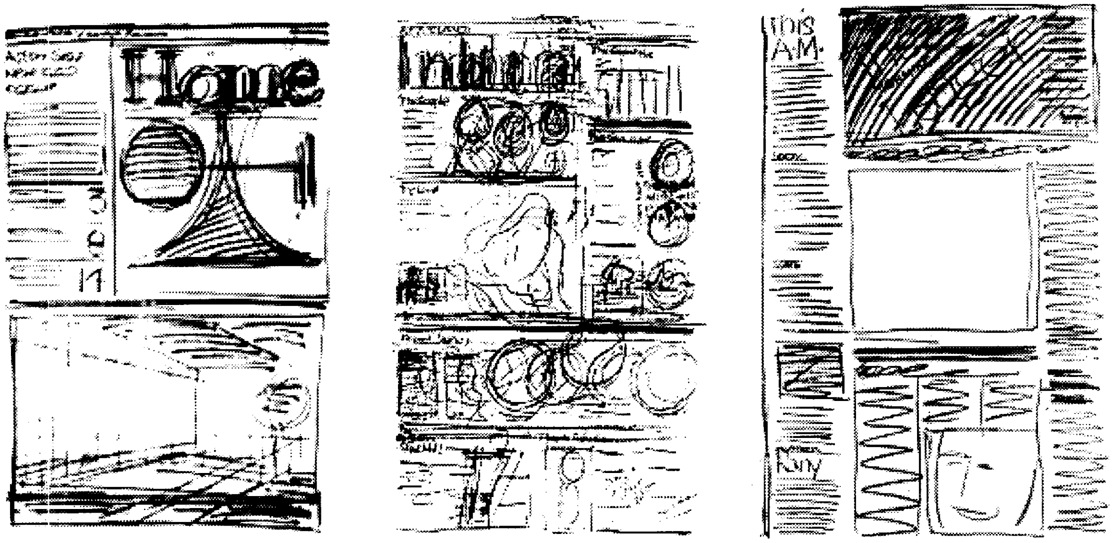


Figure 2.1: Example thumbnail sketches for page layout showing images and text. (From *Newspaper Design for the Times*, by L. Silverstein)

The process of understanding the implications of graphic input can be thought of as a recognition problem in which elements are recognized both for their syntactic and semantic content. There has been some effort that addresses the problem of understanding diagrams and graphical page layouts [23, 26, 27]. Diagram understanding offers a means for inputting information into a system which would otherwise require complicated textual descriptions. Understanding page layouts means generating rules about layout and understanding the implications of the layout content, as a means for capturing human design knowledge and encoding it in a computer system. Greenlee [24] demonstrated a sketch-based interactive system for inputting non-content layouts into a computer system, for later spatial and visual weight analysis. In this system, the sketched annotations were not analyzed for any qualities other than the bounding box of the marks made. Funt [25] employs a novel model of a parallel processing "retina" to look at geometrically-based diagrams.

2.6 Visual Languages

Visual languages incorporate non-textual elements and allow for a visual means of communication in which meaning typically does not depend on absolute positioning, size, or orientation, but instead relies on relationships between components. These relationships include aspects such as relative position and containment, and divide the field of visual languages into three main categories: network depiction, topological relationships, and geometric relationships. Flowcharting is an example of a visual language which uses network relationships: geometric symbols are defined by lines which denote connecting relationships. Venn diagrams represent an example of a topological visual language, and mathematical equations represent a class of visual language in which geometric relationships are employed. The field of visual languages is a relatively young one, but it has become important in the area of user interface design since the advent of graphical user interfaces and use of visual metaphors such as that of the Desktop Metaphor.

The ability of a computer to recognize visual languages depends on its ability to "parse" diagrams and drawings. This will become important as notepad computer interfaces become more common. Helm et al [26] have presented new techniques for building visual language parsers based on constrained set grammars. Unlike traditional string grammars, in which the adjacency is the only allowable relationship between tokens, constrained set grammars allow for a very rich set of relationships between objects such as geometry and topology which are employed in extracting salient meaning from diagrams. Wittenburg et. al [27] present a unification-based grammar for parsing two-dimensional input such as handsketched mathematical expressions and structured flowcharts.

2.7 Other Notational/Gestural Systems

There are codified notational methodologies which exist in many domains. Proofreader's marks [37] are recognized by editors and graphics designers. There are also several notational methodologies for denoting temporal and spatial behaviors, including dance notation [38], and less codified methods for rapid prototyping for multimedia user interface design [41]. American Sign Language (ASL) is a gestural visual language in which the discrete hand positions and motions are important in the communication of information [22]. Changes in features of the movements of the hands in space (dynamics of movements, manner and direction of movement) indicate morphological differences in the language. Conductors also employ a codified gestural language in the directing of orchestral music.

2.8 Discussion

This work draws inspiration from all of the methodologies and research described above, and represents a system which is both a gestural and line notation interface. The visual language is defined for the specific application domains. A goal of this thesis is to explore aspects of continuous gestures in controlling dynamic behaviors of objects such as text on a page or characters in an animation. To describe the path of an object over time can be one means of describing motion. This can be an equation, an explicit assignment, or done by analogy. In our grammar, an arrow originating from the object is considered indicative of motion, yet the way in which the arrow is made conveys meaning.

3. Approach

3.1 A Gestural Visual Language

We use gesture to communicate information: in everyday conversation, we use hand gestures as reinforcement for what we are saying. Sketching involves a form of gesturing in which the articulation of the gesture leaves some form of a visual remnant, typically a mark on a page. The system described in this thesis interprets and uses these sketching gestures to enable the rapid expression of visual ideas, especially those that involve dynamics. The system also provides the user with an intuitive and informative graphical representation of the visual remnants left by the gestures intrinsic to the sketching act, conveying useful information about how these marks were made. This chapter presents an approach to interpreting sketched marks as control mechanisms for the motion of 2D objects. The first section describes the overall model used to interpret sketched marks. The second section explains how such a model can be employed in the context of dynamic page layout. The third section describes the model as a means for controlling the motions of a 2D animated figure.

This research proposes a new gestural visual language tool for animating 2D objects. This gestural language is based on a domain-specific grammar for interpreting the articulations of sketched strokes, and used to denote temporal and dynamic behaviors of objects in an electronic environment. This grammar incorporates aspects of the sketching act such as speed, pressure, and tilt of the input device. Aspects of the notational interpretations described in this thesis have been employed in other gestural interface demonstrations, including the annotation of maps (Figure 3.1), the editing of video segments in a multimedia system (Figure 3.2), and in the interpretation of the emotionally expressions of a sketch of a face. This facial sketch is used as an input means for browsing a multidimensional visual database for animation (Figure 3.3). In the map annotation example used for tactical planning, sketch marks (circles, triangles, etc.) are replaced by canonical symbols which stand for tanks, platoons, etc. The system knows how each of these objects behaves dynamically, and when given a sketched path to follow, the symbols attempt to follow the path as constrained by their own behaviors (tanks, for example, are non-amphibious and so need to find alternative paths to avoid water). The path indicates where the objects should go, as well as how (with appropriate speed and acceleration) through the articulation of the sketched path. The video editing example uses simple gestures to allow the user to assemble digital video segments into a finished piece.

The visual database browsing example has a multidimensional description of facial expressions, and through the use of a rough facial sketch, the user can extract finished line art of faces which are generated on-the-fly by emotionally expressive aspects of the sketch. Features such as the amount of "surprise" or stretch of the eyeballs, the "angry" slant of an eyebrow, or the position of the pupil relative to the eyeball are extracted from the sketch and used as input into a multidimensional visual database. A Radial Basis Function is used as a means for intelligently interpolating along each dimension [30, 44].

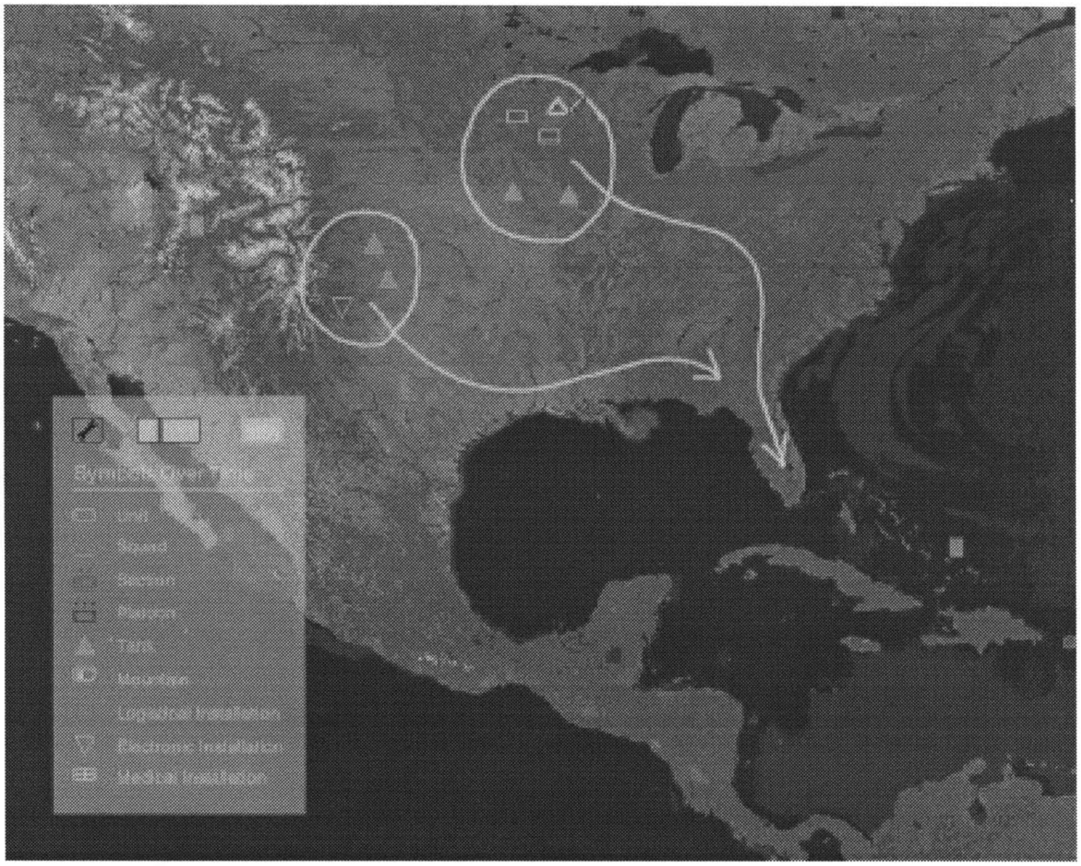


Figure 3.1: A map annotation prototype. The user sketches symbols on the map which are recognized and replaced by canonical forms (circles, triangles, etc.). These symbols represent different types of objects with specific behavior patterns in the system (tanks, platoons, etc.). By gesturally indicating a path for a group of objects to follow, each of the objects tries to follow the path while being constrained by its own motion behavior. The sketched path indicates where the objects should go, and the articulation of the path controls how the objects should move, controlling such qualities as speed and acceleration. By Karen Donoghue, MIT Media Laboratory, 1992.

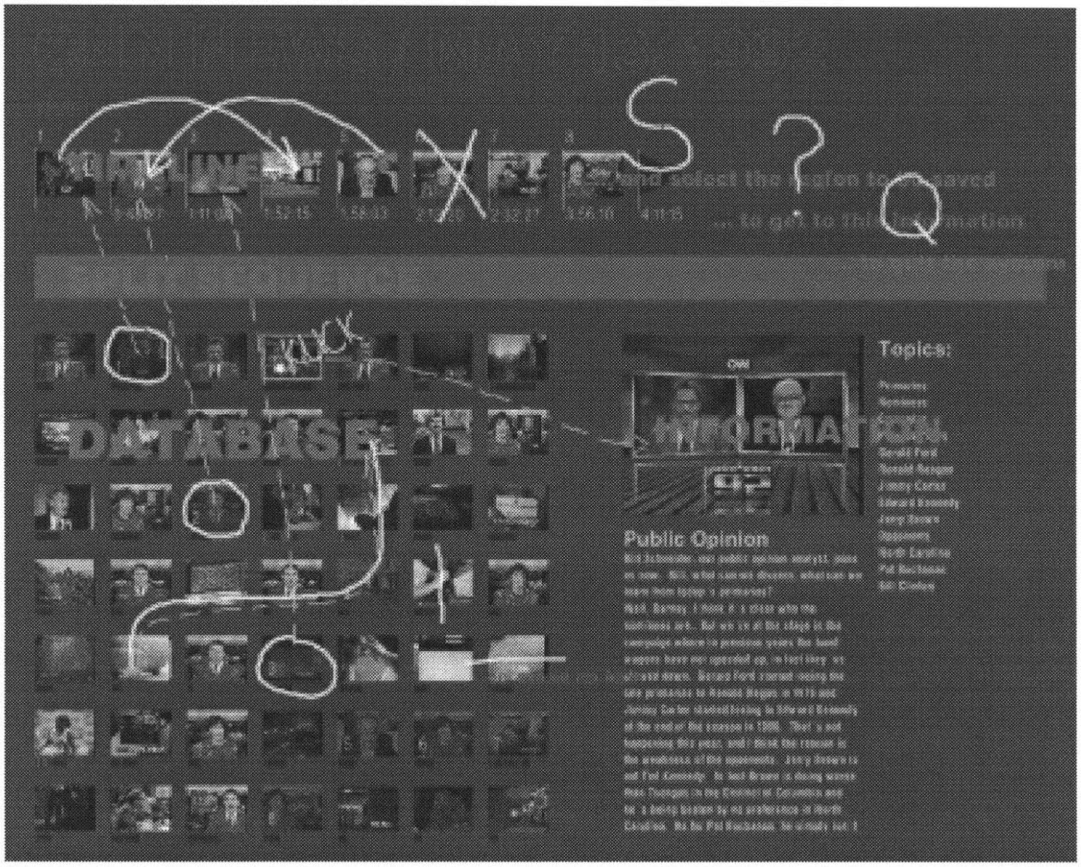


Figure 3.2: A gestural video editing system. The user can create a finished digital video sequence by using sketching gestures to assemble and edit segments of digital video. (Video editing system designed by Maia Engeli, MIT Media Laboratory. Sketching gesture interface software by Karen Donoghue, MIT Media Laboratory).

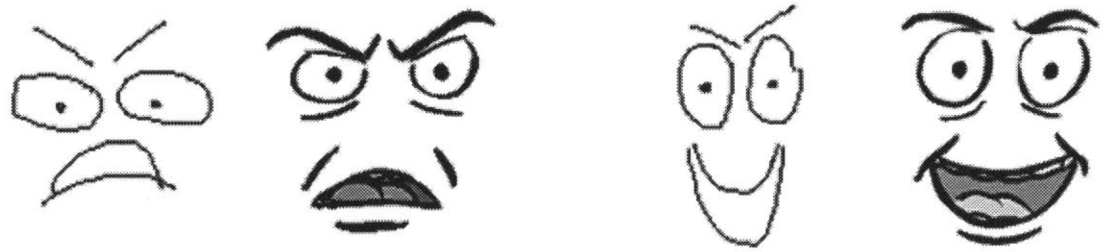


Figure 3.3: Example results from a visual database browsing system using sketches as a means of querying. The user sketches a face which approximates the emotional expression for which he desired a canonical face. The system interprets the sketch in terms of its emotional qualities, such as surprise, anger, or sadness and interpolates through a multidimensional description of canonical faces to generate a best-fit face for the sketch. (From XSpace system by Steve Librande, M.S. thesis, MIT Media Laboratory, 1992. Sketch interface written by Karen Donoghue, MIT Media Laboratory.)

3.2 The Intelligent Sketching Substrate Model

Much of the pen-based user interface software currently available uses the stylus as a mouse substitute. That is, the stylus is employed as a point and click input device for pressing buttons and toggling switches, similar to using a pen to type at a keyboard. Handwriting recognition exploits the size and shape of the stylus as a tool for allowing naturalistic handwriting, but when constrained by "boxed" input fields, this input means begins to feel constrictive to the user. Gestural systems which include a stylus as an input device allow for freehand input, which is more naturalistic for some types of applications. This thesis provides an extension to a direct manipulation interface which is based on gesture: the user can *show* the system how the object should move by demonstrating it though a sketch. The sketch can have grammatical meaning (such as the type of character) or it can have temporal meaning, such as speed and acceleration. This type of system requires that the underlying user interface software be more "intelligent" in its processing of the user's input strokes. It can be thought of as a piece of sketch paper which can "observe" and "feel" how the sketches are made, and interpret that information according to the rules of its application domain.

This intelligent piece of sketch paper can be though of as a substrate. A substrate is a surface which serves as the basis for processes which act upon it. This thesis demonstrates an electronic sketching substrate upon which marks are made and their implications interpreted as control mechanisms. To accomplish this, this system applies a translucent intelligent "sketching substrate" to the drawing area (Figure 3.4). This substrate can be thought of as a non-visible window over the sketching area: in this window marks are captured and analyzed based on the *way* in which the marks are articulated to the computer. Hence this substrate considers not only the remnants or marks resulting from the user's gestures, but the gestures used to create the marks. The substrate interprets the marks made upon it according to the its currently defined contextual grammar.

One major goal of the system is to provide a highly interactive and intuitive environment for communicating temporal behaviors of objects to the computer. Thus, the system has been designed to run in real-time, and with very expressive, realistic user feedback. Subtleties in the hand motions of the user are reflected in the visual feedback and in the interpretation of the input, such as in the use of antialiased lines whose line weight changes in relation to pressure, and whose translucency changes in relation to tilt.

Our model also considers realistic aspects of the sketching act. Stopping and hesitating are two behavioral aspects of sketching which provide information about the intent of the user. In our gestural language, we use "hesitation" to imply that the user is finished with a series of sketch marks and that together these marks constitute a "group" of sketch marks which carry meaning when considered against the grammar we have defined for our system. In this way, our gestural language employs a "temporal delimiter": the input information is broken into useful chunks of information which are separated by time. Other models might employ such delimiters as spatial arrangement and containment, as is typically employed in sketching recognition systems which manipulate symbolic languages [26, 27]. We chose this method because it most closely simulates the natural pacing and rhythm of the sketching act, providing an interactive dialogue, and does not require any post processing of input strokes.

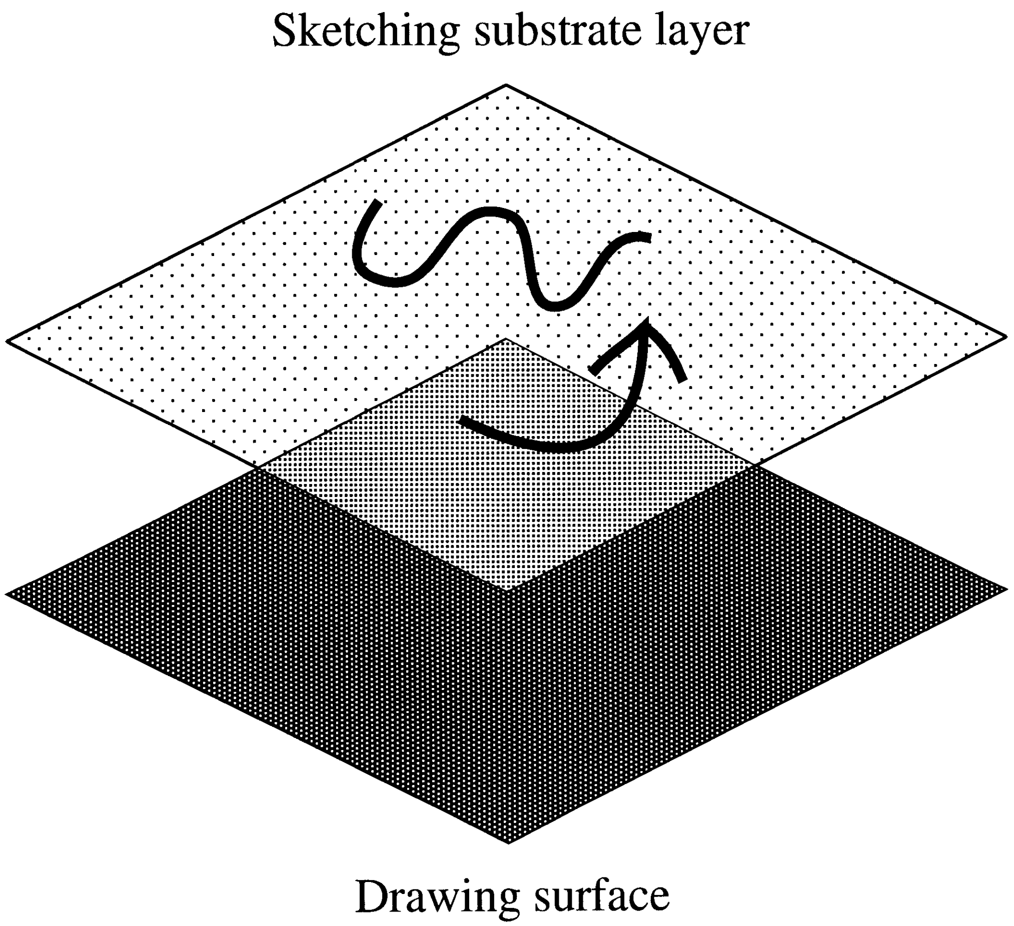


Figure 3.4. The sketching substrate can be thought of as a transparent interpretive layer above the drawing area. The user sketches into the drawing area, and the marks are interpreted by the substrate layer.

The Substrate Model employs a series of directed graphs which indicate the spatial, qualitative, and temporal state of all of the objects actively known to the substrate. These graphs are depicted in Figure 3.5. The Spatial Graph (S) contains spatial information about an object, such as its size and position on the page. The Qualitative Graph (Q) contains information about what the system considers to be the current qualities of the object, such as color and translucency. The Temporal Graph (T) contains the information about temporal annotations (such as arrows) which indicate how the objects should behave temporally. By processing the elements in the Spatial Graph based on the Temporal Graph, we update both the Qualitative and Spatial Graphs. This means that the "script" which shows the dynamics of the page are stored not as frame-by-frame changes but as a set of graphs whose interrelationships determine the behaviors of the objects over time.

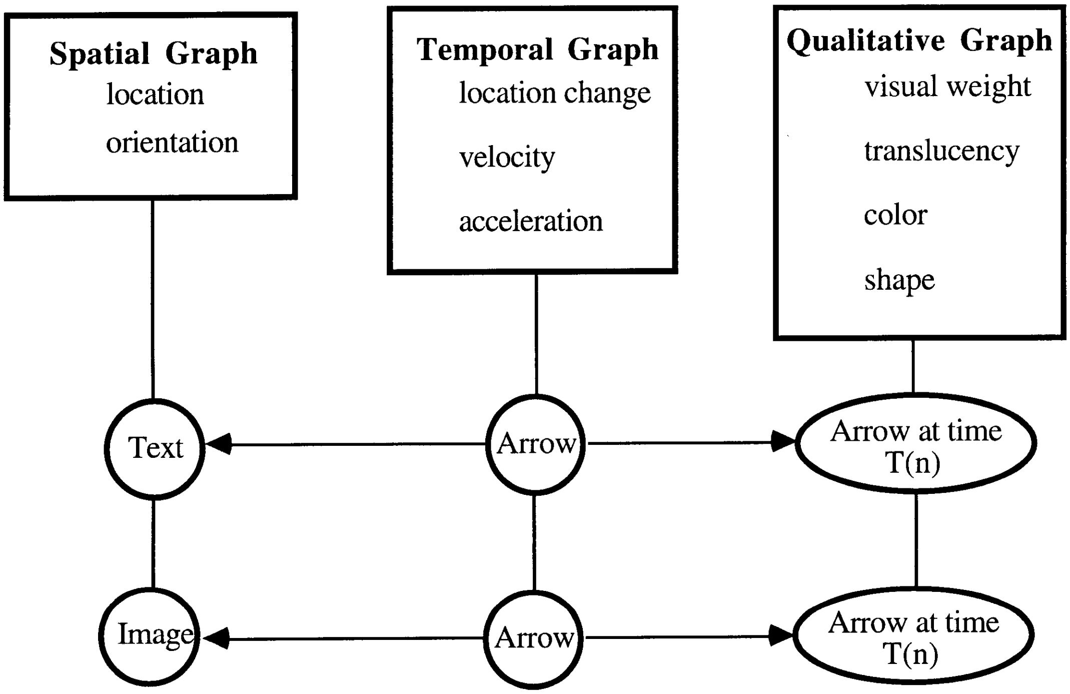


Figure 3.5: The Spatial (S) graph, the Temporal (T) graph, and the Qualitative (Q) graph.

The interpretation of the strokes is based on a feature extraction algorithm which discerns geometric qualities of each stroke. These qualities include orientation of lines (horizontal, vertical, etc.) and simple one-stroke shapes such as circles. If several strokes are sketched within some threshold time limit (the stationary dwell time), these are considered to constitute a group of strokes which can form more complex objects (such as rectangles, arrows, etc.). The use of stationary dwell time as a means of grouping during the interpretation of the sketched input implies a grammar whose syntax relies on temporal delimiters. This assumes that the user finishes drawing an entire group of sketch marks which constitute an object, and that the dwell time of the stylus (in the air above the tablet) does not exceed the threshold time limit. The threshold limit is implemented as a user-definable parameter, though users have expressed discomfort when the threshold is set below 800 milliseconds. Based on user trials[†] , the best values for the time threshold are from 800-1100 milliseconds.

Like any language, this sketching grammar is comprised of small tokens (strokes) which are organized into groups (phrases) which are delimited by time. Hence we consider stationary dwell time as the delimiter between phrases. The way in which the sketched strokes are made are the descriptive components of the language. Qualities such as pressure, tilt, and derivable measurements such as speed and acceleration are the modifiers and cannot exist separately from the strokes, unlike adjective and adverbs, which can be considered as free-standing tokens of language. The modifiers in this language exist as temporal events, and cannot be viewed or experienced statically.

3.3 Sketch-Based Dynamic Layout: The Dynamic Page Model

The Dynamic Page is a demonstration which allows for rapid realization of visual prototypes. These prototypes can be either static, as in a page layout, or can be designated to be interpreted as "temporal maps" which indicate how the objects on the page should behave over time. These prototypes can be entered quickly and easily using a stylus and tablet, and the user creates these sketches using the same hand motions and muscles which he would use to sketch a design if he were to use paper and pencil. There is no need for the designer to learn how to use a new input device or a complicated syntax for representing his thumbnails.

The mapping from the user's input parameters expressed in the sketching act is shown in Figure 3.6. The syntax has been kept intentionally simple so that the user does not have to remember many symbols which represent information: the syntax is based on the "visual language" that graphic designers use in creating thumbnail sketches. Sketching page layout components result in the sketched marks remaining on the screen.

[†] User trials included sketching sessions by the author and two novice individuals not involved with the system design but familiar with stylus input devices

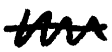
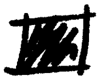
Object	Sketch	Boundary	Pen pressure	Pen tilt	Slant of sketch	Frequency of sketch
Text		Size and Shape	Visual Weight	Trans-parency	Italics	Condensed (Stretched)
Image		Size and Shape	Visual Weight	Trans-parency	Direction (Texture)	None

Figure 3.6: Relationship of sketched input features to text and object physical qualities.

If the user wants to consider the thumbnail sketch at a more detailed level, he or she can choose to have the sketched components replaced with "canonical" text and image forms. The visual qualities of these canonical forms, such as font weight, italics, transparency, etc., is based on how their representative sketches were made. To indicate that a page title should be a heavy element on the page, the designer might use slow, heavy sketch strokes in sketching it. The system "records" the way in which the marks were made, and analyzes the gesture against its grammar to figure out the type of mark which is being made and the physical qualities of the mark. The algorithms for parsing the sketch component and for mapping it into a canonical form are given in the appendix in Chapter 8.

After viewing these sketches as canonical forms, the user can gesturally control their visual appearance. By pointing to a piece of type and applying more pressure to the stylus, the user can increase the "weight" of the type. To change translucency, the user can change the visual weight of a piece of type by tilting the stylus in the positive or negative X direction. Tilting the pen to extreme angles results in increased translucency. A stylus position perpendicular to the tablet causes the type to become completely opaque. We employ a continuum of translucency values in the range of 0 to 255. We can easily change the grammar which maps behavior to visual characteristics. For example, we could use pressure to control translucency, or point size. By extending our gestural grammar we could also allow for the use of different fonts, so that serif fonts are represented as different types of sketch marks. A system which allows for this type of rapid input for rough sketches could, with some modifications, be extended to employ an intelligent rule-based grammar to discern spatial relationships, such as the type which stands for the title, the captions, etc. This would allow for the introduction of content into the design, so that the layout can be viewed in a near-final form and can be modified while considering the actual words and images which need to be placed in the final layout.

The grammar employed to interpret the sketches representing text and image objects can be further augmented to imply other visual qualities of the type. For example, by adding "serif" marks at either end of the sketch mark which represents a piece of type, we can tell the system to consider the type as being a serif font (Figure 3.7). Future work could extend this grammar to make provisions for different families of serif fonts (e.g. Baskerville) whose sketches might reflect some aspects of the typeface's visual weight or serif to stroke relationship.

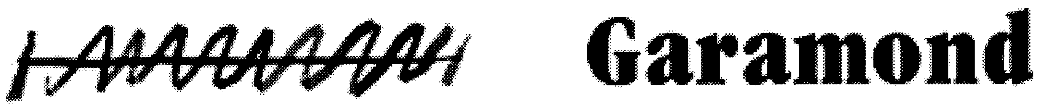


Figure 3.7: Sketch mark on left represents a serif font text object. The marks at either end of the horizontal line distinguish the sketch from a san-serif text object. On the right is a canonical serif type form which represents the sketch at left.

After the user has input the static sketch, we can proceed to visualize how these objects on the page can be controlled over time. This can be thought of as a quick motion study, as typically used by animators. In this case, we are using this method to rapidly indicate the way in which we want the elements on our dynamic page to behave. For example, if we are designing a page of an interactive magazine [47, 48] and want to get a sense of how a piece of type might change its visual appearance and location over time, we can gesturally indicate how this behavior should be expressed based on our sketch. We refer to the sketch marks which indicate how objects should behave over time as "temporal annotations". Much as in the way that the static thumbnail sketch components are recognized for their visual qualities based on how the user makes the sketch, the temporal annotations gesturally indicate how the object should change over time. These temporal annotations are shown in Figure 3.8.

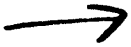
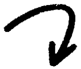
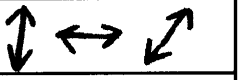
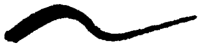
Change to Object	Sketch
Location	
Orientation	
Scale	
Visual Weight	

Figure 3.8: Relationship of temporal gestures to physical qualities of text and object .

In Figure 3.9, we see the results of this sketch process, its interpretation as canonical type forms based on sketch articulation, and further temporal annotations indicating how a piece of type should move and change its visual qualities over time (Figure 3.10). In Figure 3.10, we see that the canonical forms of type have been replaced by content forms, so that we can envision how this final layout will appear. The system described in this thesis does not include any intelligent provisions for incorporating content into the layout, but this would be a logical step for extensions to this system. This would include the integration of a rule-based spatial relationships between objects in a layout so that actual content could be visually realized. Additionally, this system might prove useful as an input means in a unification-based grammar system to capture stylistic aspects of the layout, as is currently being researched by Wittenburg [27].

Much as there is an implicit notion of "time" as shown in the football play diagram shown in Chapter 1, so too is there a means for temporal control in this model. Time can be interpreted according to any of several rules: these can involve the order in which the marks were made, their visual weight, speed, or color. This allows the system to generate its dynamic visualization on-the-fly, based on the current sketch.

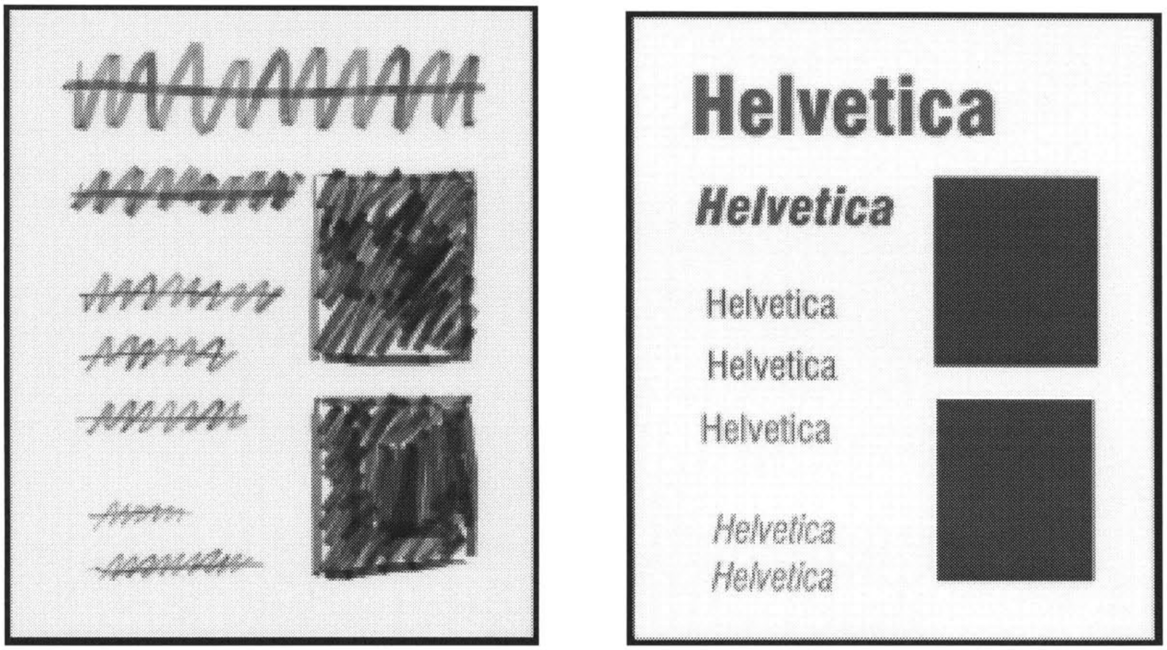


Figure 3.9 An example of a page layout sketch interpreted by the system. On the left is a thumbnail sketch made by the user with the stylus showing representations for text and image elements. The right page shows how the system interpreted these marks and replaced them with canonical forms of different typeface, weight and size based on the features expressed in the sketch such as pressure, tilt, and color.

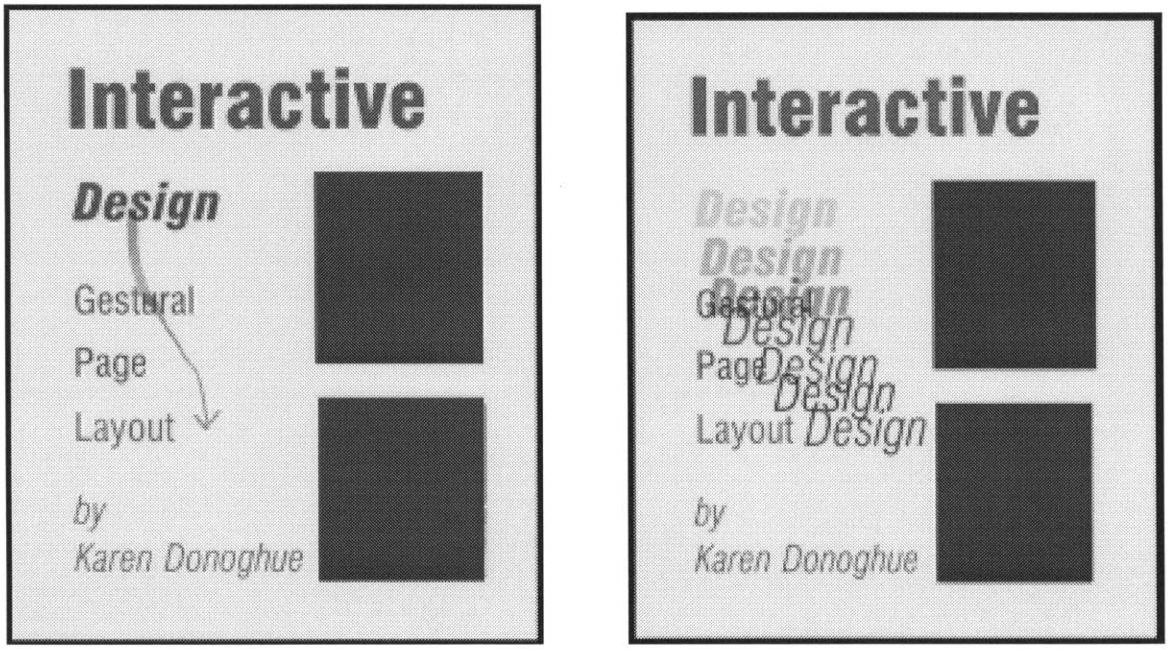


Figure 3.10: An example of interpretation of a sketching gesture shown at left to control dynamic behavior of a type object "Design" on the page at right.

3.4 The Sketch-Based Animation: The Intelligent Pencil Test Model

The goal of the Intelligent Pencil Test is to allow a user to create an expressive animated scene simply by sketching. Since animators first start to visualize the dynamics of a scene by creating rough storyboards and pencil tests, our system attempts to enable for rapid and intuitive means for gesturally indicating how a character should move through a scene.

The grammar which maps the user's input parameters expressed in the sketching act into useful control mechanisms is shown in Figure 3.11. The model for interpreting sketching gestures is based on empirical data coupled with an inferencing mechanism. From this data we have built a grammar for interpreting sketching articulations for controlling the motion of a two-legged figure. To prove this model, we have chosen a subset of four basic motion types. These motions include walking, running, jumping, and shuffling. The articulation of the gesture controls the rate at which the character moves, as well as the type of motion which is visually manifest as a result of the user's input sketch. The overall control mechanism for the demonstration is shown in Figure 3.12. Figure 3.13 shows the model's mechanism for interpreting sketched input in terms of the type of motion it represents.

All characters move relative to a background. In order to set the character in the scene, the user must specify where the character should be placed relative to the background. The system provides a means for doing this interactively. To determine the duration of time between two keyframes, the user enters a number greater than 0 into the appropriate Inbetween Timing window. This number represents the number of seconds during which the events specified in the appropriate keyframe are “played out”. The rate at which the character is moving (based on stroke qualities) and the type of motion also controls the rate at which the background moves. The user sketches a rough representation of a character which is recognized by the system as being a canonical character. Our current implementation recognizes two joined circles as representing a 2D man. By making sketching notations as indicated in Figure 3.11, the user gesturally indicates the type of motion which the character should perform relative to that frame. The system provides feedback to the user as to where the character appears relative to the background after the character has moved.

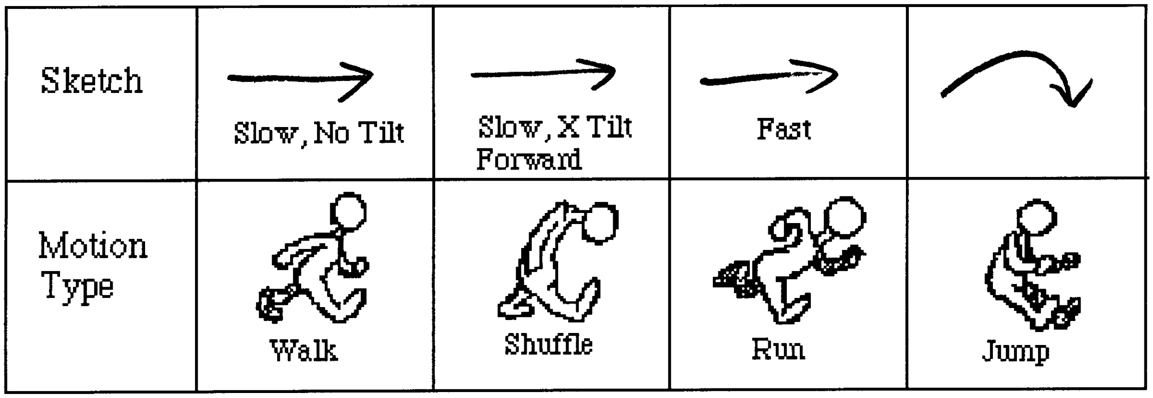


Figure 3.11: Relationship of sketching gestures to motion types in the control of a two-legged figure. Both the geometric aspects of the sketched marks as well as the articulation of the marks determines the type of motion performed by the character.

Sometimes, an animator wants the movement to slow down or speed up between two key frames. In this case, the positioning of inbetween frames is in time is not a linear interpolation between the two key frames. Animators typically use a chart which shows the positioning of each inbetween between the two key frames. A larger number of inbetweens results in faster motion. Conversely, fewer inbetweens results in slower motion. Our system allows the animator to sketch a line which represents the type of motion between key frames, namely a function which denotes the interframe dynamics. In this way, the user can control the movement between two key drawings simply by sketching in the function which controls the interpolation.

One problem which results in the control of animated characters is the notion of time, especially the duration of scenes and the number of inbetween frames between key frames. To solve this problem, the demonstration has included the use of explicit means for denoting the number of inbetweens by use of a slider. Additionally, we use color and sketch rendering to show the temporal aspects of the sketch components which denote motion. Sketched lines which are made slowly are shown as blue, and lines which are made quickly are red. Any intermediate speed is shown as purple. For example, an arrow drawn quickly results in a line of a more reddish hue, and slower parts of the line are shown in blue. In this way, the visual remnant carries important information which is not ephemeral like the gesture itself, but which remains visible.

3.5 New Analysis of Sketches

Though there has been much research into recognition of handwriting, recognition of sketches has not been well researched. Most likely this is because it is a difficult problem, and the domains for this type of work may not be well defined. By constraining the domain for this thesis to two areas in which we can demonstrate useful applications for sketching gestures, we demonstrate a system which explores the concept of controlling an object's behavior through time using a very expressive input means.

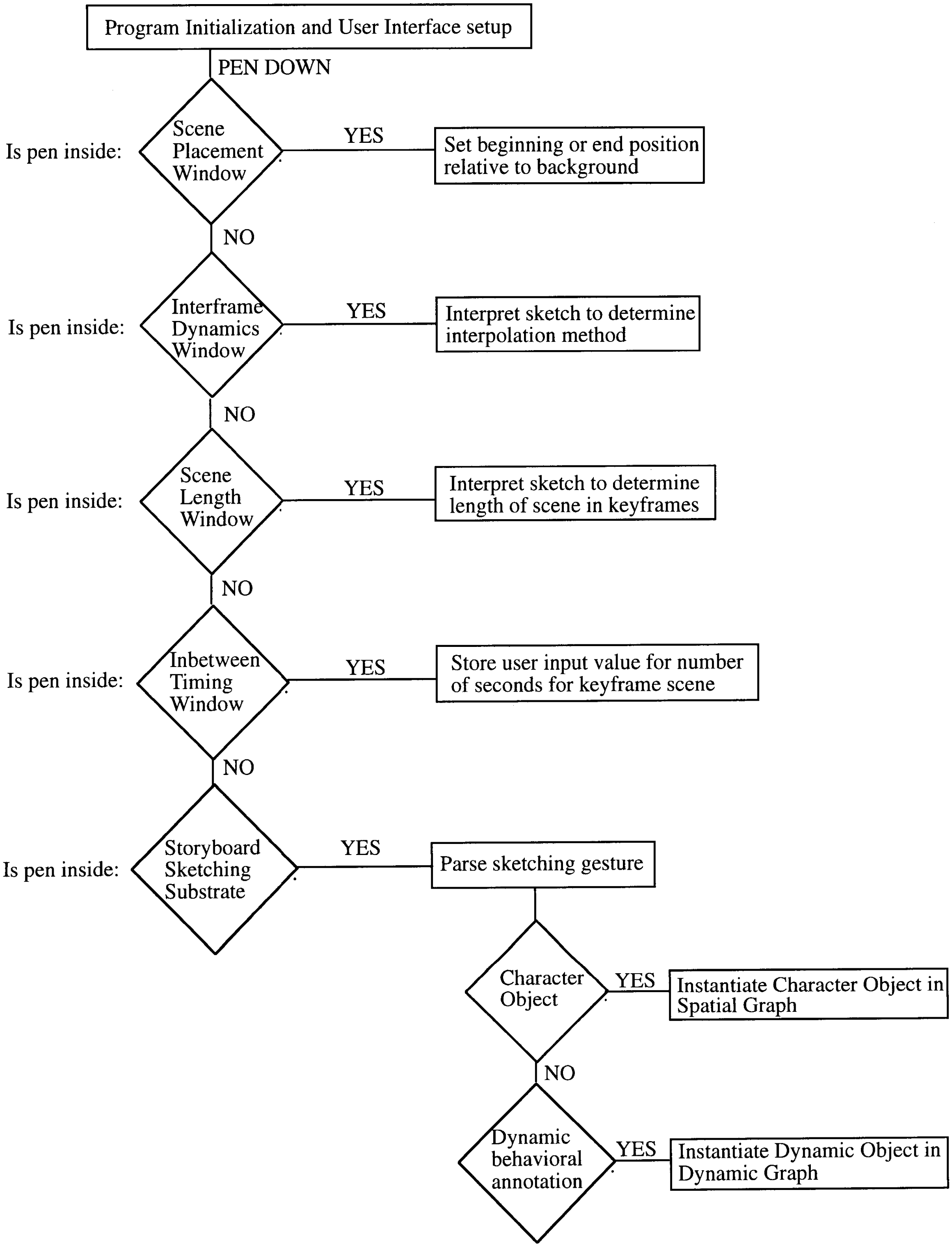

Program Initialization and User Interface setup
PEN DOWN
Is pen inside:
Scene Placement Window
YES
Set beginning or end position relative to background
NO
Is pen inside:
Interframe Dynamics Window
YES
Interpret sketch to determine interpolation method
NO
Is pen inside:
Scene Length Window
YES
Interpret sketch to determine length of scene in keyframes
NO
Is pen inside:
Inbetween Timing Window
YES
Store user input value for number of seconds for keyframe scene
NO
Is pen inside:
Storyboard Sketching Substrate
YES
Parse sketching gesture
Character Object
YES
Instantiate Character Object in Spatial Graph
NO
Dynamic behavioral annotation
YES
Instantiate Dynamic Object in Dynamic Graph

Figure 3.12: Overall control mechanism for the "Intelligent Pencil Test" system.

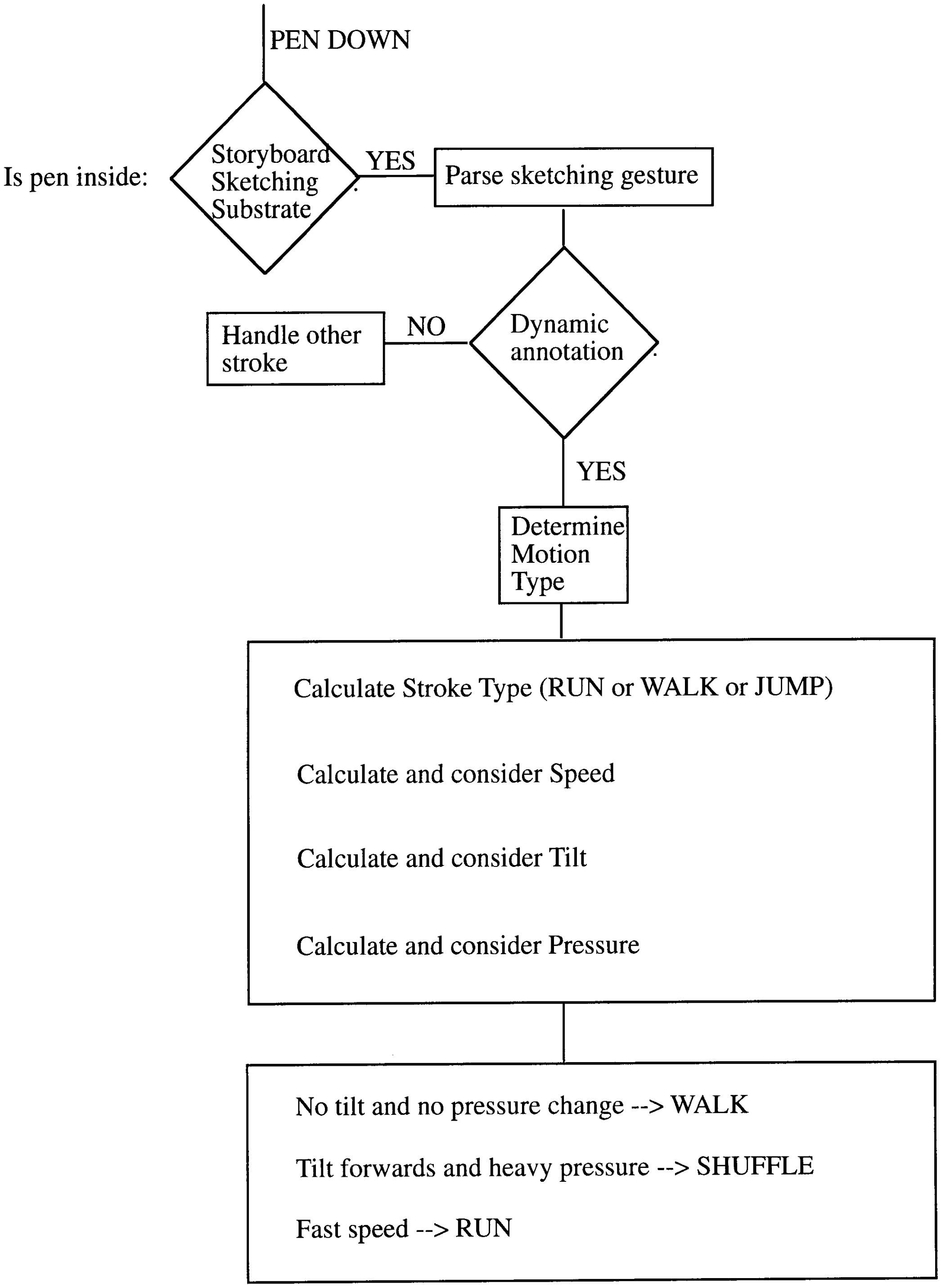

PEN DOWN
Is pen inside:
Storyboard Sketching Substrate
YES
Parse sketching gesture
Handle other stroke
NO
Dynamic annotation
YES
Determine Motion Type
Calculate Stroke Type (RUN or WALK or JUMP)
Calculate and consider Speed
Calculate and consider Tilt
Calculate and consider Pressure
No tilt and no pressure change --> WALK
Tilt forwards and heavy pressure --> SHUFFLE
Fast speed --> RUN

Figure 3.13: Overview of motion interpretation component of the "Intelligent Pencil Test" system. Motion type (run, walk, jump, etc.) is based on the geometry as well as the temporal features of the sketch gesture (arrow) used to indicate motion.

4. System Design and Implementation

The Sketching Substrate system was designed to run on an HP Bobcat series 835 workstation, with a RISC-based processor and two 24-bit frame buffers. A Calcomp DrawingBoard II pressure/tilt-sensitive stylus [54] is used as the input device. The software is written in C under the UNIX operating system, with underlying drawing functions supported by the Starbase Graphics Library and the antialiased and translucency line drawing functions available through Bad Windows, the VLW window system. The interface is built using the Bad Windows libraries which pertain to window creation and manipulation. Additional hardware which was used in the process of creating the demonstrations includes a 24-bit color scanner.

4.1 Sketch Recognition

In this system, the sketch recognition aspect of the interface is aimed at quickly and efficiently interpreting the user's sketched strokes. The Sketching Substrate can be thought of as a surface associated on top of a drawing surface, such as a window on the screen or a virtual document. The most important aspect of the Substrate is that it translates sketches into viable control mechanisms for motion later on. To do this, it relies of a grammar to translate raw sketched data into useful mappings for controlling dynamic behavior of objects such as a two-legged character in an animation.

This Sketching Substrate attempts to classify every mark upon it and aspects of how each mark is made, and considers both the time that the user has the stylus on the table as well as when the user has the stylus in the air above the tablet. This time during which the user holds the stylus above the tablet, without being in contact with its surface but still in sensing range, we refer to as the *stationary dwell time*. We employ a multi-perspective means of analyzing strokes which considers the geometry as well as spatial and temporal relationships between sketched strokes. The sketch recognition is based on a feature extraction approach, and augmented with a temporal constraint to indicate groupings of sketch components. The algorithms for discerning specific stroke types are documented in the appendices in Chapter 8.

Strokes are interpreted either as single strokes or as groups of strokes. A set of strokes which are drawn within a specific amount of time are referred to as a stroke "group". Our sketching grammar uses time as a delimiter to separate chunks of information. The stationary dwell time threshold is considered the time limit for classifying a group as strokes as a real group and not as individual sketched marks. For example, in order to qualify as a rectangle, a groups of four strokes each must be drawn within some amount of time *t* between each other. After this, the user must allow the stylus to dwell in the air not touching the tablet for an amount of time which is equal to or longer than our between-stroke threshold limit.

The design decision to constrain the interpretation of sketch stroke groups by time originates from observations of the way in which people sketch geometric figures such as boxes and triangles. There are other means for delimiting chunks of sketched information which could have been employed, such as the user of speech (e.g. - having the user say "Done" after completing a sketch group) or by using spatial relationships. Instead, the designer of this system chose to use time as a delimiting factor because it allows the user to sketch at a speed which he controls, without having to stop and change input modes to indicate that he is done. Additionally, it forces the user to complete an entire visual idea before proceeding to the next.

Our grammar definition is one which can be easily changed to provide for additions to the language. At the simple stroke level, current elements of the sketch language include lines, arrows, curves. From these simple shapes are built more complex shapes such as rectangles. Though this thesis will not focus on allowing the user to add to the grammar, the section of extensibility includes discussion on provisions for allowing for the addition or editing of gestures.

4.2 Recognition of Single Strokes

Our knowledge representation scheme is one of procedural knowledge used in conjunction with rule-based knowledge. When the user draws a stroke, it is filtered for any outstanding physical qualities such as shape and orientation. If the system discerns that the stroke is a line, it is classified as either a curve or straight line, and aspects such as direction are considered. Once the general form for the mark is realized, we determine whether the mark falls outside of an acceptable range from a canonical form for the type of line. For example, a line which is initially tagged as "diagonal" is compared against the canonical diagonal line of the same size. For each stroke, we associate with it the points which constitute it, and the articulation qualities which are expressed at the time the stroke is created, such as pressure and tilt in X and Y at every time *t*. As strokes are added to the

Substrate, they are entered (sorted in Y major) into a Spatial Graph S which contains knowledge about spatial aspects of the stroke as well as relationships between the stroke and any of its neighboring strokes. Also associated with each spatial element Sn is an element of a Qualitative graph Qn, in which information about the probable canonical form of the sketched object is interpreted according to a set of rules and inference methods. A connecting link between the spatial object Sn and its qualitative object description node Qn is set.

4.3 Recognition of Stroke Groups

Stroke groups are recognized according to a temporal delimiting scheme. Figure 4.1 depicts the recognition of multiple-stroke groups. A typical stroke group might be an arrow, in which the stem determines both spatial information and behavioral visual information. For example, an arrow drawn emanating from an object such as a 2D figure might indicate how this figure should move spatially and how his motion should change over time. At a higher level, the Spatial Graph keeps track of objects to which temporal marks (such as arrows) have been applied.

4.4 Interpretation of Stroke Qualities

The Calcomp Drawingboard II tablet and stylus are used for this demonstration, both for the high data rate (125 points per second) and for the tablet's ability to sense the tilt in the X and Y direction of the stylus. The data rate of 125 points per second is adequate for giving very realistic visual feedback to the user, though the graphics and floating point math required by the Substrate layer and the successive graphics display routines slow the system, resulting occasionally in jerky or aliased strokes on the screen.

In order to use this expressive information which reflects the articulations of the sketched input, we have designed a grammar which maps physical articulation qualities to visual and behavioral qualities. At the lowest level, the Sketching Substrate considers the raw input signal from the stylus, does the low-level gesture recognition, and parses these gestures according to the current contextual grammar. These contextual grammars can be either static or dynamic. The static grammar represents a snapshot of the stylus position at any single point *t* in time. The dynamic grammar considers the continuum of time during the creation of the stroke, and the samples of the stylus over time represent the static "views" of this dynamic profile over time.

This grammar can be extended to encompass more symbols which have contextual meaning. For example, in our current grammar, tilt and pressure control the qualitative graph Q which maps these qualities to translucency and line weight. We could instead switch these grammar mappings so that tilt controls line weight and pressure control translucency. In this way, pressing harder on the stylus would cause an object to fade or become more opaque (depending on the polarity of the mapping). Tilting the pen from one side to the other would cause the visual weight of the object to change. Additionally, the simplicity of the gestural syntax directly affects how "sloppy" the user can be in gesturing to the system. In our graphics design example our syntax includes only sketches to mean text and images, both of which are extremely simple but are different types of sketches and whose articulation can carry much visual meaning. A heavily drawn squiggle will result in a heavier mark on the page, which translates into more visual weight in terms of the layout. Also, the simplicity of the sketching syntax allows the user to sketch quickly, without having to remember many complicated symbols or gestures to remember..

4.5 Translating Sketched Strokes into Motion

Hand-drawn motion is transferred into motion control via the Sketching Substrate and the contextual gesture grammar, which provides a mapping between the input characteristics and the final resultant visual effect of the sketched input. The Sketching Substrate system "parses" a sketch made by the user, first by syntactically analyzing the sketch in terms of it contextual sketching grammar. In the context of the Dynamic Page demonstration, an example exercise might be to sketch the design for a page layout of an interactive magazine [47, 48] by using notations for images and text, and to annotate these components with indications of how they might behave temporally or dynamically. An example of this might be to indicate that all images should fade and that all text becomes more transparent, as the page is actively being read. These temporal annotations are then interpreted over time by projecting the components of Temporal Graph onto the Spatial Graph, using the current contextual mapping (as stored in the Qualitative Graph) to control the visual appearance of objects over time.

Temporal annotation is invoked when the user opens a virtual transparent substrate over the page through a menu pick. All of the annotations made in this mode are thus "captured" by the substrate and are considered to be temporal in nature and interpreted as such. They are instantiated as elements of the Temporal graph with associated links to the related Spatial group. We can choose to view the results of these temporal annotations by using different criteria. We can have the page play out based on visual weight as a criterion, or by using the sketching speed as a criteria for selecting which temporal annotations to apply first.

The sketch marks left on the virtual page are 2D visual remnants of the sketching gestures used to make these marks. In the substrate, we capture the information about the way in which the marks were created for later use by the system in controlling the visual appearance and temporal behaviors of objects. This information is stored by the system and associated with each notation through the Spatial and Temporal, and Qualitative Graphs. The Qualitative Graph reflects the physical qualities of the objects. These include color, size, shape, type, etc. Our Spatial graph reflects the spatial orientation of the page, and knows about spatial relationships between objects.

5. Results

This chapter contains information about the overall results in the design, implementation, and use and experimentation of the prototype systems. We also discuss what was learned about the design of such a system.

5.1 The Dynamic Page Demonstration

This demonstration allows for rapid sketched input of layouts (Figure 5.1), where sketching gestures indicate such qualities as visual weight, translucency, and type style (Figure 5.2). After replacing the sketched strokes with canonical type forms, the user can change visual qualities of these objects by using gestures. For example, pointing to a piece of type and applying more force to the stylus on the tablet changes the boldness, or "weight" of the type. Pointing to a piece of type and changing the tilt of the stylus results in a change in translucency. Hence, we provide the user with a new "surface" through which he can control visual information on the screen. Since type qualities are usually explicitly controlled through menu choices, this gestural interaction is probably not the best way to do fine-granularity control of objects on a page. Additionally, users asked to see real-time feedback of quantifiable information, such as the current font weight or point side based on the current sketching motions. To solve this problem, we incorporated a visual feedback window into the user interface to provide the user with information about how the system has interpreted the last object recognized.

We can use this prototype to create layouts whose text and image objects change their spatial and visual qualities over time. This would be useful in sketching out a graphical user interface in which objects move (such as windows) as a result of user interaction. In our demonstration, the user chooses "Dynamic Annotation" from the menu to enter an annotation mode in which all marks are interpreted as temporal in nature. These marks can be thought of as a "temporal markup" language. These marks result in the use of another inking color, the translucency of which is controlled by tilt in the X direction, and the line weight is controlled by pressure. The visual aspects of the objects to which these temporal annotations apply will change over time as a result of changes in the articulation of the temporal sketch. This aspect of the design proved a challenging one to represent in the markup language, since we are not accustomed to viewing static

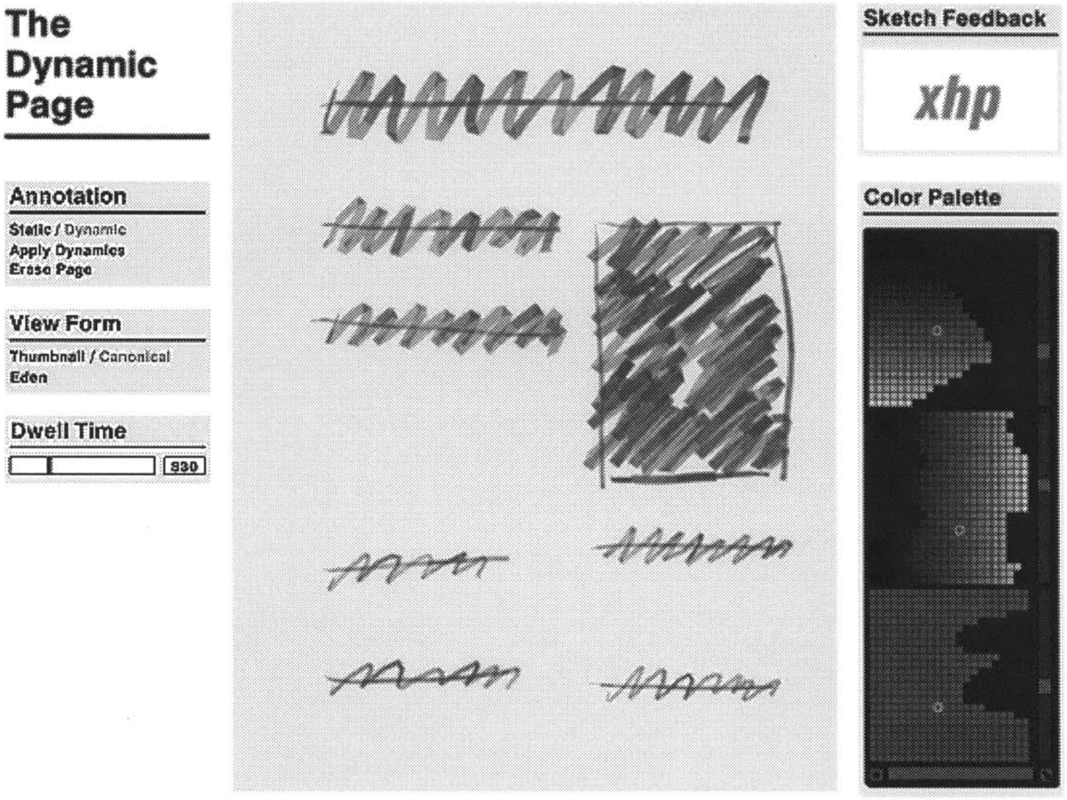


Figure 5.1: The Dynamic Page prototype user interface showing sketching area and visual feedback window in upper right corner.

information in terms of its temporal or dynamic qualities. In trying to show how the articulation of the sketched marks change over time, we employed the use of a colored "visual remnant" whose color reflected changes in speed and acceleration.

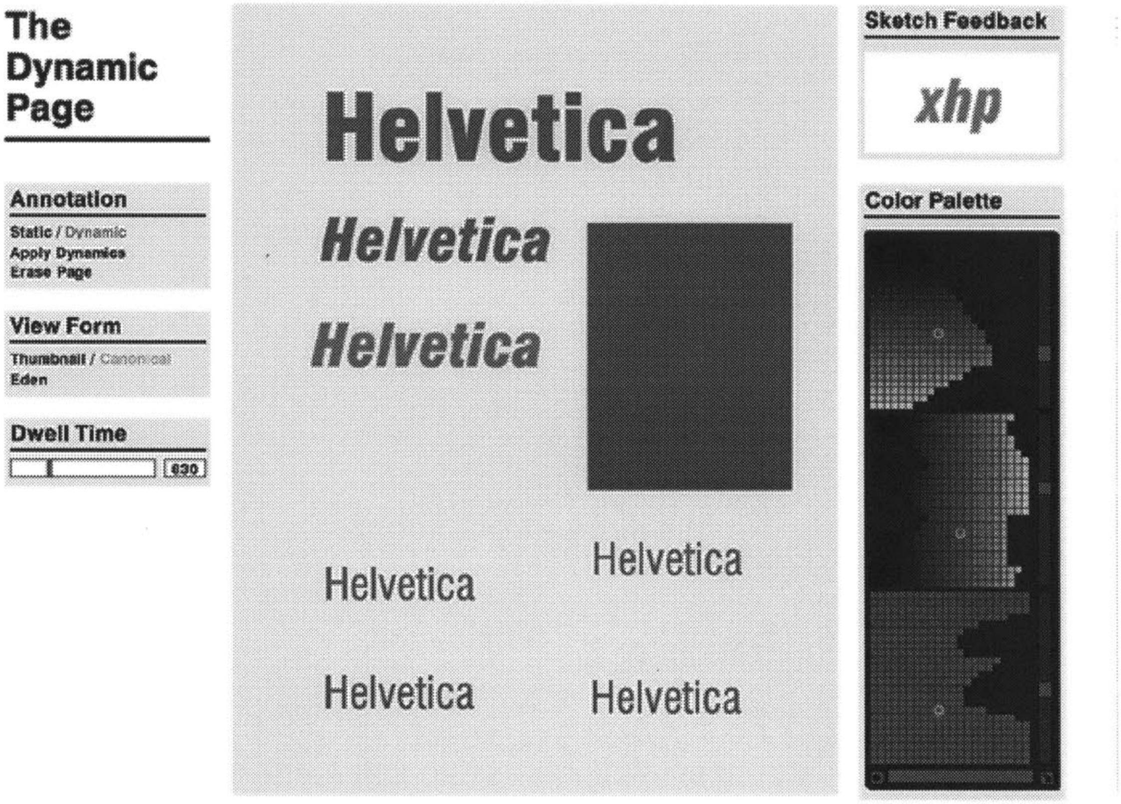


Figure 5.2: The Dynamic Page prototype user interface showing sketched marks replaced by canonical type and image forms, based on the pressure, tilt, and color of the sketches.

5.2 The Sketch-Based Animation Demonstration

Animation is a temporal event in which an object behaves according to a physical or other law. It is represented by the renderer whether human or machine. Viewing frames or "small multiples" of the object over time allows us to see static snapshots of this behavior. This prototype system allows an animator to very quickly rough out sketches which determine the motions and behaviors of a character over time (Figure 5.3). By interpreting these sketches according to the current grammar which controls the Sketching Substrate, the system recognizes the rough sketches as denoting either a canonical character or a motion for the character to perform. By sketching out motions for each character in each frame, the system can interpreted these marks and cause the canonical forms to move accordingly. In this way, the animator can very quickly sketch out a scene, and get a feel for the type of motions employed in each set of key frames.

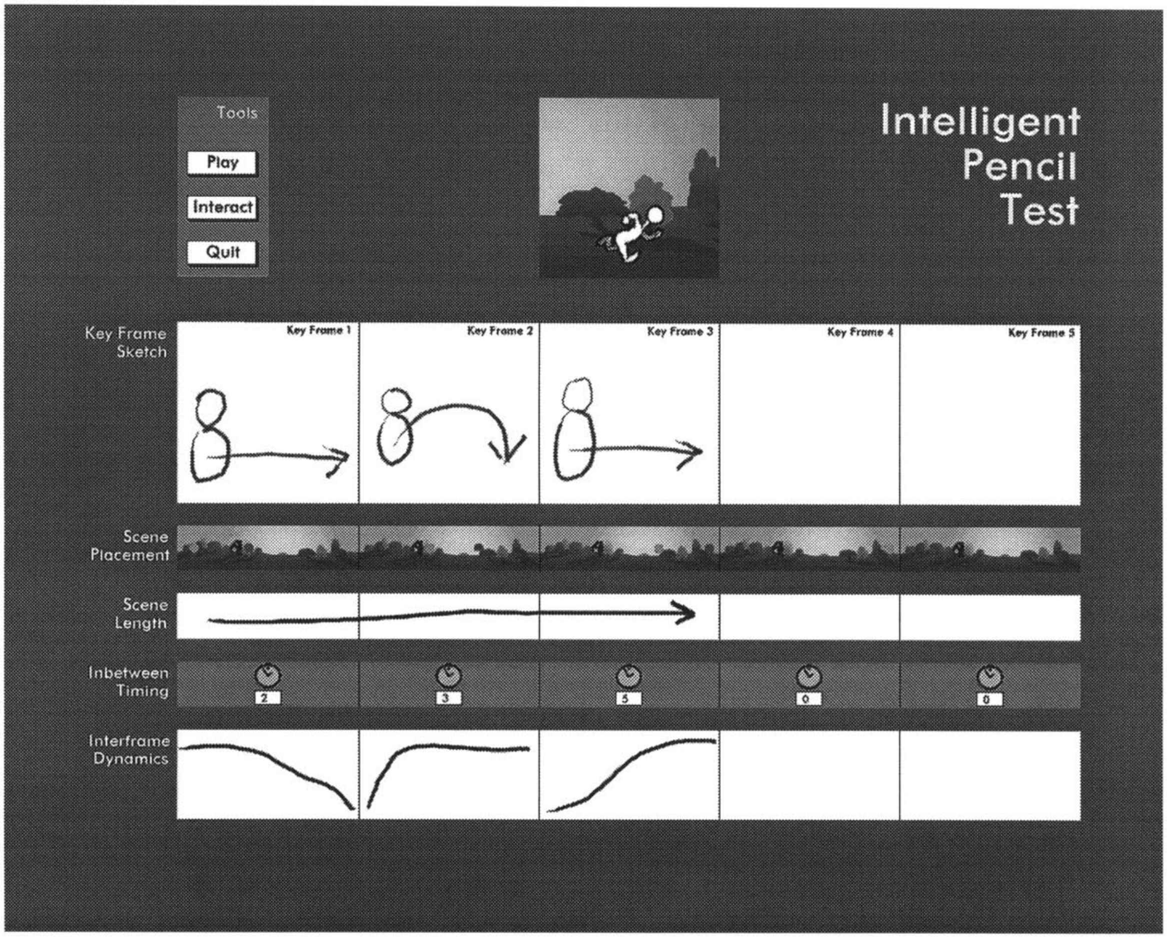


Figure 5.3: The Intelligent Pencil Test prototype user interface. Shown here are three sketches which indicate motion of a 2D character over three keyframes. The three motions are run, jump, and walk. The motion types are based on the articulations of the sketch marks. The user has indicated on the Scene Placement window where the character should appear relative to the background scene for each keyframe, and has indicated through sketches in the Interframe Dynamics window how the system should interpolate between key frames.

Because of the HP Bobcat platform, we encountered a problem while building this system. The problem arises out of storage requirements for the 2D motion cycles, since the motion cycles need to be stored in memory for rapid block-moving onto the screen. Within a virtual memory system, this would have not been a problem. However, the HP Bobcat workstation provides only a 768 by 1024 (pixel) offscreen double-buffering area, and for our purposes we needed to store two-plane motion cycles (image and mask planes), each of which was at least 640 by 80 pixels, so that the 2D character would appear as a cut-out figure against the background.

The original goal of the Intelligent Pencil Test system was to enable the creation and control of an animation of a 2D figure strictly by using sketching. This stemmed from the author's dislike of drawing inbetween frames to make even simple animations, and desire for a writing surface which could more intelligently interpret sketches as control mechanisms for motion. The original design of the system assumed that through control of the stylus, the user would be able to control the type of mark, the speed, and specify the duration of the motion. This proved to be too difficult to keep track of while actually using the stylus, since too many controls through one input device tends to become confusing for the user. Also, people are not used to having the duration of their sketch articulation actually impart cntrol information, so novice users had difficulty in controlling the length of their sketch marks relative to the type and duration of the resultant motion. To solve this, we used explicit controls of time between key frames. The user now specifies this with an integer value.

5.3 User-definable Parameters

It became apparent during the design of this system that the interface needed to be adaptable to accommodate sketching speed and sketching habits. To this end, we designed a slider control of the time threshold for the stationary dwell time. (The stationary dwell time threshold is the number of milliseconds the user's stylus is in the air before the system tries to interpret the last group of sketched stroke created.) Most users preferred this threshold to be in the range of 800-1100 milliseconds. Because of the intermittent lag from the Calcomp stylus, this parameter never went below 600 milliseconds.

5.4 Experimentation with Sketching Grammars

This demonstration provides an interface, or surface for interactively dealing with the layout of objects such as type and images on a page. "Pushing" is a good mapping for size. Tilt is a good mapping for translucency, since we typically tilt our writing implement in order to lighten up the force perpendicular to the writing surface. However, tilt is a problem when the user wishes to maintain steady pressure since changes in tilt in the positive X direction (to the right) usually result in decreases in pressure. By changing the grammar to map sketch articulation qualities into dynamic behaviors, we experimented with different mappings. These mappings included: the control of translucency by pressure instead of tilt, and the control of point size (of type) by pressure.

In order to minimize the learning curve for our system, we have chosen notational conventions which, when available, are the same as the visual language of the domains which we have explored. We employ the same notational conventions that graphic designers use in designing thumbnail sketches. Through just a few simple marks, we see through our results that it is possible to impart useful information for designing a page. For animation, the problem is more difficult, since animators do not have a conventional notation language which they employ to communicate such dynamic behaviors as motion. Instead, they create a series of sketches which, when viewed in time, give the viewer the experience of motion. In our system, we have provided the animator with a sketchpad which enables a shorthand means for drawing motion cycles. This shorthand notation is based on a simple grammar and mechanisms to manipulate the temporal aspects of the display of motion cycles.

5.5 User Testing

We tested several users, of varying skill levels. Novel implementation issues included the use of tilt as a means for controlling translucency, which was quickly learned. For the Dynamic Page demonstration, users asked for more tools which would give them more quantifiable visual cues as to what was being interpreted by the system. One of these tools included a means for moving back and forth between sketched marks and canonical forms.

The difficulty in trying to realize the original idea of this thesis of controlling an entire animation using only a sketch. By constraining the system so that only speed and motion type are reflected in the sketch, it proved to be a much better design decision. The most difficult aspect of this thesis was in deriving a quantifiable model of different types of sketching based on empirical data, and incorporating this information into a set of software routines which could classify the marks (circle, line, etc) and map their articulations into useful control mechanisms for animation.

6. Future Work and Applications

6.1 Future Work

This thesis explores the use of expressive, gestural-based sketching as a means of communicating information to a computer. The computer system uses this information inherent in the gestures to capture and interpret this information in ways which are meaningful and allow for the control of processes which previously were realized on paper. Hence we provide a novel "surface" through which we translate sketching articulations into useful control mechanisms. We can control the visual qualities of information on a page: this surface, for example, allows us to simply "push" on a piece of type to increase its boldness, pointsize, or any other visual quality for which we define a mapping between input signal and final visual form. In addition, we capture the user's interaction with this surface during the design process, so that we have an accurate record of this process over time. This could provide new means for capturing stylistic solutions to visual design problems, and for capture of the design process.

Examples of the benefits of such as system can be found in the current research on understanding layouts and encoding visual design knowledge. The system described in this thesis would provide an intuitive means for inputting layout designs for use by a unification-based grammar system such as that being explored by Weitzman [27]. Colby's LIGA system [42], which maintains stylistic and legibility requirements for layouts in dynamic environments, could also use a sketch-based interface to quickly impart information about the design of a layout, and how its constituent object would appear visually. For example, a rule bar drawn with the sketch system could automatically produce a canonical rule bar of the correct weight, based on relationships to other pieces of type in the layout. Current layout encoding research typically uses Lisp-like rules or frame knowledge which requires the user to encode the design knowledge as a series of procedures in Lisp or CLOS. The Sketching Substrate system offers the non-programmer the chance to encode design knowledge rapidly, intuitively, and perhaps more efficiently.

Additionally, this system could provide means for demonstrating exemplar types of design solutions being investigated by researchers in the field of Learning by Example system. Turranksy's [43] system for capturing design solutions by demonstrating specific examples to the computer could be extended to use a gestural interface as a means of imparting design knowledge to the system.

The Sketching Substrate sketching gesture interface provides a good means of communicating temporal information about the placement and motion of a 2D character and could be of use as an interface to the XSpace system [30] demonstrated by Librande. The XSpace system would make provisions to generate the interpolated frames in real-time, from a vector-based database of the frame descriptions. Unlike the Intelligent Pencil Test demonstrated in this thesis, the motion cycles would not need to be stored offscreen, using valuable memory and requiring complicated multi-layered drawing routines to animate the character on the screen. The frames could be generated on-the-fly and could reflect even more of the nuances of the sketched input since the frames which constitute the motion cycles would not be pre-determined, and could in fact be affected in real-time.

6.2 Extensibility

There are many aspects of this system which could be extended, both to afford more powerful control of dynamic information and in the interpretation of gestural input. The obvious first modification would be to use a virtual system which is not constrained by limited video memory and whose real-time bitblitting capabilities would allow for real-time multi-layer block moves of information. This would do away with the use of a backing store, which hindered progress in the development of the Sketching Substrate system. A parallel processor would enable the separation of tasks: one processor could handle display and others could handle interpretation of input and control of the motion display. Another set of processors could control the motion of several characters. A higher data rate of the stylus, in the vicinity of 200 or more points per second would alleviate the antialiasing artifacts occasionally present on the screen, and would perhaps allow for a better means of capturing subtle changes in sketching articulation. Better tilt information would also benefit this process, and the ability of the pen to sense rotation would also be beneficial in the control of objects whose orientation would need to change.

Coupled with a multimodal means of input such as voice and force-feedback, this system could become a sophisticated animation tool. Augmented with speech input, the system would allow the animator to tell the system how the character should move and perhaps with the integration of a 3D database of scene objects, this would enable complex motion scenes. 3D knowledge of background content would also enable the background to play a larger role in the animation, and allow the character to avoid other characters or objects in the background.

The integration of an interface agent would also be an area for further research. A learning component in the Sketching Substrate interface would allow the system to build up an extensive model over time of the user's sketching behavior, especially with regards to dynamics. This would facilitate a "designer's assistant" which could use multiple agents to consider different aspects of the user's sketching activities in different contexts, and store and reuse this information

6.3 Applications

The system described here could benefit several application areas, including animation, graphic design and the rapid prototyping of user interfaces. The Sketching Substrate model provides a means of rapidly inputting sketched layouts into a system such as Colby's LIGA [42] prototype. Beyond communicating static stylistic information to an intelligent layout system, the system described in this thesis forms a basis from which to build rapid prototyping tools which enable designers to quickly communicate dynamic design knowledge to a computer system. User interfaces designers could use aspects of the Dynamic Page demonstration to do rapid prototyping of interactive graphical user interfaces. Using the Dynamic Page demonstration as a visualization tool, for example, a designer could rapidly denote how a window should behave visually in response to user input. As generations of input styli become more responsive to reflect the user's gestures, one could apply the gestural recognition model in studying motor coordination and the kinetics of hand motion: for example, as a tool to measure and detect hand-tremor in heart attack patients [49]. The Sketching Substrate could also form the basis for an interactive teaching tool for learning layout design, in which the user's input sketches are analyzed and compared to canonical layouts. The system could provide an interactive tutorial experience based on its knowledge of expert layouts and the sketches used to created those layouts.

7. Conclusion

A prototype research computer system has been designed and implemented to interpret sketching gestures made with an expressive stylus. The sketching gestures are interpreted in terms of the information implicit in the articulations of the gestures and their use in controlling dynamic behaviors of objects in an electronic environment.

This methodology, referred to as the Intelligent Sketching Substrate, has been used as a prototype tool for visualizing dynamic behaviors of 2D objects over time, as demonstrated for the two domains of graphic design and animation. With this system, a designer can quickly sketch a layout, and have the system interpret information such as the type of objects (text or image), visual weight, size, translucency and style expressed at the time the sketches were made. Using more sketching gestures, the designer can impart information about how these objects should change their appearance over time. As an animation tool, this system provides the animator with a means for rapidly visualizing motion cycles relative to a background scene. Through sketching gestures, the animator can quickly change temporal aspects of the animated sequence, such as the timing of the interpolation between key frames.

Useful measurable and derivable information is implicit in gesture, and we have shown in this thesis that it is possible to impart instruction through sketching gestures about visual form and control of objects over time in a computing environment. Current stylus technology accurately reflects the gestures of the user, giving us an accurate record of the sketching process. In this thesis, we have made use of several aspects of this rich input signal. For example, tilt is used to control translucency, and the speed with which an arrow is drawn determines whether its meaning is *walk* or *run*. Difficult questions still remain: is there a minimum acceptable resolution needed to "capture" the creative process behind the sketching act? Once captured, can we construct models which enable us to use this information in useful ways, perhaps to "teach" a computer system how to mimic our drawing style? This type of interface is not realizable today but will be in the near future, as computer-based models of learning develop, along with computer speed and storage, and stylus technology.

Stylus-based computer interfaces continue to evolve in tandem with the rapid growth of personalized portable computing. In these future interfaces, sketched input will carry more meaning. Sketched marks will not be mere inked annotations, but instead will be marks that "have a sense of themselves", and whose articulation can carry meaning. One could imagine annotating an architectural blueprint with a stylus whose "ink" knows when it is being drawn over plans for a house: for example, a sketching gesture across a wall might indicate the need for a doorway. This is but one example in which the marks themselves can infer in what context they are being made, and can adapt themselves accordingly.

What we have been able to show in this prototype system is that current stylus hardware and software can capture enough of the subtleties of the sketching act to distinguish between two marks which visually appear to be the same but whose articulations differ. Humans can tell a slight curve from a straight line, but it is more difficult to enable a computer to do it. The most difficult aspect of developing this prototype system was the creation of a quantifiable model understandable by the computer which allowed a raw signal from the stylus to be used as a useful control mechanism for controlling visual qualities of objects on the screen. Another problem arose from scaling: there is no direct correlation between the tactile experience of the hand and what the user sees on the screen. This may be alleviated if future stylus hardware evolves to include a tactile feedback mechanism. In order to provide a useful computer interface, a responsive surface which captures and interprets sketched gestures into useful control mechanisms, the system must calibrate what the hand does to what the eye sees as a result.

8. Appendices

This appendix contains the algorithms used to discern strokes and stroke groups. The following algorithms are written in pidgin C, and use the STROKE and GROUP definitions shown below. The values used as thresholds in the algorithms given below are based on a virtual coordinate system which is 100 dots per inch. The tablet sampling resolution is 1000 dots per inch.

8.1 Structure Definitions

POINT
x position on screen
y position on screen
pressure (0.0 - 1.0)
x tilt (-64 to +64)
y tilt (-64 to +64)
pointer to next POINT in list
pointer to previous POINT in list

RECTANGLE
upper left hand corner (x, y)
lower right hand corner (x, y)

STROKE
type of STROKE (HORIZONTAL, VERTICAL, etc)
number of POINTS in STROKE
color of STROKE (red, green. blue)
RECTANGLE boundary of STROKE
pointer to list of associated POINTs
center POINT of STROKE
length of STROKE in pixels
timestamp for STROKE (time at beginning and end)
pointer to next STROKE in list
pointer to previous STROKE in list

GROUP
number of STROKEs in GROUP
type of GROUP (MOTION ARROW, etc)
color of GROUP (red, green, blue)
RECTANGLE boundary of GROUP
pointer to first and last STROKE in GROUP
pointer to next GROUP in list
pointer to previous GROUP in list

8.2 Stroke Recognition Routines

int is_dot(s)

struct STROKE *s;

Returns TRUE if all the points of the input stroke are within a radius of 10.0 from the center of the stroke, returns FALSE otherwise.

int is_line(s)

struct STROKE *s;

Determines the slope and an offset (m and b) from the first and last points of the stroke, predicts a y value for each point using the x value of the point and m and b, returns -1 if the y value of any point is greater than the length of the stroke/5 from its predicted value, otherwise returns HORIZONTAL if the absolute value of the stroke < .5, VERTICAL if the absolute value of the slope is > 5, SLASH if the value of the slope is between .5 and 5, or BACKSLASH if the value of the slope is between -.5 and 5. Calls is_vert() if the slope of the stroke is > 5 and the stroke does not satisfy the above criterion, because for large m, y = mx + b can vary widely for small variations in x, making it difficult to recognize a vertical line.

int is_vert(s)

struct STROKE *s;

Returns TRUE if all x values of a stroke are within the length of the stroke/5 from the average x. Returns FALSE otherwise.

int is_focus(s)

struct STROKE *s;

Returns TRUE if the stroke is closed (the first point is closer than 10.0 from the last point) and crosses its average radius more than twice. Returns FALSE otherwise.

int is_ellipse(s)

struct STROKE *s;

Finds the two longest and two shortest radii of the stroke, checks to see that the two long radii have approximately the same length and slope, that the two short radii have approximately the same length, that the long radii are sufficiently longer than the short radii, and that the stroke is closed (the first point is closer than 10.0 from the last point). Returns TRUE if the above criterion are satisfied, Returns FALSE otherwise.

int is_circle(s)

struct STROKE *s;

Returns TRUE if each point is within an acceptable distance from the average radius and the stroke is closed (the first point is closer than 10.0 from the last point).

int is_curve(s)

struct STROKE *s;

Returns TRUE if all the points in the stroke do not cross the line connecting the first and last points of the stroke. Returns FALSE otherwise.

8.3 Group Identifiers

int is_arrow(g)

struct GROUP *g;

Checks to see that the group consists of 3 strokes, two of which are lines, and the third of which is either a line, wave, circle, or spiral. Check that the two shortest strokes are lines, that the long stroke is sufficiently longer than the short strokes, and that all three strokes meet at a point.

int is_scale(g)

struct GROUP *g;

Checks to see that the group consists of 5 strokes whose types are lines, finds the longest line, checks to see that the longest stroke is connected in two points with the other 4 lines. Returns YSCALE if the longest stroke is a vertical line, XSCALE if the longest stroke is a horizontal line, or XYSCALE if the longest stroke is a slash or a backslash. Returns FALSE if the above criterion are not satisfied.

8.4 Intelligent Pencil Test Demonstration Algorithms

This is the main motion-control algorithm which maps sketched marks into motion cycles of a 2D two-legged character. The motion cycles are stored offscreen in a backing store as separate image and mask planes, and bitblitted onto the screen in the correct location relative to the background.

```
Play(motion_type, current_cycle_element)
        int motion_type, current_cycle_element;
{
        /* the motion cycles are stored offscreen in 8-frames per cycle format */
        Calculate correct speed over time based on background scene placement and Inbetween
                Timing information. Incorporate the sketched function derived from the
                Interframe Dynamics window to control the interpolation, if applicable.

        Display the motion cycle which represents motion_type, beginning at the
                current_cycle_element, and continuing through the entire motion cycle.
}
```

```
PlayMotion(motion_type, current_cycle_element)
      int motion_type, current_cycle_element;
{
        /* assume that the motion type has been previoulsy tagged according to the routines
listed above  */
        if(motion_type == JUMP) Play(JUMP, current_cycle_element);
        else
        {
                Play(motion_type{WALK | RUN | SHUFFLE}, current_cycle_element);
        }
}
```

9. Bibliography and References

9.1 Bibliography

Expressive Input

[1] Sutherland, I. SKETCHPAD: A Man-Machine Graphical Communication System, PhD Thesis, MIT Department of Electrical Engineering, 1963

[2] Bleser, T. W., Sibert, J. L., and McGee, J.P., Charcoal Sketching: Returning Control to the Artist, ACM Transactions on Graphics, 7, 1, 76-81

[3] Donoghue, K., Knowlton, K. Methods of Simulation of Calligraphic Pens and Brushes, Proceedings of Second International Conference on Electronic Art, Groningen, The Netherlands, November 1990

[4] Ware, C., Baxter, C. Bat Brushes: On the Uses of Six Position and Orientation Parameters in a Paint Program, CHI '89 Proceedings, ACM Conference on Human Factors in Software, 155-160

[5] Fractal Design Painter: pressure sensitive paint program. Fractal Design Corporation, 1992

[6] Studio/32: pressure sensitive paint program, Electronic Arts, 1992

Gesture Recognition

[7] Buxton, W., Fiume, E., Hill, R., Lee, A., and Woo, C. Continuous Hand-Gesture Driven Input. Proceedings of Graphics Interface '83, Edmonton, May 1983, 191-195.

[8] Coleman, M.L. Text editing on a graphic display device using hand-drawn proofreader's symbols. *Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinois Conference and Computer Graphics*, M. Faiman and J. Nievergelt. University of Illinois Press, Urbana, Chicago, London, 1969, pp. 283-290.

[9] Zimmerman, Lanier et al. A Hand Gesture Interface Design. CHI+GI 1987, 189-192.

[10] Buxton, W., Sniderman, R., Reeves, W., Patel, S., and Baecker, R. The evolution of the SSSP score-editing tools. *Foundations of Computer Music*, C. Roads and J. Strawn, Eds. MIT Press, Cambridge, Mass., 1985, Chapter 22, pp. 387-392.

[11] Ward, J.R., Blesser, B. Interactive Recognition of Handprinted Characters for Computer Input, IEEE Computer Graphics and Applications, 5(9), 24-37

[12] Pencept "Penpad"(TM), 320 Technical Data, Pencept, Inc, 39 Green St, Waltham, MA 02154 (1984)

[13] Rubine, D. Specifying Gestures by Example, SIGGRAPH Proceedings, Computer Graphics, Volume 25, Number 4, (July 1991), 329-337

[14] Hardock, G. Design Issues for Line-Driven Text Editing/Annotation Systems. Graphics Interface '91

[15] Buxton, W. The Pragmatics of Haptic Input, Tutorial Notes, CHI 1990, Seattle, Washington.

16] Herot, C. Graphical Input Through Machine Recognition of Sketches, *Computer Graphics*, 10(2), 97-102

[17] Kurtenbach, and Buxton, W. GEDIT System

[18] Fels, S. and Hinton, G. Building Adaptive Interfaces with Neural Networks: The Glove-Talk Pilot Study. Dept. of Computer Science, University of Toronto, 1990. Submitted for publication.

[19] Segen, J. GEST: A Learning Computer Vision System that Recognizes Gestures, in *Machine Learning IV*, Michaelski et al eds., Morgan Kauffman, 1992.

[20] Krueger, M., Gionfriddo, T., and Hinrichsen, K. VIDEOPLACE: An Artificial Reality, CHI 1985 Proceedings, 35-40.

[21] Pausch, R., Vogtle, L., Conway, M. One Dimensional Motion Tailoring for the Disabled: A User Study, CHI 1992, 405-411.

Gestural Communication

[22] Poizer, H., Klima, E., and Bellugi, U. *What the Hands Reveal About the Brain*, Bradford Books, MIT Press, Cambridge, MA, 1987.

Diagram and Layout Understanding

[23] Montalvo, F. Diagram Understanding: the Symbolic Descriptions Behind the Scenes. I n *Visual Languages*. Eds. Chang, S, Ichikawa, T., Ligomenides, P. New York: Plenum Press, 1986

[24] Greenlee, R. From Sketch to Layout: Using Abstract Properties to Generate Page Layouts, Master's Thesis, MIT Media Laboratory, 1988

[25] Funt, B. Problem-Solving with Diagrammatic Representations, *Artificial Intelligence* 13(3), 1980, 201-230.

[26] Helm, R., Marriot, K., Odersky, M. Building Visual Language Parsers, CHI 1991 Proceedings, ACM Conference on Human Factors in Software, 105-112

[27] Wittenburg, K., Weitzman, L., and Talley, J. Unification-based Grammars and Tabular Parsing for Graphical Languages, Journal of Visual Languages and Computing (1991) Volume 2, 347-370.

Animation

[28] Thomas, Frank and Johnson, Ollie. *Disney Animation - The Illusion of Life*. New York: Abbeville Press, 1984.

[29] Sabiston, W. R. Extracting 3D Motion from Hand-Drawn Animated Figures, Master's Thesis, MIT Media Laboratory, 1991

[30] Librande, S. Example-Based Character Drawing, Master's Thesis, MIT Media Laboratory, 1992

[31] Donoghue, K. Extraction of Expressive Emotions in a Facial Sketch, MIT Media Lab Internal Paper, May 1992

[32] White, T. The Animator's Workbook. New York, N.Y: Watson-Guptill Publications, 1988.

[33] Baeker, R. (1969) Picture-driven animation. Proceedings of the Spring Joint Computer Conference, 273-288.

[34] Macromind Director 3.0, ©1991 Macromind.

[35] Adobe Premiere, ©1991 Adobe Systems, Inc.

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

Graphic Design

[37] Gatta, K., Lange, G., Lyons, M. *Foundations of Graphic Design*, Davis Publications, Worcester, MA, 1991

Dance Notation

[38] Guest, A.H *Dance Notation: The Process of Recording Movement on Paper* (London, 1984)

Information Design

[39] Tufte, Edward R. *Envisioning Information.* Cheshire, CT: Graphics Press, 1990.

[40] Tufte, Edward R. Visual Explanations. Lecture at MIT Media Laboratory, September 23, 1992.

User Interface Design

[41] Won, Y.Y. Rough and Ready Prototypes: Lessons from Graphic Design, CHI '92, Short Talk Session, (April 1992), 83-84

Layout Design Encoding

[42] Colby, G. Intelligent Layout for Information Display: An Approach Using Constraints and Case-based Reasoning, Master's Thesis, MIT Media Lab, 1992.

Learning by Demonstration

[43] Turransky, A. Capturing Graphic Design Knowledge from Interactive User Demonstrations, Master's Thesis, MIT Media Lab, 1993.

Radial Basis Function

[44] Poggio, T. and Brunelli, R. A Novel Approach to Graphics. AI Memo No. 1354. C.B.I.P Paper No. 71. Massachusetts Institute of Technology. February 1992.

Hyperinstruments

[45] Machover, T. Hyperinstruments: yearly progress report to the Yamaha Corporation, Music & Cognition Group, MIT Media Lab, 1991.

[46] Gershenfeld, N. Sensors for Real-time Cello Analysis and Interpretation. Proceedings of ICMC, Montreal, 1991.

Interactive Magazines

[47] *Nautilus*, CD-ROM Multimedia Magazine, published by Metatec Corporation, Dublin, Ohio.

[48] *Verbum Interactive*, CD-ROM Multimedia Magazine, published by Verbum Magazine, 1991.

Hand Tremor Measurement

[49] _______, Hand Tremor Measurement, Journal of Occupational Medicine, Volume 25, 6, June 1983. p. 481

Dynamic Constraints Systems

[50] Pentland, A. and WIlliams, J. Good Vibrations: Modal Dynamics for Graphics and Animation. Proceedings of SIGGRAPH 1989, *Computer Graphics*, Volume 23, Number 3, pp. 215-222.

[51] Baraff, D. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. Proceedings of SIGGRAPH 1989, *Computer Graphics*, Volume 23, Number 3, pp. 223-232.

Stylus and Tablet Hardware

[52] Wacom Pressure-sensitive pen and tablet. SD Series. Manufactured by Wacom Technology Group, Inc.

[53] XGT Graphics Digitizing Tablet. Cordless pressure-sensitive pen and tablet. Manufactured by Kurta Corporation.

[54] Calcomp Drawingboard II pressure and tilt-sensitive pen and tablet. Manufactured by Calcomp Corporation.

9.2 References

Bolt, R.A. *The human interface: Where people and computers meet.* Belmont, CA: Lifetime Learning Publications.

Buxton, W. The Pragmatics of Haptic Input. CHI '90 Tutorial Notes, Tutorial 26, Seattle, Washington, 1990.

Visual Languages. Eds. Chang, S, Ichikawa, T., Ligomenides, P. New York: Plenum Press, 1986.

Duisberg, R. Visual Programming of Program Visualizations: A Gestural Interface for Animating Algorithms. I n *Visual Languages.* Eds. Chang, S, Ichikawa, T., Ligomenides, P. New York: Plenum Press, 1986.

Kaprow, A. Graphic Design and the Graphical Interface in the New Media Environment. SIGGRAPH 1991 Tutorial Course Notes.

Laurel, Brenda, ed. *The Art of Human-Computer Interface Design.* Reading, MA: Addison-Wesley, 1990.

The Electronic Design Studio, McCullough, M., Mitchell, W., Purcell, P. (Editors), MIT Press, Cambridge, MA.

Mitchell, W., McCullough, M. *Digital Design Media.* New York: Van Nostrand Reinhold, 1991.

Silverstein, L. *Newspaper Design for the Times.* New York: Van Nostrand Reinhold, 1990.