# Organic Information Design

Benjamin Jotham Fry

BFA Communication Design, minor in Computer Science
Carnegie Mellon University
May 1997

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 in Media Arts and Sciences at the
Massachusetts Institute of Technology
May 2000
[June 2000]

Benjamin Fry
Program in Media Arts and Sciences

John Maeda
Sony Career Development Professor of Media Arts & Sciences
Assistant Professor of Design and Computation
Thesis Supervisor

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

# Organic Information Design

Benjamin Jotham Fry

BFA Communication Design, minor in Computer Science
Carnegie Mellon University
May 1997

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 in Media Arts and Sciences at the
Massachusetts Institute of Technology
May 2000

## Abstract

Design techniques for static information are well understood, their descriptions and discourse thorough and well-evolved. But these techniques fail when dynamic information is considered. There is a space of highly complex systems for which we lack deep understanding because few techniques exist for visualization of data whose structure and content are continually changing. To approach these problems, this thesis introduces a visualization process titled *Organic Information Design*. The resulting systems employ simulated organic properties in an interactive, visually refined environment to glean qualitative facts from large bodies of quantitative data generated by dynamic information sources.

Thesis Supervisor: John Maeda
Sony Career Development Professor of Media Arts & Sciences
Assistant Professor of Design and Computation

# Organic Information Design

Benjamin Jotham Fry

*Thesis Reader*

Mitchel Resnick
Associate Professor, LEGO Papert Chair
Epistemology and Learning Group
MIT Media Laboratory

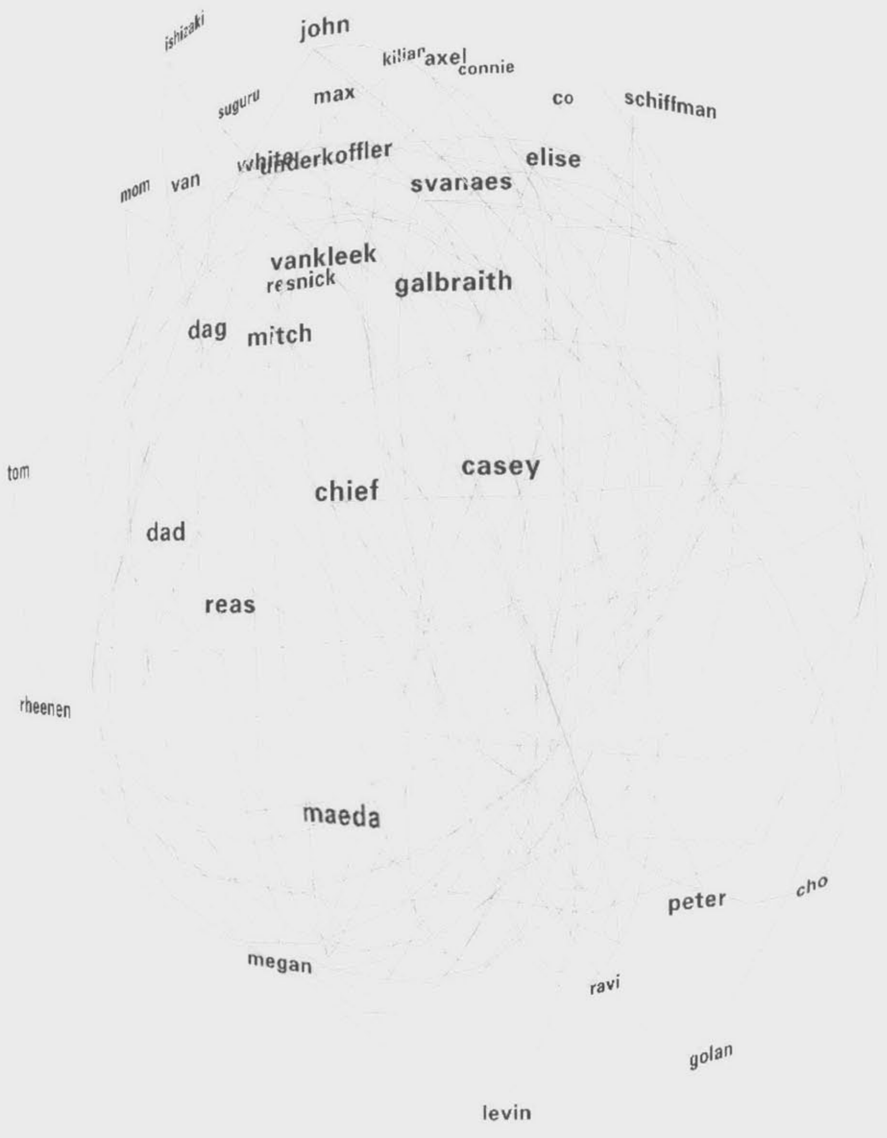
# Organic Information Design

Benjamin Jotham Fry

*Thesis Reader*

Dag Svanæs
Department of Computer and Information Science
Norwegian University of Science and Technology

# Acknowledgements



# Table of Contents

# 1 Introduction

Design techniques for static information are well understood, their descriptions and discourse thorough and well-evolved. These techniques fail, however, when *dynamic* information is considered. Dynamic information is continually changing data taken as input from one or many sources. Changes in the data can be alterations in values, or modifications to relationships within a data set. There is a space of highly complex systems for which we lack deep understanding that could be made accessible through visualization. What does the world economy look like? How can the continuously changing structure of the internet be represented? It's nearly impossible to approach these questions because few techniques exist for visualizing dynamic information. The solutions in this area begin with simple representations—a picture that can be a basis for a mental model. More advanced solutions aspire to full predictability and more objective methods of analysis.

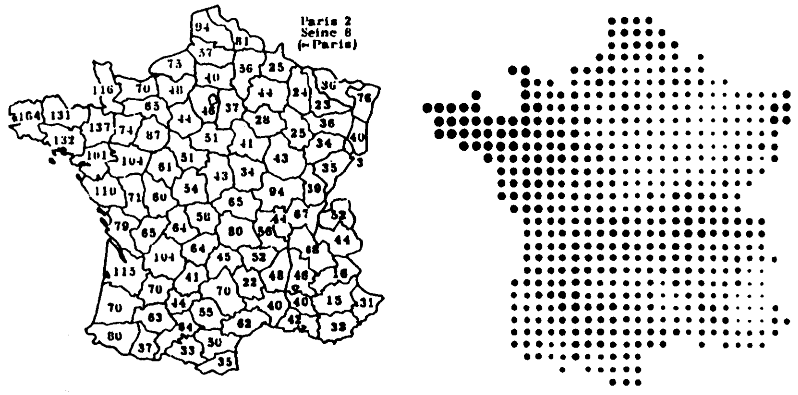
There are multiple reasons for the lack of effective examples for the visualization of dynamically changing structures and values. How can extremely large quantities of data be handled? What happens when the extents and bounds of a data set are unclear? How can a continually changing structure be represented?

To approach these problems, this research introduces a process of creating dynamic visualizations called *Organic Information Design*. This process was developed through the study and analysis of decentralized and adaptive systems, in particular, the traits of simple organisms. The traits: structure, appearance, adaptation, metabolism, homeostasis, growth, responsiveness, movement and reproduction, all relate to a set of features that enable an organism to survive and respond to a complex and changing environment. By examining how these features make an organic system effective, insight is gained into how to design a visualization that responds to and synthesizes data in a similar manner. The result of the design process is an *Organic Information Visualization*, a system that augments the perception of qualitative features of dynamic data.

## 1.1 Qualitative Representations: Relying on Human Perception

Learning the qualitative features of a data set is the first step towards understanding it. The pursuit of qualitative representations is a practical matter, because it will be useless or impossible to consider individual quantities for very large, continuously changing data sets. Impossible because the mind is not capable of handling hundreds of thousands of individual quantities simultaneously. Or useless because only a small amount of the information will actually be useful, and time would be wasted in analyzing the unnecessary parts. Instead, the most important part is a picture that provides the context that will give meaning to specific quantitative values. The second chapter of this thesis describes relevant background on previous approaches.

Because of the accuracy and speed with which the human visual system works, graphic representations make it possible for large amounts of information to be displayed in a small space. A telling example is found Bertin's *Semiology of Graphics* [Bertin, 83] and is reproduced in figure 1.1.1.



1.1.1 *comparison between two modes of representation for sociographic data. left: quantitative version using numbers to depict data. right: graphic version that relies on dots of changing density to depict relative differences*

In this example, both maps describe varying sociographical data throughout France. On the left, numbers are used to represent values, and at the right, the numbers are depicted through changing densities in a pattern of dots. Unlike the image with the numbers, the graphic is immediately readable and quickly makes apparent the qualitative characteristics of the data: a dense area can be seen in the upper-left, with other sparser regions throughout, illuminating less significant values. By making a visual representation for the hundred or so values that construct the map, quickly discernible relationships of the numbers can be obtained.

## 1.2 An Approach to Depicting Complexity: Organic Systems

The third chapter of this thesis describes how properties from organic systems, such as growth, response to stimuli, and metabolism, provide a framework for thinking about visualizations capable of handling dynamic sources of information. There is much to be learned from organic systems because even the simplest organisms deal with complicated stimuli and must adapt to a changing environment. Instead of environmental conditions, organic visualizations use data as stimuli, and their reactions are prescribed in a set of rules crafted by an information designer.

A key feature of organic systems, even synthetic ones, are the psychological phenomena associated with their perception. In *Vehicles: Experiments in Synthetic Psychology*, Valentino Braitenberg elucidates this well:

> *Interest arises, rather, when we look at these machines or vehicles as if they were animals in a natural environment. We will be tempted, then, to use psychological language to describe their behavior. And yet we know very well that there is nothing in these vehicles that we have not put in ourselves.* [*Braitenberg, 84*]

Braitenberg continues with a description of machines whose characteristics seem to evoke emotions or personality traits. A machine that moves away from an object seems to 'dislike' or 'fear' it. Another machine might move towards a similar object with much speed, appearing 'aggressive'. Such a system can be extremely simple: Braitenberg's formulas for the fearful and aggressive machines have just one sensor and one motor. These psychological metaphors can be an extremely powerful tool for constructing an organic visualization, particularly with regards to how they are read by the user, and providing a vernacular for their description.

Emergent characteristics, such as aggregation and coordination, also play an important role in an organic visualization. Interaction rules must be constructed such that behavioral features emerge (i.e. causing a visual clustering of related information). These emergent features often find psychological metaphors. For instance, some elements in the system may 'like' related elements. Others might disassociate themselves from a grouping, appearing primitively 'antisocial'.

## 1.3 The Organic Information Visualization as a Tool for Thought

Organic Information Design is concerned with augmenting one's ability to process large amounts of data. The fourth chapter of the thesis describes a set of experiments that are examples of visualizations implemented with simulated organic properties. They are a starting point for how people can begin thinking about very complicated systems of relationships in a data set.

Complexity is a perceived quality that comes from the difficulty in understanding or describing many layers of inter-related parts. An Organic Information Visualization provides a means for viewers to engage in an active deconstruction of a data set. The complexity is pulled apart through a combination of real-time user interaction as well as control of the data set through modification of the rules used for representation.

A paper by Ben Shneidermann discusses "training and education by exploration" [Shneidermann, 94] and the positive reactions users had with such systems. It states that "the enthusiasm users have for dynamic queries emanates from the sense of control they gain over the database." This highlights the engaging quality of learning about a data set. The Shneidermann work is limited, however, because each visualization must be constructed by a programmer, who also determines the parameters and ranges used for the data. This problem suggests a model where the programming is simpler and accessible to the viewer. In this model, they can become more involved in the creation of representations with features most important to their goals.

## 1.4 Summary of Contributions

This chapter exposes the need for a model of visualization capable of handling dynamically changing information in a flexible manner. The work described in this thesis makes four primary contributions in this area. Each part is delineated as a chapter in the body of this thesis.

*Context and Definitions*—the second chapter characterizes Organic Information Design, based on a convergence of themes from visualization, art, information design, and computer science. The synthesis is placed in the context of previous projects in these respective fields.

*Properties of Organic Systems*—The third chapter describes how properties from organic systems, such as growth, response to stimuli, and metabolism, provide background for a computational framework for visualizations capable of handling dynamic sources of information.

*Experiments in Organic Information Design*—the fourth chapter explains a set of experiments that are examples of visualizations implemented with simulated organic properties. It considers the structures used to construct such a visualization and describes the software model behind them. In addition, it studies the process of creating such a visualization.

*Analysis*—the final chapter describes salient themes of the work, listing its successes and shortcomings. Most importantly, these themes point to future work and continued improvements to the initial model presented.

# 2 Context and Definitions

Visualization as a sub-field of science, statistics, and graphics has only been recognized as its own entity since the mid- to late-80s. The depth of seminal work is in line with that of a young field, but finds its strength in background drawn from years of statistics and graphic design.

A succinct definition of Visualization is found in [Card *et al.*, 99]

> VISUALIZATION–*the use of computer-supported, interactive, visual representations of data to amplify cognition.*

Visualization is concerned with non-abstract data sets, for example imagery from weather data or an animation describing the movement of a fluid. For this kind of data, representations or physical analogues already exist.

> INFORMATION VISUALIZATION–*the use of computer-supported, interactive, visual representations of abstract data to amplify cognition.*

Information Visualization, by contrast, is concerned with making an abstract set of information visible, usually in circumstances where no metaphor exists in the physical world.

The previous two terms can be used as both a verb describing a process or a noun describing an outcome. In order to avoid this ambiguity, this thesis separates the roles by defining Organic Information *Design* as the process used to create an Organic Information *Visualization*. The latter expands on the definition of Information Visualization in several ways:

> ORGANIC INFORMATION VISUALIZATION–*a system that employs simulated organic properties in an interactive, visually refined environment to glean qualitative facts from large bodies of quantitative data generated by dynamic information sources.*

This chapter closely examines this definition, using five sections to relate each of the five parts of the definition to precedents in the fields of visualization, art, information design, and computer science. A summary of the individual sections:

*Simulated Organic Systems*–the method being employed, systems of simple rules that result in a more complicated whole (e.g. cellular automata and decentralized models of programming)

*Interactive Environments*–the means with which a user can learn about a system of data through direct manipulation (dynamic queries, focus+context techniques, other advanced models of user interaction)

*Visual Refinement*–a priority is placed on craft and sensitivity to visual issues, which are too often overlooked in visualization or deemed less relevant and relegated to secondary consideration or worse

*Qualitative Facts from Large Bodies of Quantitative Data*–examples of exemplary work that does an effective job of creating a strong qualitative impression from a large amount of quantitative information

*Dynamic Information Sources*–a discussion of the apparent lack of visualization work capable of handling dynamic information sources

An Organic Information Visualization is a synthesis of these five areas. By examining the successes and failures of previous projects in each area, the process is put in the context the issues that it addresses, and its goals and contributions are made clearer.

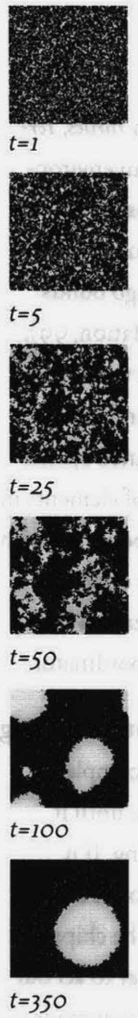
## 2.1 Simulated Organic Systems

The predominant trait of organic systems is their decentralized, distributed structure. The structure can be highly complex due to the interactions of their simpler component parts. Later chapters describe how this type of complexity can be used to create visualizations by causing data to coalesce and organize into structures in a similar manner. Computational models such as cellular automata are decentralized, rule-based systems that have been used for all manner of simulation, from games demonstrating social phenomena to highly mathematical physics to primitive models of simple organisms.

*StarLogo*—Decentralized, self-organizing systems were examined in *Turtles, Termites, and Traffic Jams* [Resnick, 94]. The text introduced StarLogo, an environment based on the Logo programming language. StarLogo's purpose is the simulation of *massively parallel microworlds*, systems made up of thousands of actors interacting with one another and their environment. StarLogo builds on the simplicity of the Logo programming language [Logo Foundation, 99], enabling developers with a wide range of skill levels to experiment with such distributed phenomena as ant colonies, traffic jams, slime molds and forest fires. Unlike a more general purpose programming language, its tuned syntax makes it straightforward to model the behaviors and interactions of elements in a distributed system. Using such an environment, the user can experiment with the parameters of these systems, observing how changes affect the outcome of the simulation. It provides firsthand experience for how organization can emerge from component parts without the direction of a central coordinator.

Figure 2.1.1 is an example StarLogo program, a simulation of termites collecting wood chips and organizing them into piles. The termites act with complete independence of one another. Each termite moves about randomly, until it bumps into a wood chip, which it picks up and continues wandering. If it bumps into another wood chip, it will find a nearby empty space to set down the chip it was carrying. It then returns to wandering. Eventually, the chips will be collected into a single pile, as if the termites had worked together to act out this explicit goal. However, this 'goal' was never a part of the rules that made up the program, but was instead *emergent* from the interaction of several entities (the termites) acting out the simple set of instructions.

Emergence is an essential strength of decentralized systems. It means that a meaningful whole can be developed from the interactions of many elements acting on very simple rules.



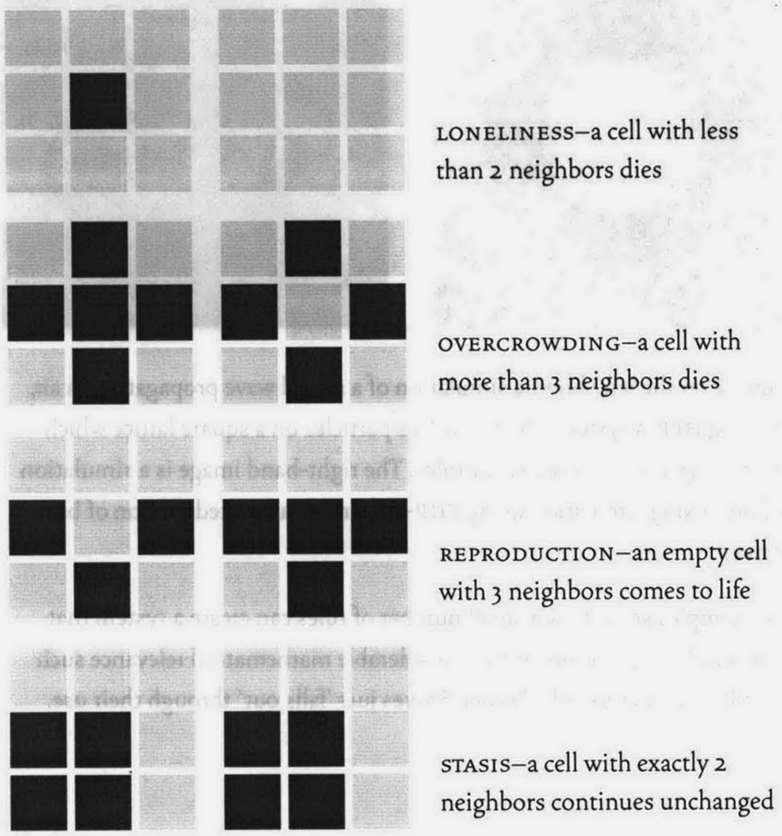
```
to setup
setc red
seth random 360
jump random 200
end

to go
search-for-chip   ;; find a wood chip and pick it up
find-new-pile     ;; find another wood chip
find-empty-spot   ;; find a place to put down wood chip
end

to search-for-chip
if pc = yellow    ;; if find a wood chip...
 [stamp black     ;; remove wood chip from patch
  setc orange     ;; turn orange while carrying chip
  jump 20
  stop]
wiggle
search-for-chip
end

to find-new-pile
if pc = yellow [stop]   ;; if find a wood chip, stop
wiggle
find-new-pile
end

to find-empty-spot
if pc = black       ;; if find a patch without a wood chip
 [stamp yellow      ;; put down wood chip in patch
  setc red          ;; set own color back to red
  get-away
  stop]
seth random 360
fd 1
find-empty-spot
end

to get-away
seth random 360
jump 20
if pc = black [stop]
get-away
end

to wiggle
fd 1
rt random 50
lt random 50
end
```

2.1.1 *termite simulation written in StarLogo, with pictures of the simulation at increasing time steps*

*Cellular Automata*–The first theories on computation with decentralized systems trace back to John von Neumann in 1948, when he gave a lecture on the "General and Logical Theory of Automata." Stanislaw Ulam later worked out these ideas and proposed that distributed systems could be modeled on a regular grid of 'cells', which updated itself according to a set of rules. The rules are local, meaning that the state of individual cells are affected only by a cell's immediate neighbors in the grid. During the 1950s, Arthur Burks continued to extend von Neumann's work and coined the term *cellular automaton*. [Coveney & Highfield, 95]

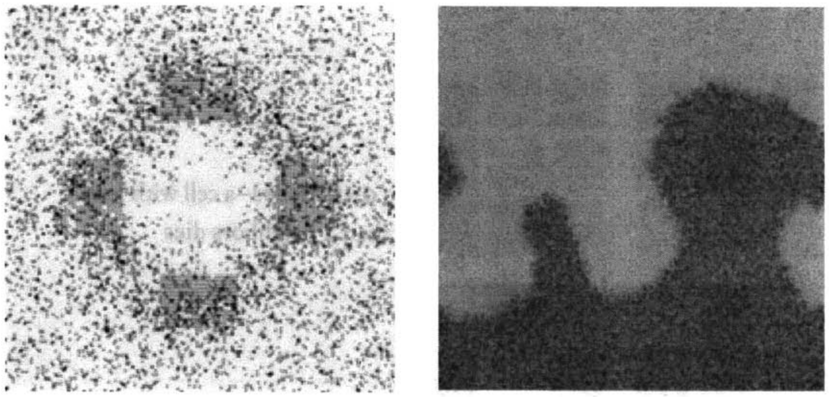
In 1970, the field saw a resurgence when John Conway invented *The Game of Life*, a set of rules for a cellular automaton that simulated a kind of microworld. The simplicity of the four rules (chart 2.1.2) can be deceptive, because of the variety and depth of configurations that can be created. The figures at the right

*2.1.2 rules for Conway's Life–in the examples, only the center element is affected by the rule*


LONELINESS–a cell with less than 2 neighbors dies
OVERCROWDING–a cell with more than 3 neighbors dies
REPRODUCTION–an empty cell with 3 neighbors comes to life
STASIS–a cell with exactly 2 neighbors continues unchanged

depict examples of several possible configurations, including static, periodic and moving objects [examples from Flake, 98]. Using Conway's Life, it is possible to create systems that regenerate and reproduce in a primitive manner. In addition the combination of stable, persistent structures; the ability to 'count' with periodic structures; and the ability to move information qualifies the system as Turing complete, meaning that it is possible to fully simulate the kind of computation done with today's machines.

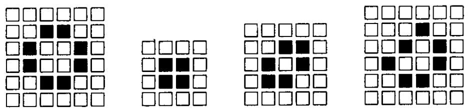
Another notable kind of cellular automaton is a class known as *lattice gases*. These systems use a set of rules not unlike Conway's to model complex fluid dynamics. Using a systems such as HPP and FHP, after [Hardy *et al.*, 76] and [Frisch *et al.*, 86] it is possible to accurately represent both macro and microscopic dynamics of particles which obey the Navier-Stokes equation [d'Humières *et al.*, 87], the basis for most work in fluid dynamics.


2.1.3 *examples of simulating fluid dynamics using lattice gasses. at left, a sound wave propagating through a air; at right, the mixing of two fluids of different densities*

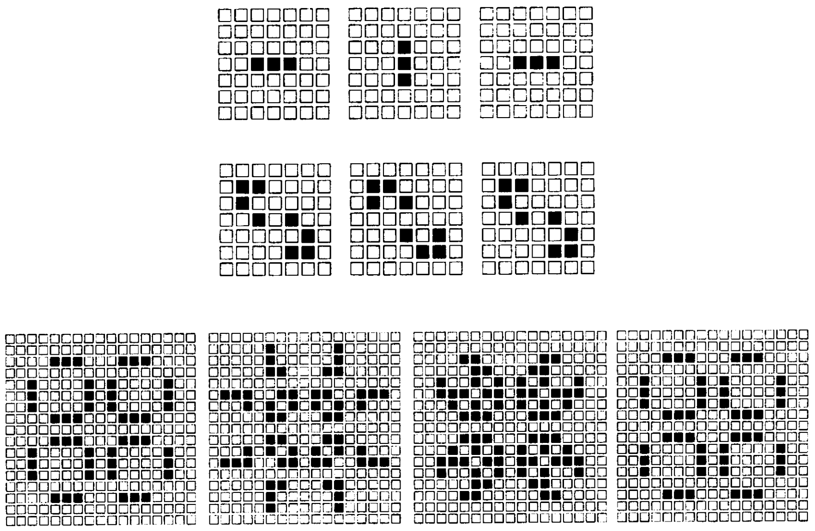
The image on the left shows a simulation of a sound wave propagating in air, built using HPP, a system which involves particles on a square lattice which interact using a very limited set of rules. The right-hand image is a simulation of a fluid mixing simulation using FHP-III, a more advanced version of basic FHP rules.

These examples show how a small number of rules can create a system that emulates full computation, or has considerable mathematical relevance such that a difficult equation like Navier-Stokes just 'falls out' through their use.

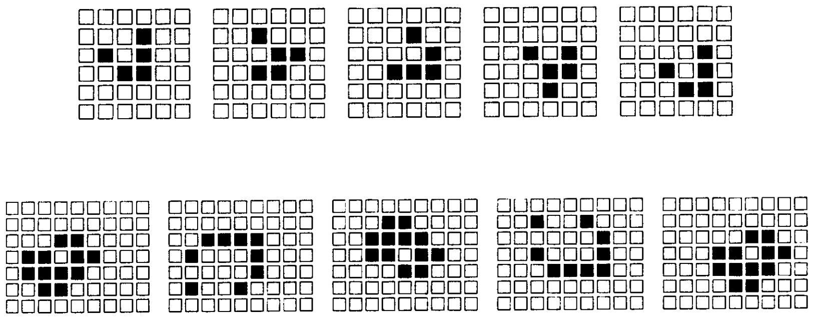
*2.1.4 four structures that will remain unchanged unless affected by other elements*



*2.1.5 three sequences of periodic structures. the last frame of each sequence is the same as the first (i.e. for a 3-frame sequence, there are two steps, the third frame is the beginning of the next sequence)*

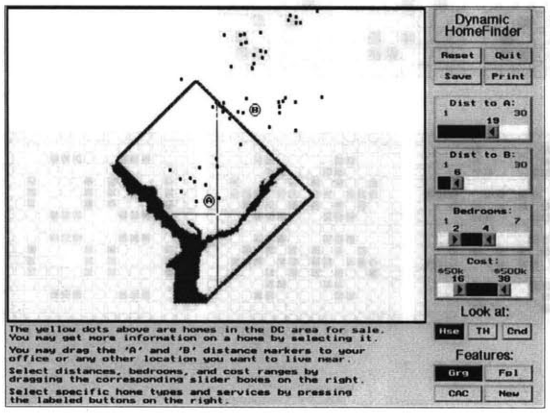


*2.1.6 two sequences of moving 'objects' in life, a set of cells that 'acts together' to propel themselves as a group*



## 2.2 Interactive Environments

Interaction is an essential component of visualization, particularly for enabling the representation of much larger structures by relying on user interaction. The ability to show and hide elements of interest, or to zoom in to a particular area of interest for a more detailed view are capabilities unique to interactive interfaces. Along with number-crunching ability, interaction is the other half of the strength in relying on the assistance of a computer for the task of visualization. Much work has been done in creating software environments for interactively exploring large sets of data. These software tools support a kind of active viewing or detective work.

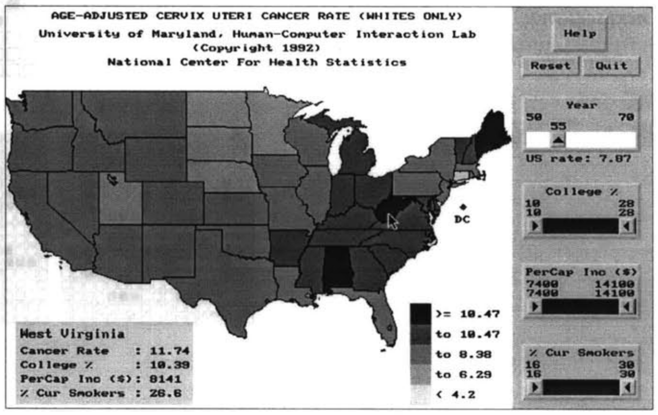


The yellow dots above are homes in the DC area for sale.
You may get more information on a home by selecting it.

You may drag the 'A' and 'B' distance markers to your office or any other location you want to live near.

Select distances, bedrooms, and cost ranges by dragging the corresponding slider boxes on the right.

Select specific home types and services by pressing the labeled buttons on the right.


AGE-ADJUSTED CERVIX UTERI CANCER RATE (WHITES ONLY)
University of Maryland, Human-Computer Interaction Lab
(Copyright 1992)
National Center For Health Statistics

West Virginia
Cancer Rate : 11.74
College % : 10.39
PerCap Inc ($): 8141
% Cur Smokers : 26.6

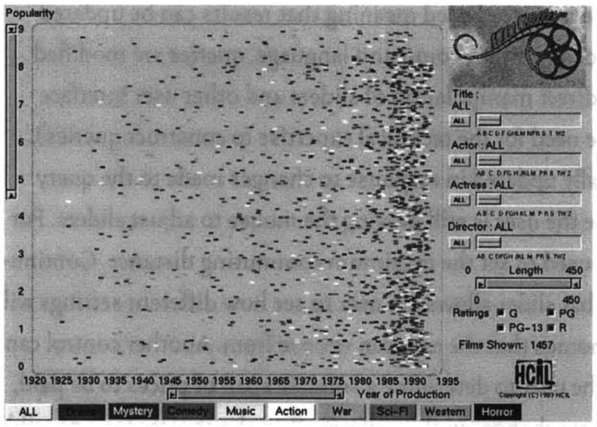
2.2.1 *Dynamic Home-Finder*

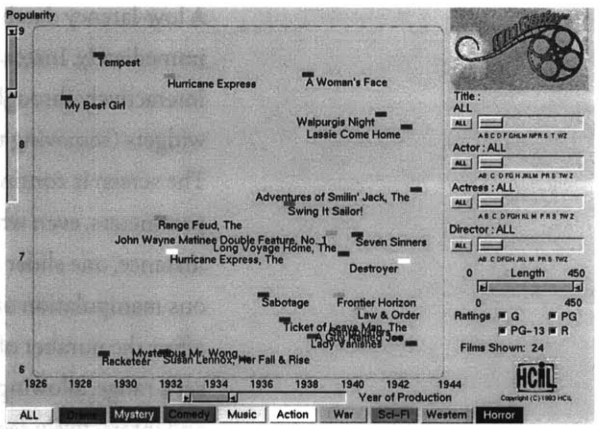
*Dynamic Queries* – in 1993, Shneidermann and his students at the University of Maryland first presented work associated with a well-formed model of direct manipulation [Shneidermann, 94]. The image on the left depicts an interface that allows the user to search for homes based on criteria such as price, number of bedrooms, and distance for one's commute. Previously, when interacting with a database, a user would choose a set of criteria, then use that as a query to be sent to the database. Constructing this type of query requires a great deal of expertise, making it inaccessible to all but the most advanced user. After a delay for the query to process, the results of that query would be shown on the screen. Dynamic queries make several improvements to this model.

A low-latency database is instead used meaning that results can be updated immediately. Instead of a complex command language, queries are modified interactively through direct manipulation of sliders and other user interface widgets (removing the need for the technical expertise to construct queries). The screen is continually updated in response to changes made to the query parameters, even while the user is still moving the mouse to adjust sliders. For instance, one slider is used to set the maximum commuting distance. Continuous manipulation of that slider allows the user to see how different settings will affect the number of homes that the user can choose from. Another control can set a range, allowing the user to determine a suitable span of prices to be paid, and interactively see how changes in this criteria affect the choices for a possible home. Another application example (shown in the right-hand image) uses a similar system to examine a database of medical criteria related to cancer rates sorted geographically.

Dynamic queries are a good example of what's possible when employing the computer in exploratory data research, and the work came in tandem with a time when the machines being used in the research laboratories were becoming fast enough that such a system was plausible. It demonstrates the effectiveness of using the computer to rapidly prune through large amounts of information.

It is difficult to argue with the basic concept of dynamic queries, it is such a simple idea that they are without a doubt extremely pervasive. Perhaps the most limiting factor not addressed in the work of Shneidermann or his students is that the types of systems for which the queries are effective is limited by things that can be represented as a singular instance on-screen. In particular, they make heavy use of metaphors. For beginning users (perhaps the target audience for these applications) this might make sense. However, as mentioned earlier, simple metaphors are often lacking in information visualization, so the scope of their use can be quite limited. Most data will not have simple physical correlations (a map of a downtown area, a map of the United States) that make sense, the way they do in the examples shown here.





*2.2.2 FilmFinder by Christopher Ahlberg and Ben Shneidermann*

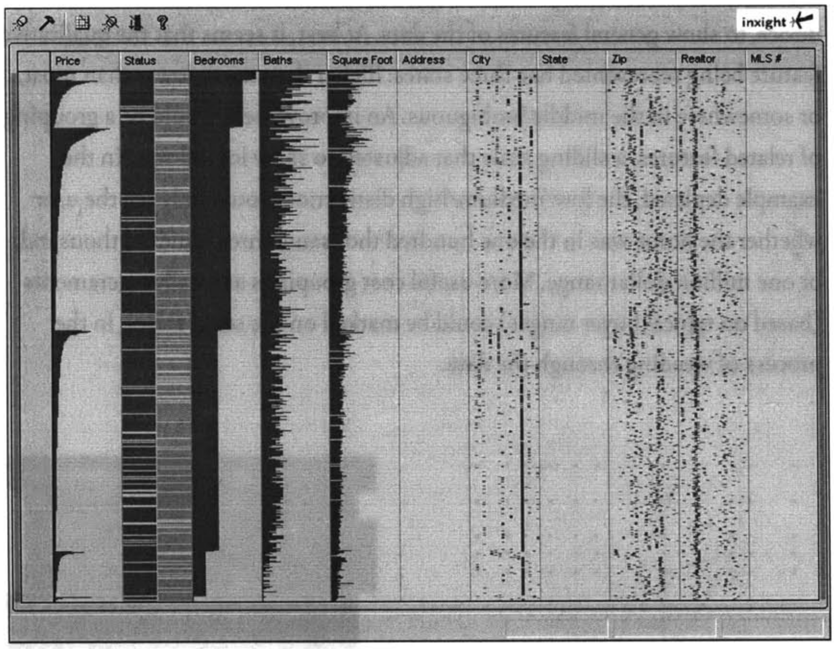
*Starfield Displays* –closely related to the previous projects with dynamic queries work is the FilmFinder [Ahlberg & Shneidermann, 93], a project by Christopher Ahlberg, at the time a student of Shneidermann's. It relies on a widget called the Starfield Display, which enables applications with much larger data sets.

Having chosen to directly discretize axes (one-for-one mapping of an axis to a dimension of data), this piece is limited by the number of axes it can represent. This too quickly leads to a cluttered interface that attempts to support many axes, with both continuous (year) and discrete (genre) selections.

Each axis is linearly spaced, evidenced by an apparent need for a logarithmic scaling on the year axis. It's perhaps interesting to see the change of density in number of movies over the years, but the attraction seems fading, past the initial glance. As a result, the already cluttered interface suffers from poor use of space.

The representation used is primitive, using small blocks of highly saturated color for instances of data, a form fails to be evocative of the data being presented. The lack of care seems as though it could potentially distract from the data being represented.

It is also unclear if the application (a film finder) is particularly relevant to how people would want to access this particular kind of data (movies). However, I believe the author's intent was to explain an idea, which stands on its own in spite of the application.



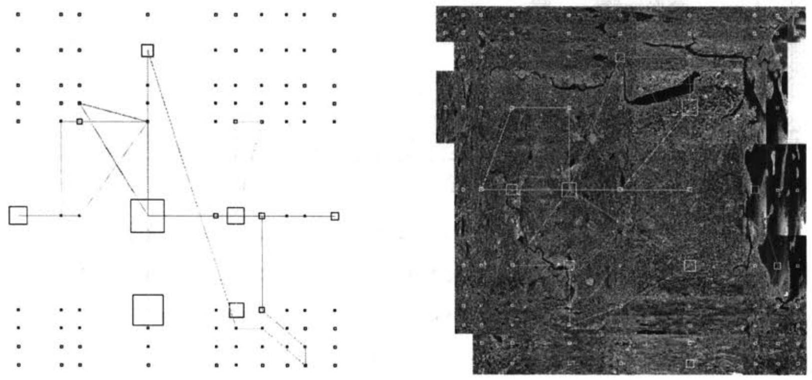
2.2.3 *Table Lens implementation from Inxight software*

*Table Lens* – The Table Lens, reproduced in figure 2.2.3, is a system created by Ramana Rao and Stuart Card at Xerox PARC [Rao, 94]. It is an interface that allows the user to view multiple dimensions of data for easy sorting and visual correlation. A positionable lens provides the ability to locally zoom in to particular sections of data, based focus shifts by the user.

The concept of an adjustable lense providing *focus+context* is extremely useful. The general problem of zooming into a set of data is that all context is lost because the zoom moves everything out of the viewing area except the actual targeted area.

The shortcoming in this approach, however, is that it assumes that all of the data is equally relevant, requiring the user to spend much time weeding through the data set. The work of Axel Kilian, described in the next section, is an example of a significant improvement on this model.

In addition, the reliance on a heavily quantitative view based on devices like bar charts shows how such devices break down when applied to large volumes of data. The strength of a bar chart is being able to compare a handful of quantities against one another, and determine their relationships semi-explicitly through the use of its numeric scale. But here it is used as a pseudo-qualitative device, to show general features of the data. At best, it seems that the qualitative feature being represented has three states: higher than most, lower than most, or somewhere in the middle/ambiguous. An improvement would be a grouping of related features, a sliding scale that adjusted to show logical sets. In the example depicted, the low/medium/high distinction would only tell the user whether the home was in the one hundred thousand, three hundred thousand, or one million dollar range. More useful cost groupings at smaller increments (based on typical buyer ranges) could be marked on the scale, aiding in the process of weeding through the data.



2.2.4 *space-warping with interaction history by Axel Kilian. at left: structure only, at right: interaction model applied to a photographic map*

*Axel Kilian*–Kilian's thesis work is an in-depth study in the use of nonlinear space [Kilian, 2000]. His studies in architecture led him to an interest in how software unchained the designer from the spatial restrictions of the physical environment. He explores this theme through a number of smaller pieces, each playing with a related sub-theme of this idea.

One of his particularly successful sub-themes has to do with storing a kind of 'focus history', based on the interactions of one or several users with a data set over time. The result is for multiple parts of the visual composition to maintain some of their prevalence even when not the primary focus. And when returning to previous focus locations, they are allowed to more quickly regain their previous focus state. This concept of residue from interaction over time is especially useful as he applies it to large sets of information.

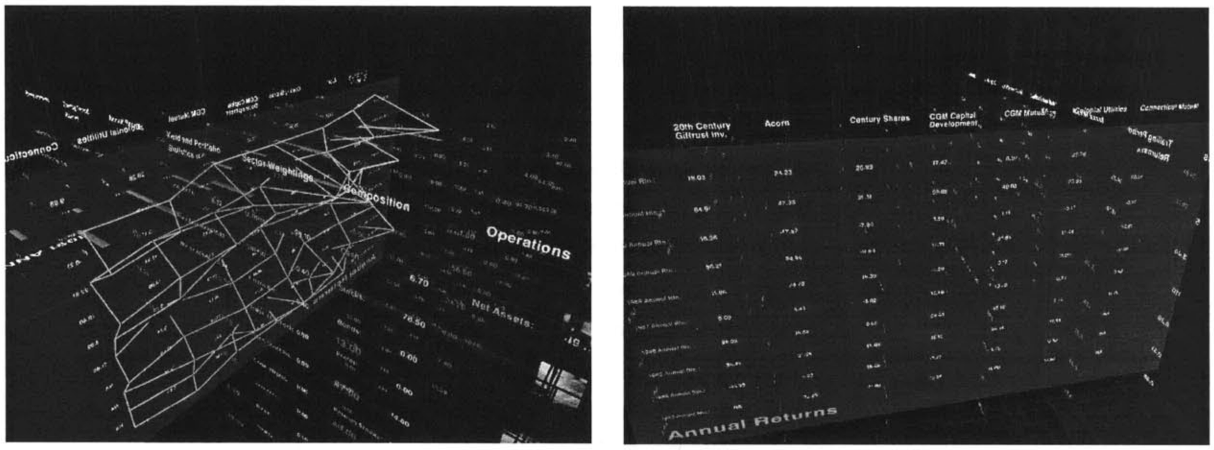
## 2.3 Visual Refinement

The following projects are examples where visual refinement has a significant positive impact on the outcome of the piece. In each of these pieces, iteration of the visual design was given high priority. A large amount of information visualization work seems to overlook the importance of a clear and elegant appearance as intrinsically linked to the usefulness of the system. There are many reasons why this is the case, and it likely has much to do with how a visualization is built. There is a wide gap in the quality of visual design for the printed page versus software interfaces.

The graphic design of software-based visualization appears to have consistently lower quality than that of printed materials. Print design is often enacted by a graphic designer, and software is generally implemented by a programmer. For the print designer, tools exist to assist in the creation of highly detailed, well crafted pieces that were developed through heavy iteration. For the programmer, no such tools exist, so it is likely that it will be either 1) not important enough, as the visuals are often seen as a facade, and not functionally dependent on the system, or 2) for the well-intentioned programmer, too hard to implement better visuals, and therefore not enough of a priority.

The only way to address this issue is to start with a design environment where the visuals are intrinsically linked to the visualization itself. The user of this environment has full access to the parameters determining appearance, allowing full control with which to iterate and modify it. In the past, this has been simulated by relying on the programmer to also be a designer, or at least for a designer to work closely with a programmer. Neither of these models are very successful, because for one person to handle the combined process of design and programming using traditional tools becomes extremely tedious, and makes it difficult to execute in either role effectively. Closely matched teams of designers and programmers are problematic as well, because they require intensely coordinated efforts and single-mindedness of purpose and goals.

The background presented here shows examples created through the tedious perserverance of designer-programmers or by teams of designers and programmers. Additional examples are also included that are not built computationally, as examples of possibilities for depicting complexity through a time-consuming, brute force process. For the process of Organic Information Design, however, the goal is for a single person to be able to act as designer and programmer as a combined role, but with some of the tedium of programming removed, through the use of simpler development languages than C/C++ or Java. In addition, with the positive aspects of the available aspects of software tools for design, most notably the ability to iterate on a solution quickly, incorporated.

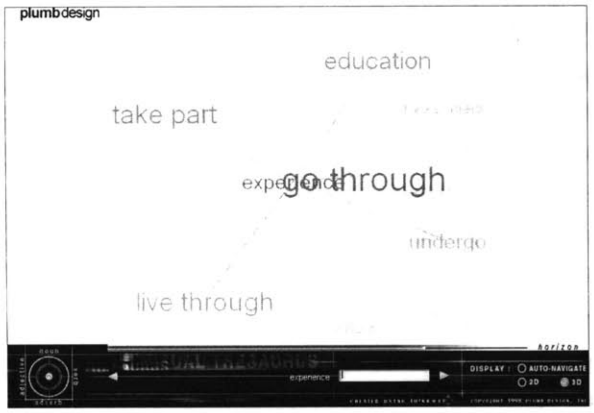


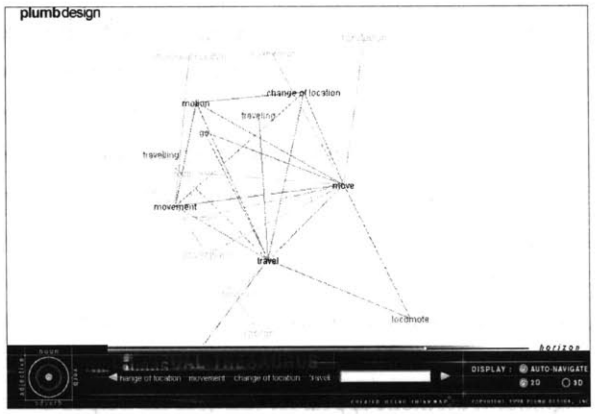
2.3.1 *Financial Viewpoints by Lisa Strausfeld, exploring quantitative data in three dimensional space*

*Financial Viewpoints*—very elegantly designed work by designer-programmer Lisa Strausfeld [Strausfeld, 95] during her tenure with the Visible Language Workshop at the Media Lab. The piece depicts a large amount of multivariate data regarding securities markets. The project uses delicately crafted tables, bar charts, and graphs but combines them in a unique way. Employing three spatial dimensions, the tables and charts (implemented in two dimensional planes) intersect one another where relations exist. To show different relations, the planes are shifted and the tables automatically updated. It is a powerful example of an effective synthesis of older models (charts and graphs) can be resynthesized in a new way, based on what is possible in software.

The limitations of this work are closely related to its strengths. By relying on charts and graphs, this piece fails in the same way that they do, namely that the extent of the data being represented is limited by the linear space they are presented in, and the number of dimensions that can be expressed in the relations shown are limited to the 3-dimensional environment.

This piece provides a telling example for the pitfalls of attempting to extend existing visual metaphors, and the reader will note that the experiments shown in Chapter 4 avoid these older models altogether. In addition, it becomes necessary, when presenting large amounts of data, to employ nonlinear spatial design.





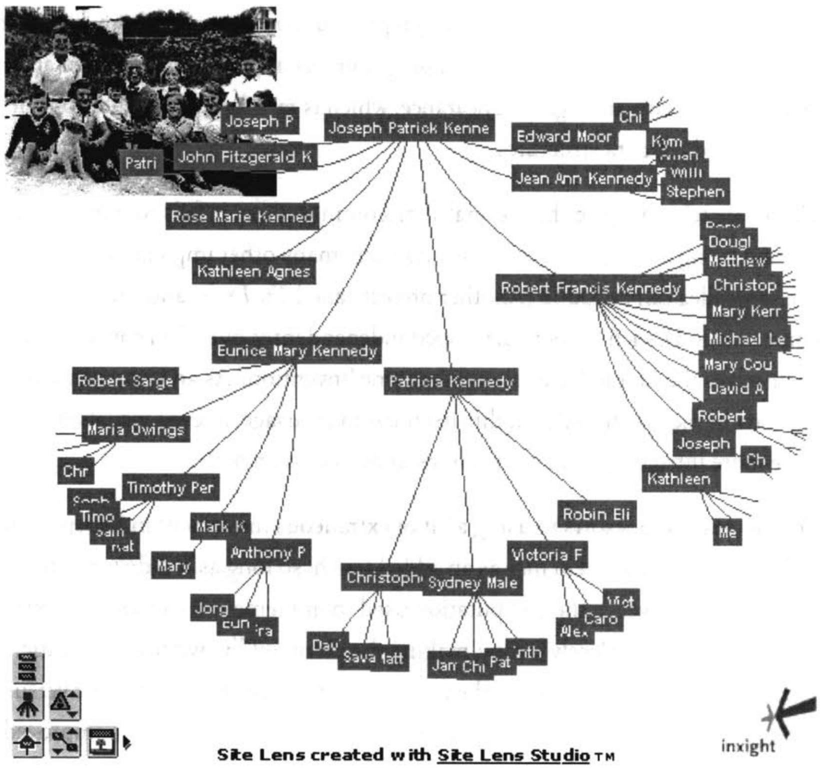
2.3.2 *two modes in the Visual Thesaurus by plumbdesign*

*Visual Thesaurus*—an interactive visual 'fly-through' thesaurus. Mostly, this piece is a visual representation of a directed graph structure, a concept from first-year computer science. The designers and programmers at plumb have created an environment with an elegant appearance, which is rare for implementations of similar kinds of data structures.

Unfortunately, this piece has several shortcomings. First, it fails to move much beyond being a visual directed graph. As such, many other implementations of the same idea can be found (Another project called *The Brain*, and several others come to mind), each of them developed independently, but often considering themselves the first to have created the same 'invention.' As an improvement, the data in the graph could modify itself based on usage, or even visuals that were more intriguing than simple lines drawn between nodes.

Second, the piece resorts to using a bit of extraneous movement in its representation, which would seem nice as an added touch, so long as it didn't interfere with its use. However, the computation used to implement it causes the movement to appear needlessly jumpy, making the piece mildly twitchy and erratic. It could be a simple problem of the implementation using second-order versus third-order equations to control movement. It's also possible that an improved mathematical integrator (see the discussion of Runge-Kutta in [Gershenfeld, 99]) could aid in smoother dynamics, perhaps with less taxing computation. Taking care with movement is extremely important because it is capable of so much expressive power (see discussion in section 3.8).

*Hyperbolic Geometry*–non-Euclidian geometry has been something of a trend for information visualization for the past five years, following a handful of published papers surrounding the subject, most notably [Lamping & Rao, 94] from Xerox PARC. Hyperbolic geometry is based on a space whose coordinates increase exponentially, rather than linearly. Therefore it is possible to condense a large number of nodes into a small space. The user interacts with the graph by shifting focus between different nodes in the system. When implemented well, it is possible for the user to maintain context as they voyage through the space, because most elements maintain their visibility. This geometry also has significant aesthetic appeal for many, as its warped and circular forms can be quite elegant.



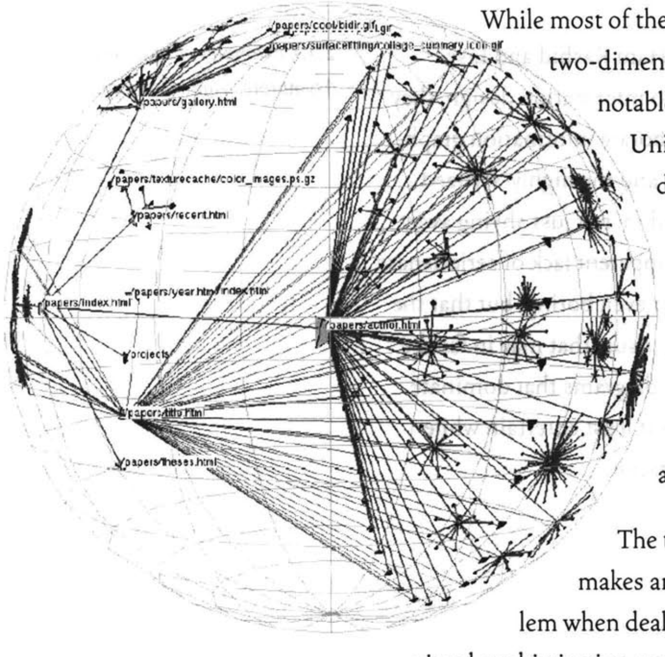
2.3.3 *Site Lens Studio, a software product based on the hyperbolic geometry research at Xerox PARC*

Figure 2.3.3 shows an example of using hyperbolic geometry to explore the Kennedy family tree. Clicking a node will move it nearer to the center, and re-disburse the other nodes along the outer edges. The image is taken from a demonstration of the product *Site Lens Studio*, from Inxight Software, a company founded by many of the original researchers from PARC.

The Inxight demo is mildly confusing to use. When a new node is selected, there is a slight pause while the system is recalculating the geometry, then in a flicker, the new layout appears. Because there is no transition state, it is difficult to maintain context as focus shifts. An interpolation of positions, even if implemented in linear space due to the computational requirements of hyperbolic coordinates, could help this piece significantly.

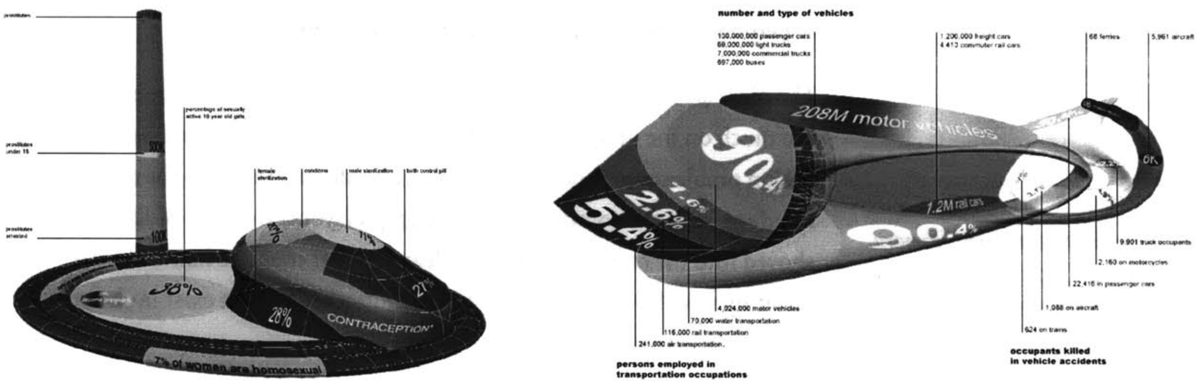
The piece also seems to suffer from a lack of attention to detail, perhaps relying too much on the uniqueness of the representation to overcome its distracting design problems with poor use of color, a confusing image in the corner that interferes with the composition of the graph layout, and a set of toolbar icons shoved into the corner that relate to features that could be handled better. The resulting visual clutter only adds to any clutter that might exist in the graph representation itself.

2.3.4 *Graph generated with Tamara Munzner's H3, software that employs hyperbolic geometries for representing large graphs in 3D hyperbolic space*



While most of the work in hyperbolic geometry has focused on two-dimensional (flat) space, Tamara Munzner's work is a notable exception. Munzner is a researcher at Stanford University who is studying ways to lay out large directed graphs, most recently using hyperbolic geometry in 3-dimensional space. The resulting structures seem somehow related to Valence, one of the projects described in the Experiments chapter of this thesis. This is a fleeting connection, however. On closer look the implementations and outcomes are quite different from one another.

The thoroughly researched work uses an approach that makes an effective use of space—a particularly difficult problem when dealing with graph layout, compounded by 3D. The visual sophistication comes from the approach and algorithm used to build this piece. However the visual details could be improved significantly. The arrowheads used to indicate directionality call more attention to themselves than necessary, and the use of straight lines to reach points laid out in a hyperbolic space make the composition considerably noisy, In fact, it would be difficult to determine that this were in fact a three dimensional sphere if there weren't an outline demarcating the boundary of the space. And without the queue to help describe the geometry being used to create the form, navigation would seem quite difficult.



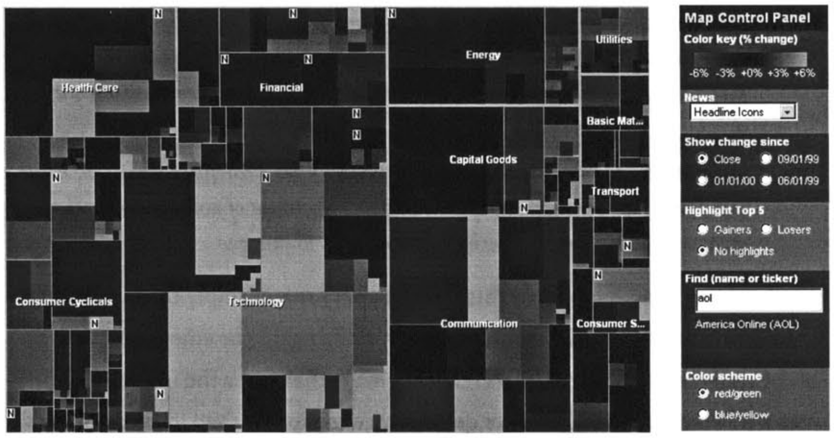
2.3.5 *examples of diagrams by Asymptote, from [Wurman, 1999]*

*Asymptote*—an architecture firm in New York, led by Hani Rashid and Lise Anne Couture with a remarkably unique approach to information design, they employ high-end 3D graphics software to create elaborate information graphics. The attractiveness of this work is hotly debated, and the highly-stylized structures that result are ostentatious without abandon, not just shying away from readability, but avoiding it aggressively. Their apparent lack of seriousness suggests that simple diagrams need not be simple or even boring, but that the same data can be presented in an extremely striking visual that may in fact be more memorable than the approaches to charts and diagrams that dominate post-Tufte graphic design. This is a useful notion to at least entertain while considering new approaches to information visualization.

## 2.4 Qualitative Facts from Large Bodies of Quantitative Data

Like the example from Bertin in the first chapter, the projects that follow all use uniquely effective approaches to create a qualitative visual narrative from a large amount of quantitative data.

*MarketMap*—Martin Wattenberg's MarketMap is a well-implemented approach to revealing qualitative trends in the stock market. The piece is sectioned into regions representing different sectors of the markets (i.e. agriculture, technology). Within those regions, rectangular sections represent individual stocks, and are colored based on the performance of the stock. Upwards movement is shown in shades of green, downwards in red. A bad day in the markets reveals a screen that is bathed in red, an extremely strong depiction of the negative activity that evokes an immediate psychological reaction. It's hard to argue with a representation that induces emotions from quantitative data.



News: Headline Icons

Show change since: [✓ Close] 09/01/99 01/01/00 06/01/99

Highlight Top 5: Gainers Losers [✓ No highlights]

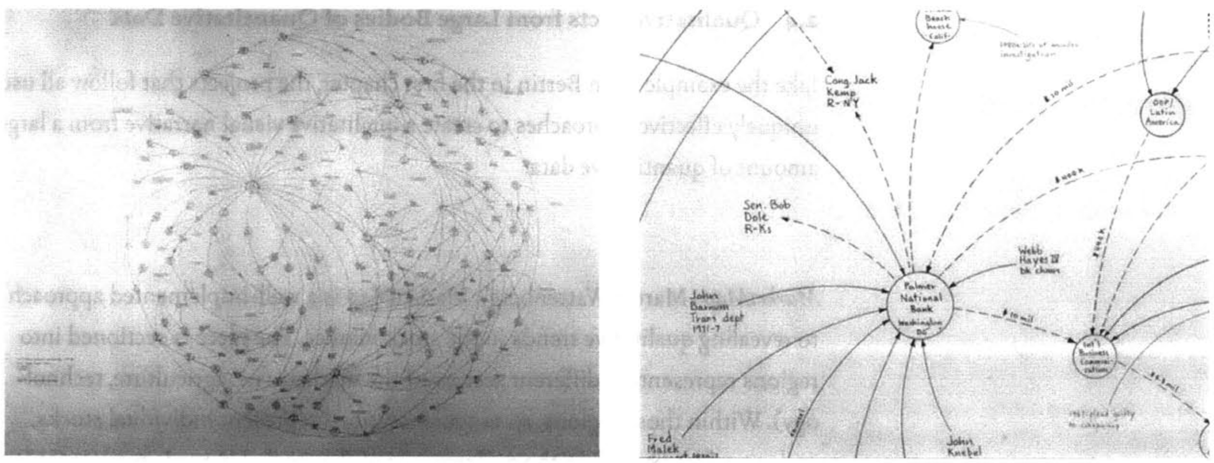
Find (name or ticker): aol

America Online (AOL)

Color scheme: [✓ red/green] blue/yellow

2.4.1 *SmartMoney's MarketMap, by Martin Wattenberg*

Perhaps also impressive is that Wattenberg, who is Director of Research and Development at SmartMoney.com, was able to introduce a somewhat unorthodox visual representation, which is a stark contrast from the market graphing applets and charts used on related web sites. It's a considerable divergence from the norm for a potentially conservative audience of financial advice-seekers on SmartMoney.com.



2.4.2 *one of Lombardi's "narrative structures", and a detail of the piece*

*Mark Lombardi*—the late artist from New York who created highly structured, hand-drawn graphs of inter-relating parts called "narrative structures." Lombardi's description follows:

> *...each consists of a network of lines and notations which are meant to convey a story, typically about a recent event of interest to me like the collapse of a large international bank, trading company or investment house. One of my goals is to map the interaction of political, social and economic forces in contemporary affairs.*
>
> *Working from...published accounts, I begin each drawing by compiling large amounts of information about a specific bank...After a careful review of the literature I then distill the essential points into an assortment of notations and other brief statements of fact, out of which an image begins to emerge.*

The final line is telling of how this work is a very raw example of using a large amount of data in a visual environment and relying on the emergent qualities of the individual visual parts to create an image that turns the individual parts into a flowing narrative, a clearer qualitative description than its component parts. It is commendable that this work was executed in such painstaking detail, and without computational help.

## 2.5 Dynamic Information Sources

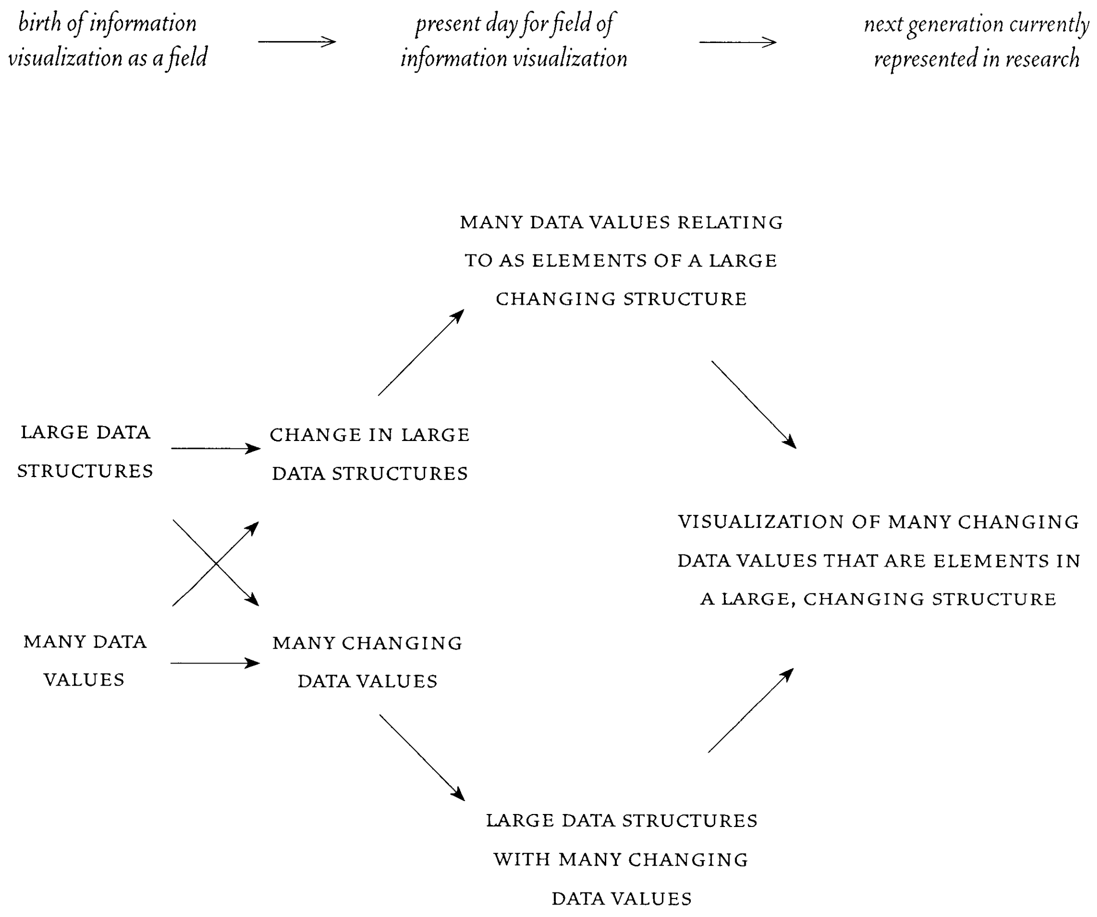
Useful examples of visualization research done with dynamic information sources are somewhat sparse. Small examples exist, for instance Wattenberg's MarketMap is an example of using a live stock feed as a dynamic data source. But a perusal of [Card *et. al*, 99] produces a disappointingly limited (as in, near zero) set of papers regarding research in this area. There are examples of using a 'current' set of data (such as a file browser), but next to none that use a stream of data that is fed to a visualization.

It seems that the majority of visualization work is done offline, likely for three reasons. First, that the computational power required to synthesize large amounts of data make it prohibitive to execute these systems in real time in most instances. Second, that it is considerably more difficult to develop a visualization that can handle a live stream of data instead of a canned database or flat file. Third, the combination of these two factors creates an environment where it is simply not worth the additional difficulty of making a visualization dynamic. Organic Information Visualization begins with dynamic information sources as its basis, in order to address this lacking of research in this area.

The field of information visualization was born out of the need for methods to represent large data structures or many large numbers of data values. Examples of large data structures include hierarchical file systems (many documents and applications inside nested folders) or tree-shaped web sites (linkages on the scale of thousands of pages). Perhaps the most prevalent example of many data values is financial or economic data.

Having developed solutions for the two areas, the notion of expressing many changing data values and change in large data structures had to be addressed. The two areas were pursued separately, because their combination was too complex. A common example of many changing data values is found in the many representations of the stock market, beginning with the stock ticker and later evolving to tools like the MarketMap. Few examples exist that cover changing structure in data.

With primitive models for all four areas, a crossover is now occurring, as depicted in figure 2.5.1. The visualization community is beginning to address large data structures with many changing values (i.e. more advanced models of economic markets), as well as many data values and their relation to a changing structure (i.e. tracking network traffic patterns). The next step comes with models that are capable of representing large numbers of changing values as members of a large changing structure—the next generation work addressed in this thesis.


birth of information visualization as a field
present day for field of information visualization
next generation currently represented in research
LARGE DATA STRUCTURES
MANY DATA VALUES
CHANGE IN LARGE DATA STRUCTURES
MANY CHANGING DATA VALUES
MANY DATA VALUES RELATING TO AS ELEMENTS OF A LARGE CHANGING STRUCTURE
LARGE DATA STRUCTURES WITH MANY CHANGING DATA VALUES
VISUALIZATION OF MANY CHANGING DATA VALUES THAT ARE ELEMENTS IN A LARGE, CHANGING STRUCTURE

2.5.1 *evolution of issues addressed by the field of information visualization*

# 3 Properties of Organic Systems

This chapter describes the properties of organisms, and begins a description in broad terms of how similar systems can be implemented computationally, when applied to information visualization. More specific details on this implementation are described in the fourth chapter, where a formalized mode of implementation is presented along with a set of experiments.

The sections in this chapter are based in part on a definition of *life* taken from [Villee *et al.*, 89]. By learning how an organism uses these traits to cope with its environment, one can infer how a visualization might take on similar characteristics, as a kind of caricature organic system. The properties listed provide a basis for the necessary components of a primitive organic system. Each of these properties can be simulated by simple rules in a decentralized system. Nine such properties are considered:

*Structure*–aggregation of elements to form more complex structures

*Appearance*–visual expression of internal state

*Metabolism*–synthesis of nutrients for raw materials and fuel

*Growth*–an increase in either scale or amount of structure

*Homeostasis*–the maintenance of a balanced internal state

*Responsiveness*–reaction to stimuli and awareness of the environment

*Adaptation*–adjustments to survive in a changing environment

*Movement*–behavioral expression of internal state

*Reproduction*–the ability of entities to create others like itself

Individual entities in the simulated organic system, called *nodes*, interact in a visual environment based on a set of behavioral rules that are determined by the designer of the system. These behavioral rules map meanings determined by the designer, such as 'importance', to a property like appearance. Based on its importance, the node can modify its appearance, for instance, making itself larger than other nodes in the environment.

In spite of the simplicity of the individual rules, the combination of several such rules results in a sophisticated self-organizing system that can adapt to changing conditions presented by the data source. The notion of many simple elements combining to form a more complete whole finds precedents in fields from biology to economics. It is the basis for a decentralized world view, where complex behaviors emerge from a small set of simple rules. The resulting systems can express themselves meaningfully and organize without direction from a central leader.

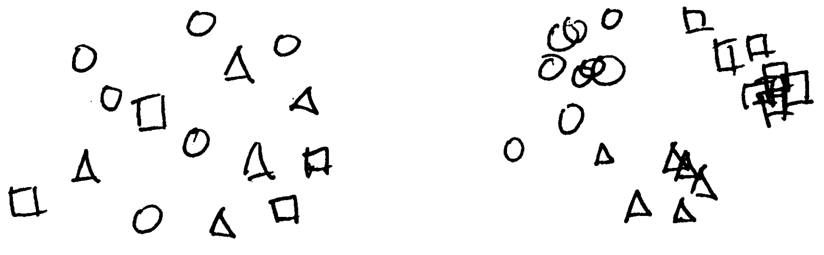
## 3.1 Structure

Organisms have *specific organization*—a cell is made up of organelles, small specialized structure that take care of various tasks within the cell. Specialization increases with tissues (such as muscle tissue) which are composed of many cells. Organs are structured out of tissue, and they perform a specific functional task as part of an organ system. The human digestive system is an example of an organ system. These organ systems make up the whole of a complex organism, through the aggregation of simple parts.

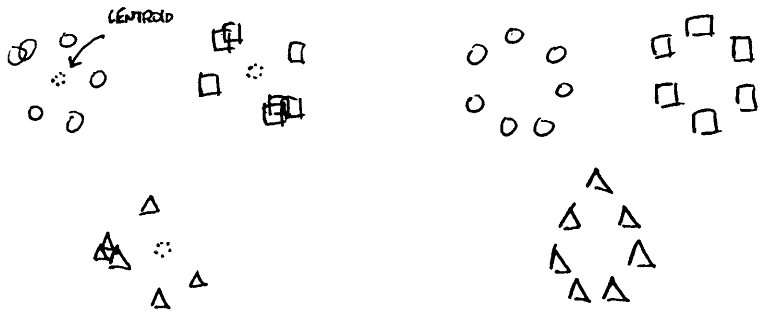
As the levels of aggregation increase, so does the sophistication of the resulting structure. This phenomenon is not limited to organisms. The world economy is made up of many billions of parts, and its description could begin at many levels. One possibility is to start with individual people, each with his or her own interests, abilities and goals. These workers aggregate in the form of a company, which acquires an individual identity. A single company is often part of a conglomerate of companies joined together through mergers or acquisitions. Next, these conglomerates have business relationships with one another, competing with other conglomerates internationally. In spite of the simplicity of the individual elements in the system, each level of aggregation creates meta-elements that are more capable of survival in a competitive environment.

In a computational system, this type of self-structuring has several modes of implementation. A node in a simulation is given a basic awareness of its neighbors (neighbors being defined spatially). The left hand side of figure 3.1.1 shows several nodes that have a single *value*—their shape. A single rule is added to the nodes in the simulation, telling them to move closer to nodes of with similar shape. The result after a few time steps is seen at the right hand side of the figure.

3.1.1 *nodes grouping based on shape*

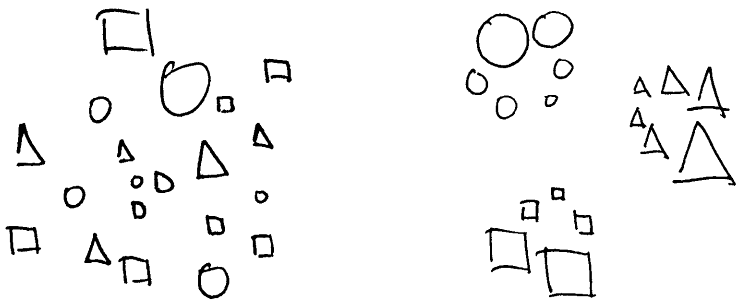


However, the viewer quickly notices that in spite of being closer to similar nodes than dissimilar ones, they are arranged in a haphazard, sometimes overlapping manner. In addition, these individual parts are not guaranteed to be in the direct vicinity of their relations. These problems can be addressed by the addition of two more rules. First, similar shapes should maintain a specific distance from their centroid–their average position in space. This rule causes the parts to group around a point, as seen at the right of figure 3.1.2. The final rule states that each node should maintain a minimum distance from its neighbors. This keeps nodes from overlapping one another, and produces the arrangement seen at the left of 3.1.2.



3.1.2 *similar nodes grouping around a centroid, then adjusting their distances relative to one another*

Several layers of such rules can be used to produce more sophisticated clustering and aggregation of parts. For instance, one might envisage a system that is based on nodes that have not only shape, but a size associated with them (figure 3.1.3). In addition to sorting by shape, another set of rules could organize the pieces based on their size, producing the result seen at the right of the figure. The more complicated ordering adds another layer to the hierarchy of the representation, not unlike the systems described in the earlier part of this section.



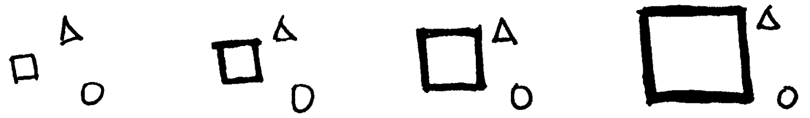
3.1.3 *nodes grouping based on shape, and ordering themselves based on size, the result of a pair of rules*

## 3.2 Appearance

An organism's appearance can express its current internal state. An animal's hair standing on end indicates fright, or a heightened awareness due to a sense of danger. Likewise, organic information systems use appearance as the primary expressive element to indicate the changing state of a system of data.

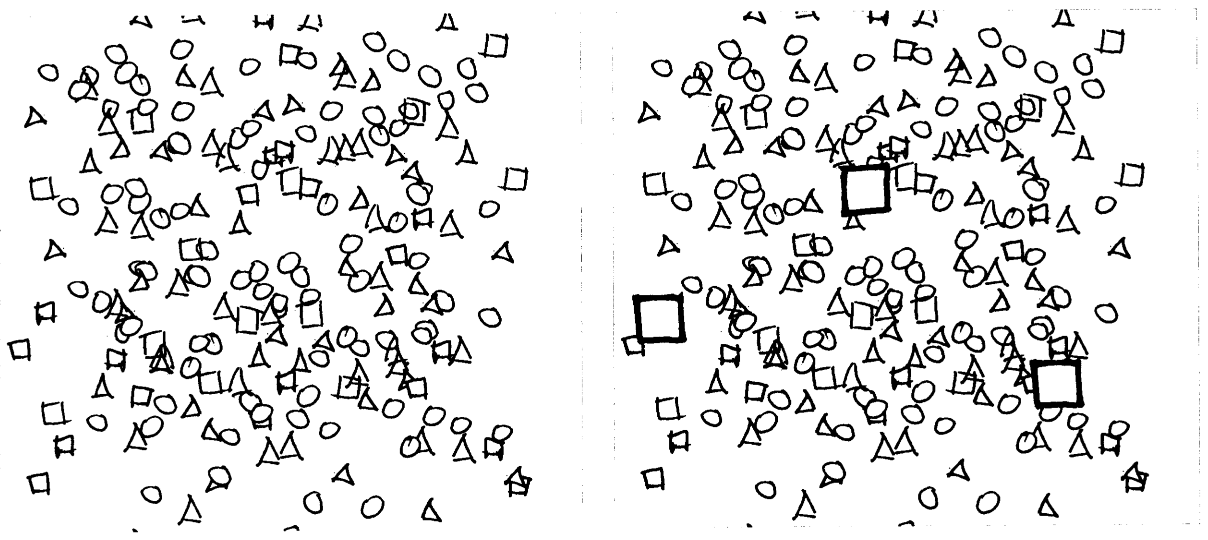
An example of a computational rule for appearance would be a node that changes in size when it is addressed more than its neighbors (the progression in 3.2.1).

3.2.1 *a progression in appearance*



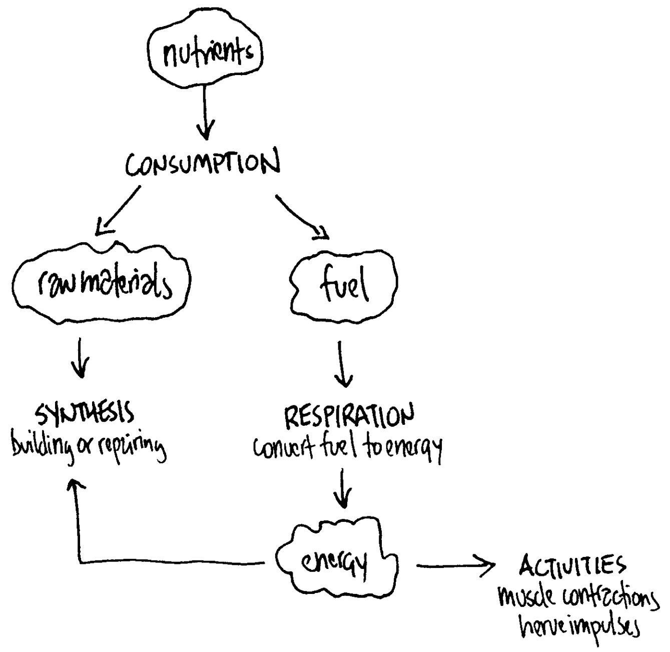
In spite of the simplistic nature of such a rule, its results can be quite striking when applied across a large number of nodes (3.2.2), as individual parts call attention to themselves as they are briefly determined to be relevant.

3.2.2 *a system of many nodes, and the impact of some nodes calling attention to themselves*



## 3.3 Metabolism

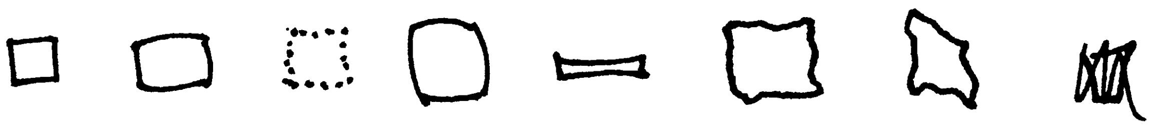
Metabolism is the set of chemical reactions that take place in an organism, relating to consumption of nutrients for raw materials and fuel. As part of cellular synthesis, the raw materials are used for building or repairing a cell. Through cellular respiration, nutrients for fuel are converted to energy to drive the synthesis as well as other activities of the cell such as muscle contractions or nerve impulses. This process is diagrammed in figure 3.3.1.



3.3.1 *metabolic process in an organism, after* [*Villee et al., 89*]

Each reaction is regulated by enzymes, which can control how and when reactions are started, the extent or vigor of the reaction, and its duration.

In a computational medium, the basic blocks of data fed to the organic visualization work like nutrients fed to the system. A set of rules determine whether these blocks of data are *raw materials* or *fuel*. Raw materials are used to build or modify structures. Fuel manipulates the attributes of individual elements. Additional rules can act as enzymes, determining when and to what degree the other metabolic rules should affect the system.



3.3.2 *appearances for comparison, driven by metabolism*

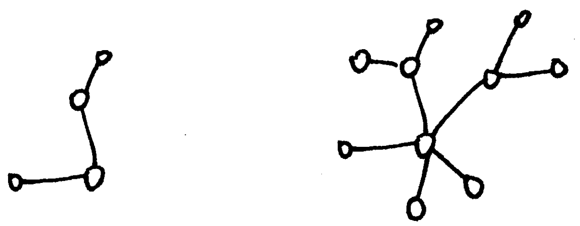
The rules of metabolization are at the heart of what drives the system, the basis for what will result in an appearance that is qualitatively 'well fed' or 'sick'. Distinctions such as these are made by comparing the appearance of individual elements in a visualization against one another (see examples in figure 3.3.2), or by watching the overall behavior of the system over time. The overall behavior might also be compared against mental images of how the system has behaved in the past. Often this process is not an explicit one–it is a natural component of human perception to continually compare and mentally juxtapose. For this reason, the properties of metabolism and appearance are very closely linked.

## 3.4 Growth

Growth in an organism refers to an increase in either the size or number of cells. Some organisms continue growing until they die (such as trees), others have a pre-determined cycle during which the majority of their growth takes place.

For the computational medium, growth generally refers to changes in the underlying structure of the data coming from an information source. Figure 3.4.1 depicts successive stages of growth in a system. New parts are being added while others are slowly being left unused. Along with growth comes atrophy, where individual cells might decrease in size or even die. This is a powerful concept, because it is essential to the ability of an organic visualization to handle changes in the structure of what is being represented. Atrophy allows

3.4.1 *depiction of growth before and after several steps*



elements to wither and die as they become no longer pertinent, without the input data having to explicitly describe when this should happen. Growth will create new structures to replace the old ones as necessary. The process of atrophy is slow and non-explicit.

3.4.2 *structure undergoing process of atrophy*

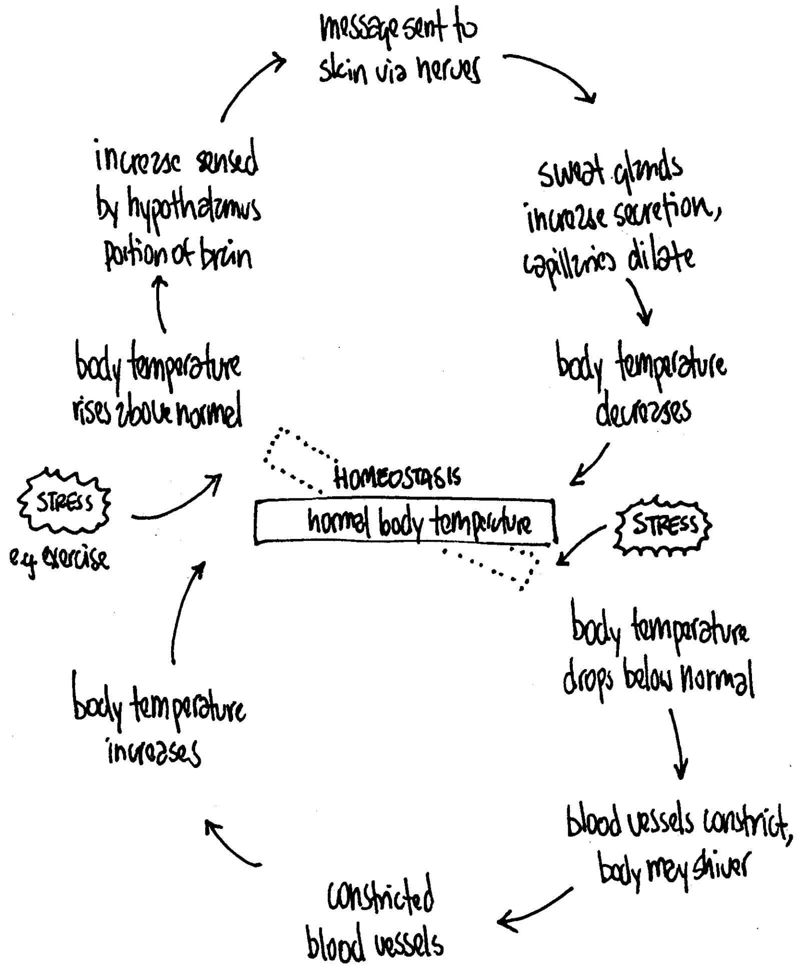
A balance of growth and atrophy is important because the presentation space will always be finite. There are both physical and cognitive limitations that affect the presentation space. It is physically limited by the resolution of the screen (or any other chosen output device). On the other hand, an infinitely large presentation space wouldn't be useful because our cognitive abilities would not be able to handle a much larger space. Instead, atrophy can be used to limit the outer bounds of a visualization, and to weed out less useful parts that are no longer in use.

## 3.5 Homeostasis

Homeostasis is the collection of a set of mechanisms that maintain the balanced internal state of an organism. Figure 3.5.1 depicts the regulation of body temperature in a human through homeostasis. The description that accompanies the diagram [Villee *et al.*, 89] follows:

> *An increase in body temperature above the normal range stimulates special cells in the brain to send messages to sweat glands and capillaries in the skin. Increased circulation of blood in the skin and increased sweating are mechanisms that help the body get rid of excess heat. When the body temperature falls below the normal range, blood vessels in the skin constrict so that less heat is carried to the body surface. Shivering, in which muscle contractions generate heat, may also occur.*

When applied to organic visualization, homeostasis has two modes of implementation. First, rules must be constructed so that they balance themselves,


message sent to skin via nerves
sweat glands increase secretion, capillaries dilate
body temperature decreases
STRESS
body temperature drops below normal
blood vessels constrict, body may shiver
constricted blood vessels
body temperature increases
STRESS
eg exercise
body temperature rises above normal
increase sensed by hypothalamus portion of brain
HOMEOSTASIS
normal body temperature

3.5.1 *regulation of human body temperature through homeostasis, after* [*Villee et al., 89*]

not allowing values to run out of control which might cause the system to 'blow up'. Second, additional rules can be added that maintain the internal balance between the actions of the original rules. For instance, a rule can be added that does not allow forces applied to a node to exceed a certain maximum.

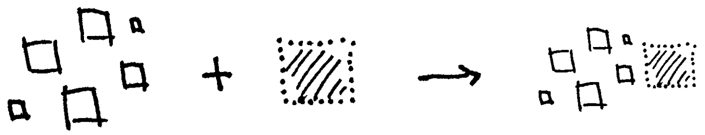
## 3.6 Responsiveness

Organisms are capable of responding to all kinds of stimuli. A stimulus can be one of a variety of environmental changes, such as a change in temperature or a sound. The stimulus and response can be simple, complex, or some combination of the two. One example of a simple response is a single-celled organism that moves away from bright light. For the same stimulus, the pupil in a human eye responds in a complex manner, changing in size as part of a sophisticated system of exposure control. More complex stimuli are also possible. Several beams of light hitting the retina might form the image of a predator, created from the combination of many stimuli. The response to this image can be simple (flee) or complex (evade in some intelligent manner).

In an organic visualization, a large variety of stimulus-response relationships are possible. These relationships take the form of rules, which are connected to the entity that they represent. The rules fall into three general categories:

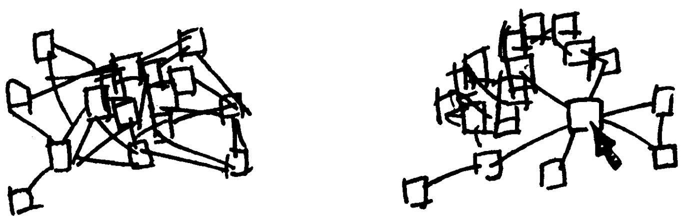
3.6.1 *elements repelling one another*

*Composition Rules*–an entity might take as stimulus the fact that another entity has moved 'too close' to it in the composition space. Appropriate responses would be for the entity to retreat away from the approaching entity, or to exert a force against it in order to repel it like a spring.


3.6.2 *resizing of elements (normalization) in composition, based on a new maximum largest element*

*Data Rules*–addition of new data (often meaning new nodes) to the environment often requires a response from the existing nodes. This response might

*3.6.3 nodes responding to a mouse pointer stimulus. adjacent elements avoid the selected element.*

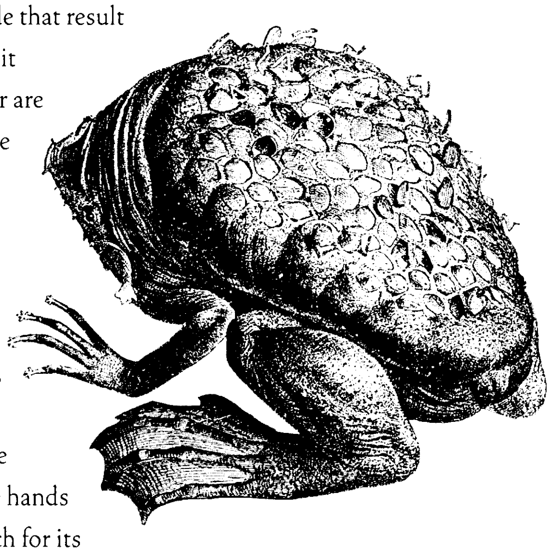


take the form of a re-scaling based on new minimum or maximum values represented by the new data. Other similar statistical phenomena, such as a shift in the balance of the median value, could have similar effects.

*Interaction Rules*—interaction devices (such as the mouse) can also be treated as stimuli. If a click of the mouse means 'focus here', then the system might reconfigure itself based on the new point of focus. For instance, a set of nodes could regroup relative to the position of the node that was clicked. This makes it easier to get more information about the connections from the selected node to its neighbors. Interaction rules become important because they provide the buffer for how the user manipulates a data set.

## 3.7 Adaptation

Adaptation allows organisms to survive in a multi-faceted, changing environment. *Long-term adaptation* is a process that takes place over generations through natural selection. Small mutations in an organism's genetic code that result in a trait that helps an organism survive better than others like it accumulate over the years. For example, animals with longer fur are better able to survive in a cold climate, therefore making it more likely to reproduce, having children that share the same trait. Mutations that cause fur to be longer are an adaptation that helps the species of organism to evolve and survive.


3.7.1 *the Surinam Toad, a prime example of a complicated set of adaptations for a unique environment.*

The image at the right is of a Surinam Toad (Pipa Pipa), a peculiar aquatic creature. It has evolved over many generations, increasing its ability to survive in the mud and murky water of South American rivers. Its eyes have adapted to their lack of use and are tiny spots on the animal's skin. Tiny feelers, almost like hands themselves, augment the ends of its fingers and aid in the search for its food. As a protective measure, the birthing process has adapted such that the eggs of the female Pipa Pipa are placed on her back, and are enveloped in its skin until several months later, when the eggs hatch and fully developed baby toads emerge.

By contrast, *short-term adaptation* covers changes such as the strengthening of a set of muscles due to regular exercise. The muscles adapt by increasing in their ability to handle the repeated stress, each time requiring less effort to do the same task. Short-term adaptations are based on interwoven networks of stimulus-response mechanisms.

Habituation is a second type of short-term adaptation. If a similar stimulus is presented many times, the response on each occasion will diminish as the system becomes habituated to the input. A shrill sound can call an animal to attention, but if such a sound occurs often, each response will be less immediate and less active.

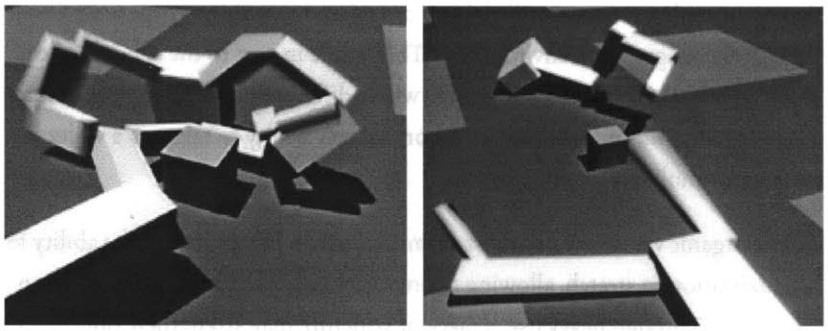
A more complex structure will yield an organism more able to cope with a wider variety of changes to its environment. This is due in part to the specialization that can occur within such a structure, where different parts are able to address more specific stimuli, resulting in the organism's ability to relate to a wider range of stimuli as a whole.

For an organic visualization, short-term adaptation brings with it the ability for a visualization to stretch, allowing the representation to slowly shift based on new input. Through heavy use, a piece of structure may strengthen, only to later weaken as use becomes negligible. Similar to a muscle, a weak part that was once strong will more quickly return to its strengthened state.

Consider as an example, a bar graph that adapts to the information that it is presenting. If the data being 'fed' to the chart required a logarithmic scaling to be useful, the chart would slowly move to that model. This mode of behavior is necessary because a basic assumption of this work is that the parameters of the input stream may not be known, but yet the visualization needs to always reconfigure itself to make the most useful representation.

Habituation is essential as an advanced kind of filter for an organic visualization. If the incoming data stream presents the same information repeatedly, the usefulness of that piece of information is obviously diminished in value, so habituation gives the visualization a way to tune out the constant flow of the similarity. However, a rapid increase or decrease in the rate of that particular stimulus would cause de-habituation, and the item would once again call attention to itself through movement or a change in shape (depending on how the value was being represented).

Long-term adaptation brings with it an entirely different set of criteria, and was studied only minimally for this research. This kind of adaptation occurs over generations, and it is easy to imagine a system where visualizations might be 'bred' for their effectiveness. A set of criteria, including features such as 'clarity of communication' could be assigned to a representation, and the visualization could be scored on these criteria, using human input for the subjective


*3.7.3 from "Evolving Virtual Creatures" by Karl Sims, an example of simple creatures adapted through breeding over several generations.*

decision-making and an algorithm for the parts that could be determined computationally. For example, Figure 3.7.3 comes from a project called "Evolving Virtual Creatures" from Karl Sims [Sims, 94], where he bred simple creatures in software using adaptation, mutation and evolution. The creatures slowly learned how to move themselves, because the mutations started a few creatures moving, and the criteria for breeding was chosen based on ability for self-locomotion—the fastest piece was able to reproduce and continue. The creatures even 'learned' loopholes in the constraint rules set in the software that simulated them—allowing them to achieve artificially high scores until these rules were adjusted.

In general, this direction is covered by a wide body of research, most specifically including work in genetic algorithms, and should be seen as a separate area of study.

## 3.8 Movement

All living things move. Even plants have streaming motion known as cyclosis, allowing them to bend and reconfigure. Movement is important for how an organism is perceived—it's the most basic test an observer uses to determine whether it is alive.

In section 1.2 all of Braitenberg's descriptions of psychological attributions given to machines were related to movement. Movement is the key indicator for what a being is 'thinking'. It helps to describe feelings or intentionality. Of course these attributes are not always present in the system, because machines of this kind are not capable of having feelings or intentionality. Braitenberg's

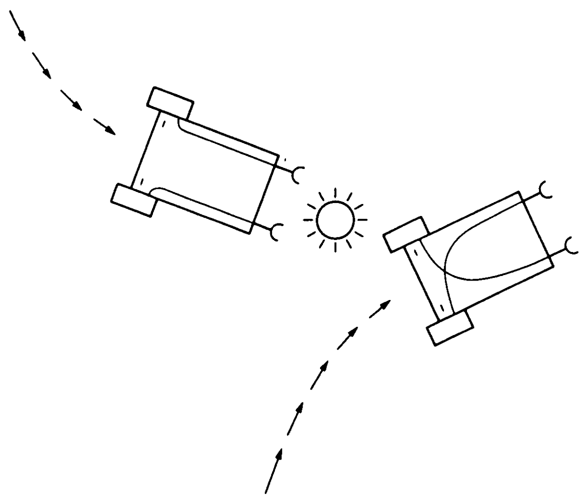
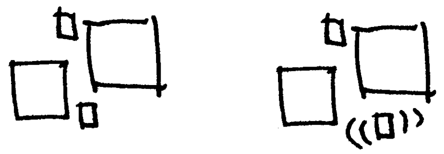

3.8.1. *two of Braitenberg's vehicles*

diagram of two machines is reproduced in figure 3.8.1. The machines are equipped with pairs of light sensors and motors. The sensors cause the motors to turn quickly and move the machines at high speed when they are far away from light, and more slowly as they come closer. Both machines will seek out and approach the light source. The first vehicle seems to 'like' the light source. It is content to approach the light and never leave its vicinity. However, because of the crossed connections in the right-hand machine, it will wander from the light source at the last moment, keeping an eye out for other stimuli.

In an organic visualization, movement is most often used to express a change in a set of relationships within a composition. The psychological attributions give an easily recognizable meaning to this occurrence. Movement is one of the most expressive dimensions available, by definition the dominating factor in temporal behavior.

Smaller, less determinate movement can also be used for a particular element to call attention to itself. Because of the nature of how the human visual system has evolved, movement can immediately attract attention and control focus.


3.8.2 *a wiggling element in a composition can quickly call attention to itself.*

## 3.9 Reproduction

Only life begets new life. New organisms are born through reproductive processes involving either a single parent that duplicates its genetic code, or two parents that produce a combined set of code, resulting in an organism with a unique hereditary blueprint—based on a mixture of traits from both parents.

Organic information systems can use reproduction as a model for how new data is added to the system, causing these new elements to 'inherit' characteristics from similar predecessors or siblings that already exist in the composition. In this manner, the new data can be assimilated into the system as related elements. They will have more valid starting points than if new data were added with 'blank slate' status. A basic example is shown in figure 3.9.1, where

3.9.1 *a new element being added to a composition, based on a mixture of traits from two ancestors.*

a new 'box' element is added. Its traits (position and size) are a mixture of two ancestors in the composition. Strictly speaking, this is not reproduction because new data is not created by the visualization. Rather, the rules borrow from the concept of sexual reproduction to create a new element that inherits its characteristics from two 'parents'.

A second method could use reproduction to selectively breed traits of the visualization itself so as to produce better visualizations. This concept is closely linked to the long-term adaptation described in Section 3.7. Such a study would require significantly tangential study with a set of goals from this thesis, and was not pursued.

# 4 Experiments in Organic Information Design

This chapter describes examples of the process of Organic Information Design. It introduces a set of computational structures used to build the most basic elements of an Organic Information Visualization and a software engineering method for implementing such a system. Finally, two example implementations are presented and discussed.

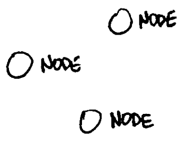
## 4.1 Structures

Four elements were used to build structure in the Organic Information Visualization experiments described in this chapter. Rules are written to be associated with each of these elements, based on the properties outlined in the previous chapter.

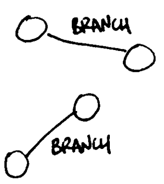
The model presented begins with *nodes* and *branches*, which are typical structures from the field of computer science. With these features, it is possible to accurately diagram a surprisingly large number of information structures. The structure could be hierarchical, connecting only mutually exclusive nodes. On the other hand, the branches could connect nodes in a haphazard manner, creating a complex network of linkages.

An alternative approach to the data structures could use vectors, dictionaries, and matrices. Connections could be expressed as intersections in a matrix. Ordering of elements could be represented using vectors. Dictionaries could relate elements together using a lookup method for their relations. This model was not pursued because usual the matrix would be very sparse, and the dictionary lookups computationally intensive.

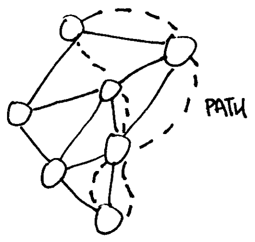
For the model chosen, rules are associated with each of the data structures. The rules take the form of the properties listed in the previous chapter. Most data structures have at least an appearance rule, determining how they are represented in the visualization.



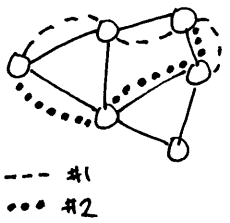
*Node*—the most basic element in the system. A single piece of data from the input is generally mapped to an individual node. Alternatively, creation of new nodes can be restricted to unique data, and a metabolism rule can increase their 'importance' as similar data is found in the input stream.



*Branch*—a connection between two nodes. The interaction rules for branches often act on the nodes they connect, for example exerting a force of attraction to bring them closer together. The visual rules for branches are important as they illustrate relationships between basic elements of data.



*Path*—a history of steps taken between the nodes. This is important because if the system consisted of only nodes and branches, it would only exhibit structure. Paths give a structure additional meaning. Dynamic information sources often describe sets of relations such as usage patterns. Paths provide a way to build time and frequency based relations between the nodes and branches in the system.



*Actor*–it is possible to have multiple paths, therefore it is necessary to differentiate between the entities that travel along the paths. An actor is an individual that is manipulating or traversing through the data in some way.

### 4.2 Values

Sets of static or changing values can be associated with each of the structural components. So far, two kinds have been formalized. But there are many other kinds of values that could also be included.

*Numeric*–a static numerical value. This is often used for keeping track of overall tallies and increments.

For example, a node might have a Numeric value associated with it called 'count'. A rule could be applied such that every time an equivalent piece of data appeared in the data source the count was incremented by one. This count might later be used by a movement rule that would cause a node to move to a different position in the composition based on its internal count, relative to other nearby nodes.

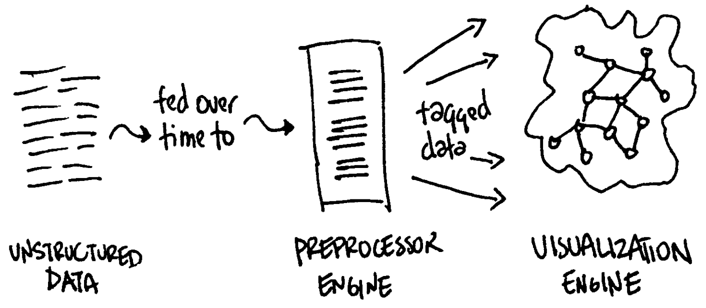
*Integrator*–a continuously changing value, perhaps the most common used in the system. Instead of fixed values, the Integrator is essential to the notion that these systems are in a state of flux. The systems exist within a continuous, rather than discrete, domain. Integrator values grow and decay rather than increment and decrement. Rules can take the following actions with an Integrator:

> *Set*–Explicitly set the current value of the Integrator. Normally, this is only used to set an initial value for the Integrator.
>
> *Impulse*–Adds a specified amount of force to the Integrator. Equivalent to incrementing a Numeric value, but executed in the continuous domain, i.e. the amount added attenuates over time.
>
> *Decay*–The opposite of an impulse. This is a decrease in the continuous domain. Often used to atrophy values over time.
>
> *Attract*–Apply a force to move the Integrator towards a particular value. Instead of setting the Integrator to a particular value, a target value is set for the Integrator that it will reach over time. Enables smooth transitions if the target is changing.
>
> *Repel*–The opposite of attract, moves the value of the Integrator away from a particular numeric value. If the value being avoided is greater, the Integrator is decreased further. If less, the Integrator is increased.
>
> *Update*–This is used internally to update the Integrator's current value on each time step, after calculating a new velocity based on the forces that have been applied to the Integrator by the rules that affect it.
>
> *Reset*–After each the current value is updated, the forces are cleared. New forces are added on each time step by re-applying the rules.

*Other values*–a node can have many data types associated with it, aside from the Numeric and Integrator values that are affected by rules. Typical data types include strings (lists of characters for words and sentences) or pointers that provide connections to other related structures.

## 4.3 Information Pipeline

The experiments are pipelined using the following set of steps.



4.3.1 *the Information Pipeline, a software engineering method to allow individual visualization components to be exchanged for new data sets or new representations*

*Unstructured Data*—begin with very generic, streaming data input. This could be raw data being read as it is added to a log file for a web site, or stream from a stock ticker.

*Preprocessor Engine*—this turns the unstructured data into something that falls into one of the four structural elements (node, branch, path, client). This is often a separate program which is implemented in an alternative programming language that is particularly good at dealing with text or binary data. In later experiments, the Perl programming language was used for this step.

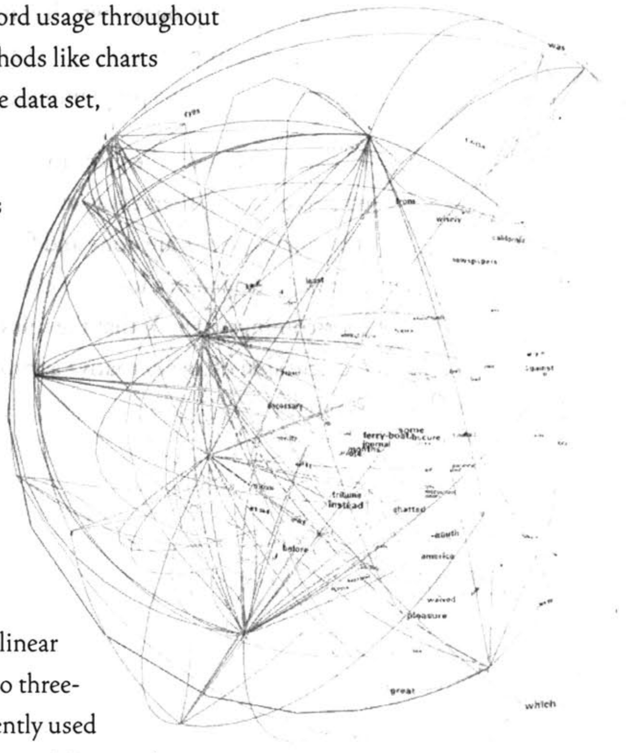
*Visualization Engine*—the piece that generates what's seen on the screen, and what the user is interacting with. This is the simulated organic system itself. It is where queries are done and filters happen. Often, for sake of speed, this part is developed in a lower level language like C/C++. This approach has its shortcomings, namely that the rules have to be built by a software developer. A discussion of an alternative approach can be found in section 5.7 of the next chapter.

## 4.4 Valence

How can word usage in a book be represented visually? "The Innocents Abroad" by Mark Twain is 200,000 words long, of which 15,000 words are unique. A bar chart containing this many elements would be nearly worthless. It would be too large to take in at a glance, or if shrunk to one's field of view, too small to understand. A focus+context technique like the *Table Lens* could be used, but due to enormous disparities in word usage (of the 15,000 words, fully half are used only once) less than 25% of the data would be worthwhile at all, with the interesting features not even appearing until the top 5%. This would leave a large amount of space with low importance and the lens focusing on the same area for the majority of the time.

Trying to find an effective solution becomes problematic, as each technique brings to light new issues. Even if these issues could be overcome by using some statistics and a modified bar chart, it's not clear that this would be a useful description of the data. There would be no concept of relationships between words. For instance, how can one tell what words appeared near one another in the text? How can changes in word usage throughout the book be expressed? Typical design methods like charts and graphs fail when applied to such a large data set, so new models are needed.

4.4.1 *Valence in use, reading "The Innocents Abroad" by Mark Twain*



Valence is a project that uses the properties of organic information visualization in an attempt to achieve a more telling representation. Every unique word in the book becomes a node. Branches are assigned to connect words that are found adjacent one another in the text. A set of rules is applied, based on the properties from the third chapter. These rules are detailed in table 4.4.2.

The resulting program reads the book in a linear fashion, dynamically placing each word into three-dimensional space. The words most frequently used make their way to the outer parts of the composition, so that they can be more easily seen. This leaves the less common words closer to the center. When two words are found adjacent to one another in the text, a line is drawn between them in the visualization. Each time these words are found adjacent to each other, the connecting line shortens, pulling the two words closer together in space.

This Organic Information Visualization continues to change over time as it responds to the data being fed to it. Instead of focusing on numeric specifics (i.e. the exact number of times a word appears), the piece provides a qualitative feel for perturbations in the data, in this case being the different types of words and language used throughout the book. This provides a qualitative slice into how the information is structured. On its own, the raw data might not be particularly useful. But when relationships between data points can be established, and these relationships are expressed through movement and structural changes in the on-screen visuals, a more useful perspective is established.

4.4.2 *table of rules used for the Valence visualization*

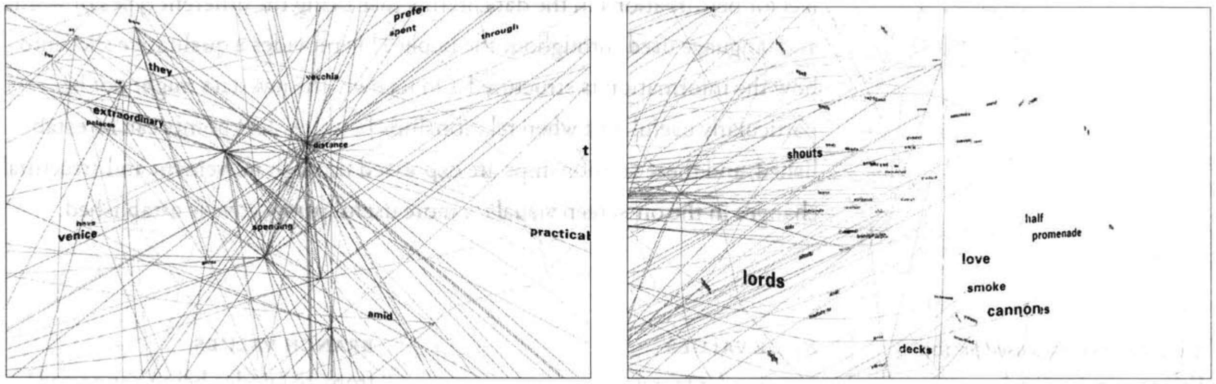
NODE VALUES
position (Integrator)
frequency (Numeric)
label (text of word, i.e. 'potato')

NODE RULES
*reproduction rule*—new words are added to random position in space as new nodes
*metabolism rule*—additional instances of same node increments 'frequency'
*appearance rule*—represent self using 'label' text
*movement rule*—compare 'frequency' to that of nearby nodes, if higher, add forces to 'position' to move self to outside of composition, and add negative forces to 'position' of neighbor, pushing it inwards

BRANCH VALUES
from, to (nodes being connected)

BRANCH RULES
*reproduction rule*—new branches are added to the composition
*metabolism rule*—additional instances of same branch applies forces to 'from' and 'to' nodes to bring them closer together
*appearance rule*—represent self with a line (algorithm for shape of line determined in the rule as well)

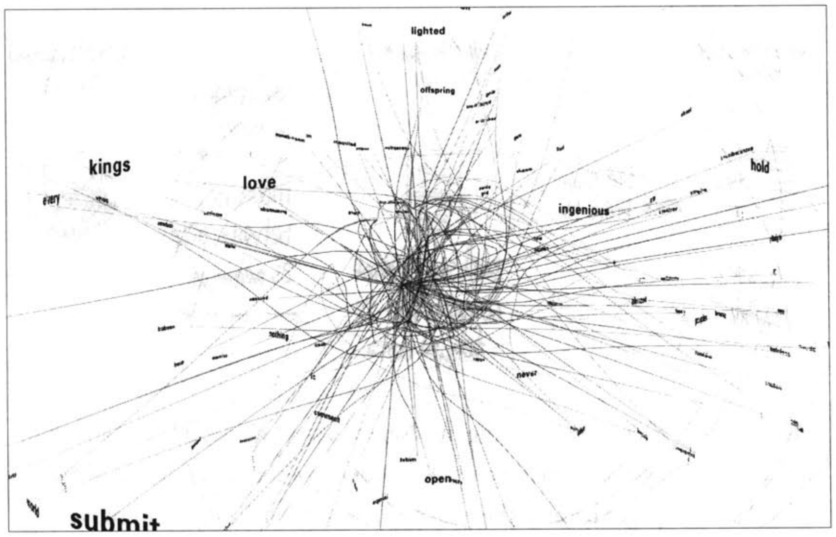
The individual movements of words coordinate into a symphony of small parts. For the viewer, focus shifts between the overall shape of the piece to anomalies that call attention to themselves through rapid movement or change in color. Related parts of the composition will group together in clusters, which is not explicitly stated in the movement rules, rather it is implied through the way the nodes interact with one another as they execute the interaction rules. Groups of relations begin to form which aid in the perception of the system.



4.4.3 *looking inside the space through zooming and rotation*

The user interaction in visualization allows for changes in viewpoint. The viewer may zoom into the center of the space to look around, or rotate the piece to view other locations. Interaction in this case could be improved by allowing for two things. This could be improved through implementation of direct manipulation, i.e. allowing the user to grab a particular piece and move it around, dragging related entities along with it. Using the selection to determine context, it would also be possible to provide additional information that was relevant to the selected node and its relations.

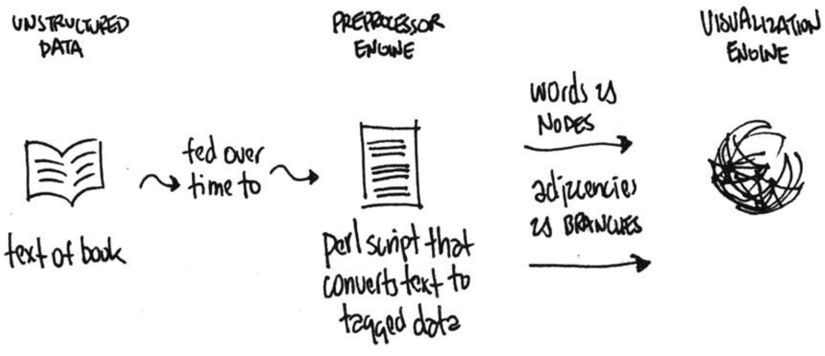
Changing the rules requires the software to be recompiled, which is a major drawback. Allowing the user to manipulate the rules being applied to individual nodes in the system (even while the system was running), would be a significant aid for the user to better understand how the rules construct the representation. For instance, if the user could change the movement rules to make the least frequently used words make their way to the outside of the representation. Figure 4.4.4 depicts the effects of this change, when applied to the same data set.



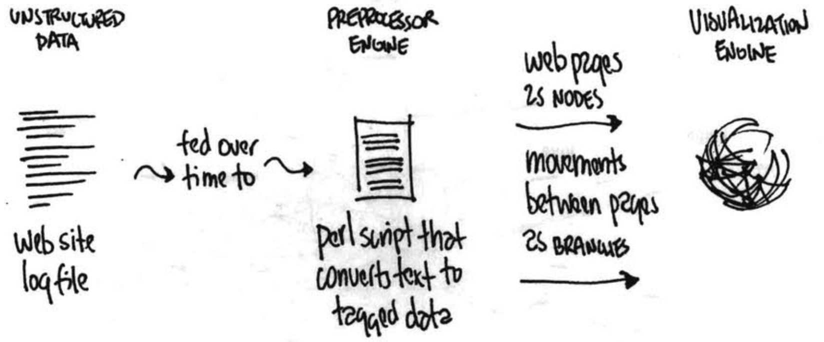
4.4.4 *after modifying the movement rules to push less frequent words to the outside of the composition*

The Visualization Engine was built using C++ and OpenGL. Using a language like Java instead of C++ would have reduced the time spent programming, and made it easier to implement features like modification of rules at runtime, but the speed of the resulting visualization would be inadequate on the target platform: a reasonably fast $2500 Intel PC with a $150 graphics accelerator.

Setting up this visualization using organic properties resulted in a versatile system that is capable of representing multiple types of data sets. The software was constructed with multiple parts, using the Pipeline process described in section 4.3. A modified figure 4.3.1 depicts the pipeline applied to this project in the diagram below. Because of this structure, it was possible to replace the Preprocessor Engine that feeds the book to the Visualization Engine, it is possible to use the Visualization Engine to represent other kinds of data.



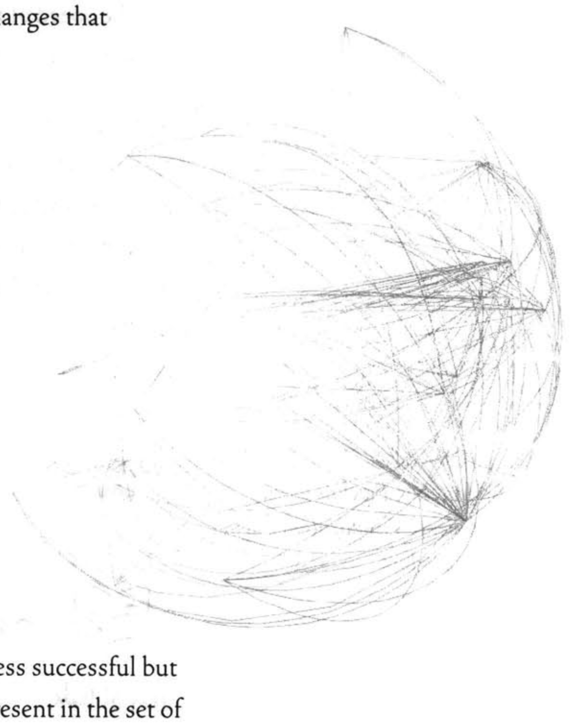
4.4.5 *Information Pipeline when using the Valence Visualization Engine with a text interface*



4.4.6 *Information Pipeline when feeding web site usage data to the Valence Visualization Engine*

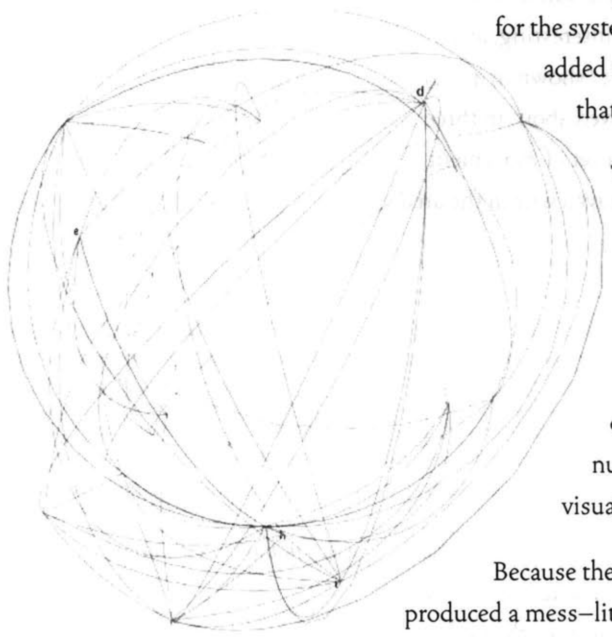
*Visualizing Web Traffic*—By replacing the Preprocessor Engine, this visualization has also been applied to tracking usage history on a web site. Instead of words, individual web pages became the nodes of the system. This is an improvement over typical web usage reports, which are typically comprised of bar charts and less useful statistics. Instead of the obvious information like "80,000 people visited the home page" and "only 10,000 people visited the 'projects' page", it builds a self-evolving map of how people were really using the site, regardless of how the site had been structured by its designer, or what changes that structure had undergone in the meantime.

4.4.7 *emergent clustering of web pages when Valence is applied to web site usage data*



As seen at the right hand side of Figure 4.4.7, areas cluster together as individual nodes begin to bunch up according to the rules used. In the web site example, these relate to web pages that are often visited in succession. This is the natural grouping that arises as people travel along the same paths repeatedly. This kind of emergent behavior is an important component of any distributed, reactive system. It is a very basic example of more complicated behavior that arises from the interactions of the various (simple) rules used to construct the system.

*Visualizing User Input*—another application of this system was less successful but makes an interesting point about some of the assumptions present in the set of rules used in the piece, and where they can break down. In an attempt to make an interactive version of this visualization, keyboard typing was used as input for the system. Individual letters (instead of words or web pages) were added to the space, with connections being made between letters that were typed in succession.



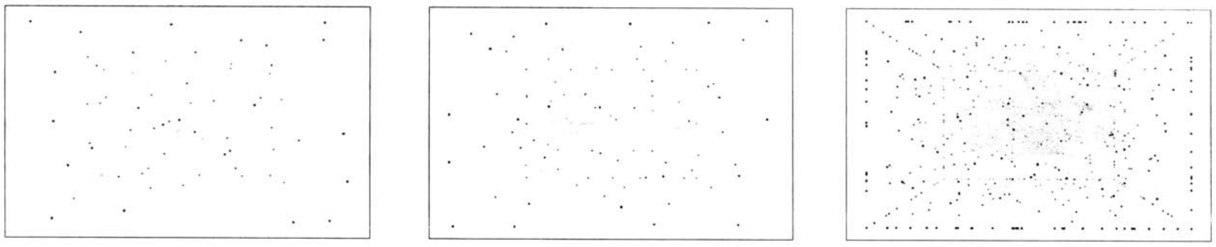
4.4.8 *degeneıar.e fiıg* ıer ofr.cng*fiıg* ı*a*ndıgct*ed* cnjor. *and* peıs*feq* nı*der*

This was an attempt to demonstrate Valence by leaving the production of data to the user. However, it turned out to be ineffective, because there was no motivation for the user to type anything but random letters. The result was usually a network of no more than 26 characters, all interconnected in haphazard ways. Because the rules were constructed to deal with large volumes of nodes, this low number contributed to the degenerate form created by the visualization.

Because the data was almost completely irregular, the visualization produced a mess—little clustering, no nodes of increased importance dancing around the perimeter. It could be argued that this was a case that a randomized input signal produced, true to its qualitative nature, a randomized garbage-like representation, but that is insufficient. It is likely that a valid data set could be nearly as irregular as typing on the keyboard. Especially because even seemingly 'random' typing on the keyboard will have some amount of correlation. i.e. higher likelihood of the use of characters appearing near the 'home' keys (s-d-f and j-k-l in particular).

To resolve these issues, rules governing habituation to the input signal should be employed. The visualization would then be capable of adjusting itself so that it recognized when the home keys were seeing slightly heavier use. An Integrator called 'habituation' could be used for each node, and a metabolization rule could impulse the Integrator each time the key associated with that node was pressed. In addition, a homeostasis rule would decay all habituation Integrator values on each time step. When the movement rules were applied (to determine what nodes should be pushed inwards and outwards) a node whose habituation value was high at that time would not be as forceful as a node whose habituation was lower. This would leave other, less habituated values (the nodes representing keys other than s-d-f and j-k-l) to become more prominent, and allow for better tracking of changes in the input stream

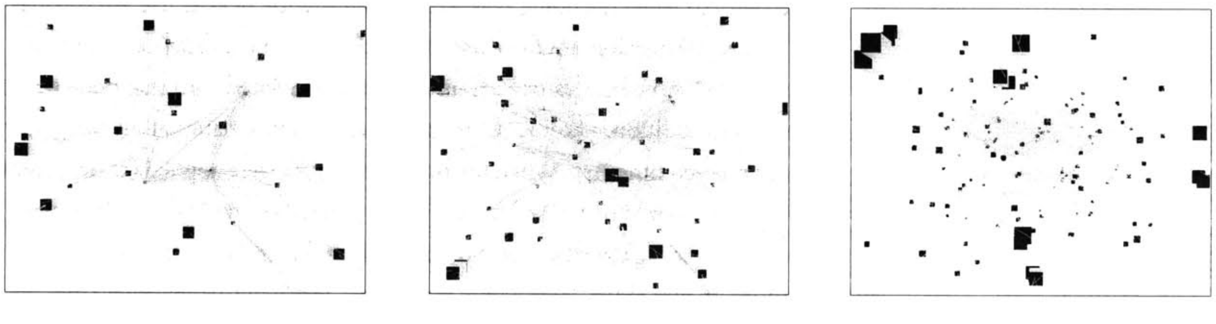
*Ajjeaiance*– a significant part of the design process was in pursuit of a set of visual rules for the system that would produce a useful, interesting, and readable form. The first appearance rules applied to Valence (shown in 4.4.9) were based on straight lines that connected nodes that moved about in three dimensional space. The space was bounded by a box, and a set of movement rules restricted the movements of the nodes so that they stayed within the area.



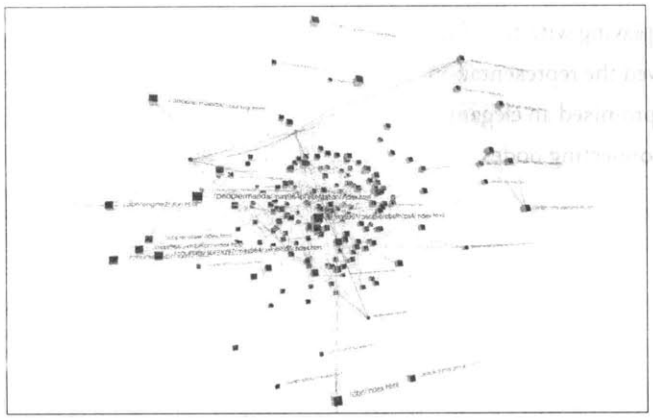
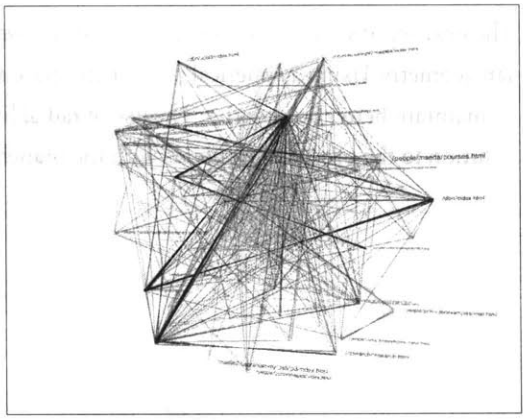
This representation produced an enormous visual jumble. As seen on the left, the system starts out messy but not completely inaccessible. It deteriorates as it moves in time towards the right-hand image, however. Forces of the nodes that repel one another cause several nodes to stick to the interior walls of the boundary box.

4.4.9 *initial representation*

Figure 4.4.10 shows three increasing time steps after applying an improved set of rules. In this example, the nodes attempt to group towards a center, with more 'important' nodes fighting their way to the front of the composition, changing their size just slightly as they did so. The change of scale enhances the feeling of perspective in this set of solutions, and introduces a nicer contrast between nodes of varying importance.



4.4.10 *with new appearance rules, introducing more visual contrast*

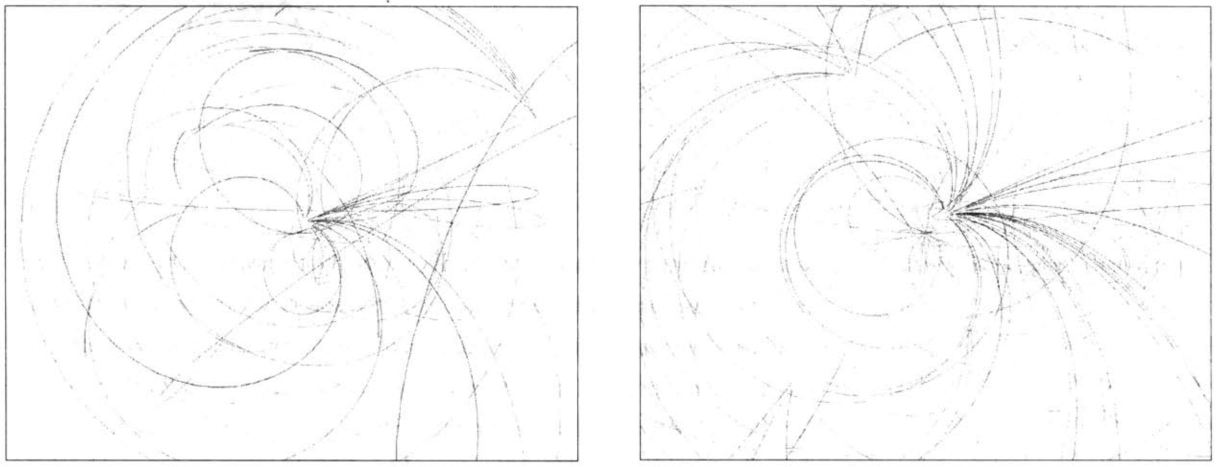
4.4.11 *initial attempt at using text*

Next, an attempt was made to introduce text into the system (figure 4.4.11 at the left) as well as creating a more constricted space, where the entire system could be viewable within the boundaries of the screen. Grouping of nodes around the midpoint in the left hand image is a positive point for the visualization, but it is flawed in its use of multiple lines to depict individual trips along the same branch. The system would quickly become dominated by these spindly fibers, which failed to communicate an amount of detail that was congruous with the amount of attention they called to themselves.

The right hand image was an iteration that backed away from the use of multiple lines by employing a single straight line that thickened based on use. The resulting figure was unattractive and even more confusing than its predecessor, thus other alternatives were sought.

At this point I was uncomfortable with the use of text, because it seemed to fall into a common trap that exists when using text. It is easy for a user to quickly latch onto a line of text, people seem to find text familiar (in a potentially unfamiliar, abstract space) and stimulating (when executed in 3D). It was easy, then, for observers to disregard the remaining visuals and be content with just text, because it was more concrete than anything else in the composition. Because I wanted to create forms that were evocative of the data, without simply relying on text to narrate the data, I temporarily discontinued the use of text in an attempt to return to the considerations of visual form.

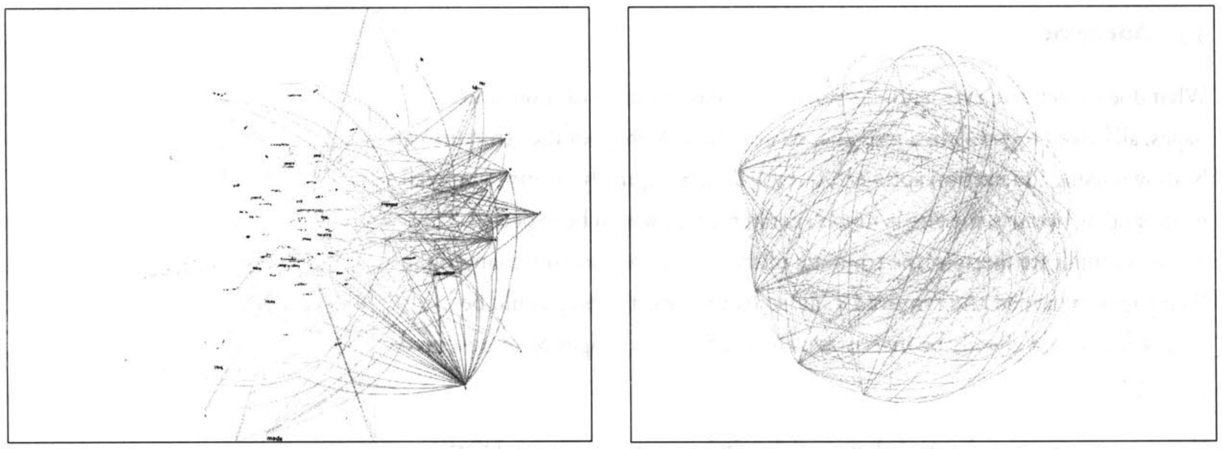
The next set of visualizations took a new direction by playing with non-Euclidian geometry. Using a spherical coordinate space allowed the representation to maintain better boundaries. The use of radial lines promised an elegant solution to the previous problems with the branches connecting nodes.



4.4.12 *using spherical coordinates*

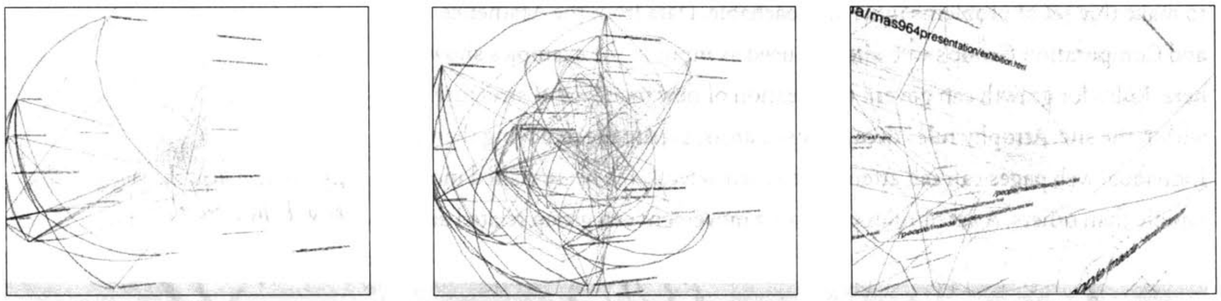
However, this solution had serious problems. While the resulting forms were quite graceful, they were also unreadable. The problem was that linear interpolation in a three dimensional spherical geometry system produces spirals, making it nearly impossible to track where branches led to, and creating much extraneous visual noise that distracted from the data.

The follow-up solution is shown in the right-hand image of figure 4.4.13. The method was to interpolate along the *great circle*, the widest arc that intersected the two points being connected. This was an improvement, but introduced new problems. The most important parts floated nicely on the outside, but the use of many concentric circles caused the visualization to become a visual ball of yarn, with internal data being almost completely obscured. The next step was to invert the great circles, producing the left-hand image of figure 4.4.13. This visual was very pointy, and gave much attention to the already obvious details of what parts were most important. However, it exposed the interior of the information space, which had been missing.



4.4.13 *too spikey, too round*

A mixture of the two appearance algorithms produced a single set of rules that shared the positive aspects of both representations, while shedding the negative. This combined format was the algorithm used in the final rendition of this project. The images of 4.4.14 show a visualization of web traffic in Valence at various time steps. The rightmost image is what the piece looks like after the user zooms inside the space using the mouse.



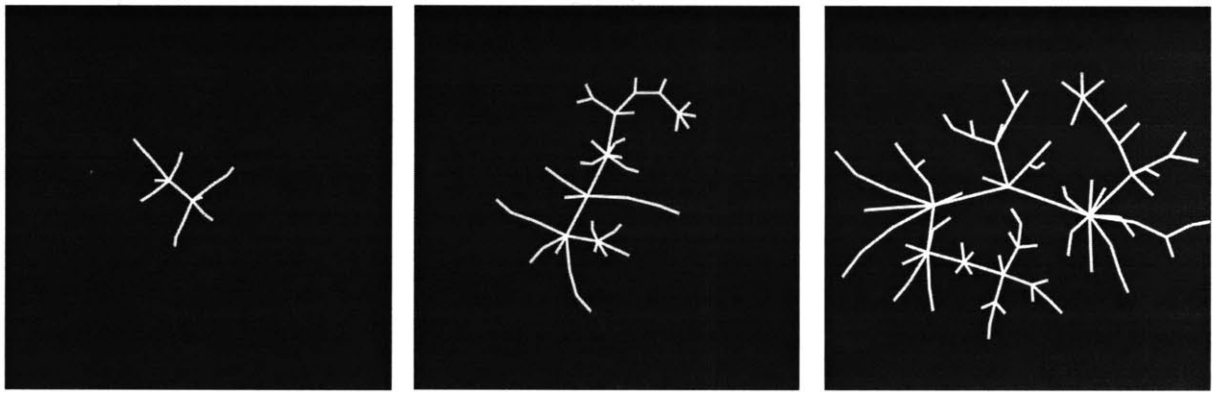
4.4.14 *just right*

## 4.5 Anemone

What does a web site's structure look like? A site is made up of thousands of pages, all linked together in a tree-shaped structure. A 'map' of the structure can be drawn using illustration software, but the diagram quickly becomes tangled in anomalies because the site is not as hierarchical as would be expected. To further complicate matters, the contents of the site are continually changing. Web pages are added and removed daily, so by the time the map could be finished, it would already be inaccurate. How can these changes be represented visually?

How can a connection be made between the site's usage statistics and that structure? How can the paths that visitors take through the site be expressed? A number next to each page could keep track of the total number of visitors, but because the traffic patterns are continually changing, it becomes clear that the totals aren't as useful as hoped. How can the day to day and month to month changes in these numbers be expressed? How can the movement between pages be represented, making it apparent that some areas are more closely related than others?

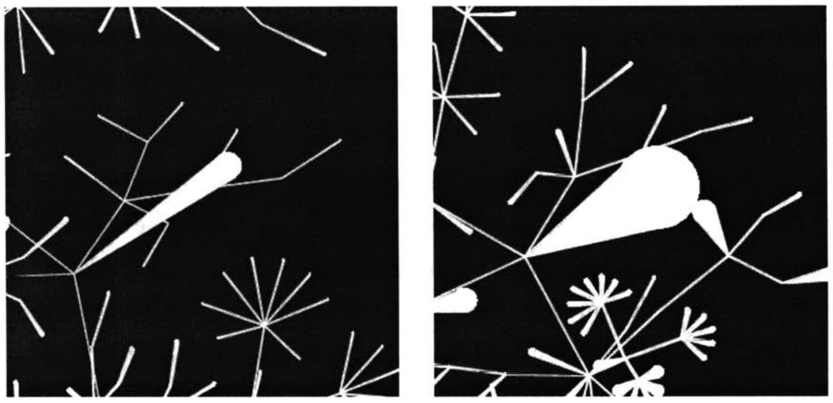
Anemone is a project that uses the process of Organic Information Design to make this set of problems more approachable. Data from the Aesthetics and Computation Group's web site was used as input in the examples shown here. Rules for growth can govern the creation of new branches of structure within the site. Atrophy rules decay unused areas, eventually removing them. Individual web pages can call attention to themselves as they are visited more rapidly than others. A set of rules governing movement can group related areas.


4.5.1 *several steps showing growth of the web site's structure*

Individual branches grow based on input from the data. As the Preprocessor Engine reads the usage log, a reproduction rule causes branches to grow whenever parts of the site are visited for the first time. This avoids the problem of having to keep track of what pages are added to or removed from the site. Using the usage data to create an implicit model of structure is a common theme in Organic Information Design.

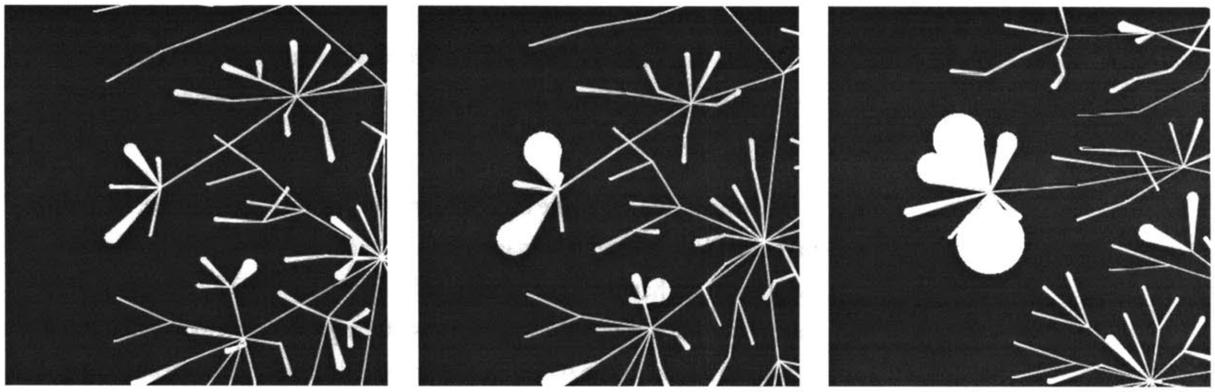
To balance growth is the notion of 'atrophy'. Branches associated with areas of the site that have not been visited will slowly wither away, causing them to visually thin out. Eventually the branches die, and are removed from the system.

A movement rule keeps the individual nodes within a set distance from their parent node. A second rule maintains a distance between nodes and their neighbors, so that branches overlap as little as possible. The composition is brought to life through the interactions of the growth and movement rules. The figure moves about the screen in a hyper, erratic fashion as it creates the initial parts of the visualization. After some time, this growth reaches an equilibrium and the pseudo-organism 'settles'.


4.5.2 *nodes attracting attention and declaring their importance through changes in thickness*

Nodes at the tip of each branch represent a web pages. As seen in figure 4.5.2, each time a user visits a page, its node in the visualization becomes slightly thicker. Nodes for pages that are visited often become very thick relative to other nodes. This can happen rapidly, as nodes attempt to 'call attention' to themselves.

If several users visit a particular section of the site, that group of nodes will collectively thicken, drawing attention to the group. An interesting phenomenon can be watched as it propagates back through the site structure. For example, many people may visit the site from a link referring to it in the Yahoo directory (www.yahoo.com). This traffic can be watched while it propagates outward from the linked page.

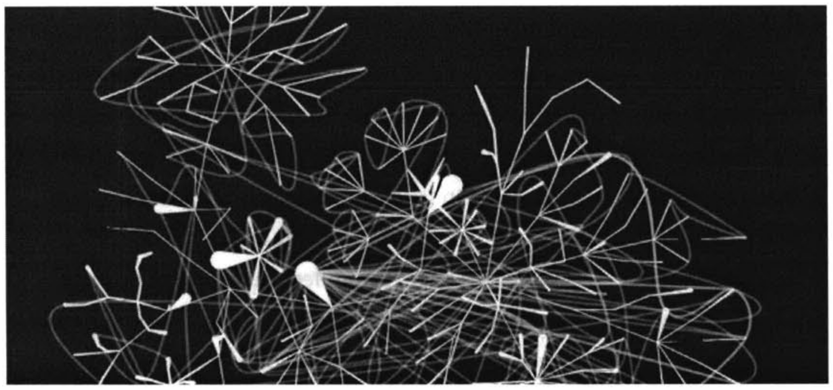

4.5.3 *progress of thickening in a group of related nodes*

In addition, the appearance of the nodes self-regulate. If a particular page is frequently and consistently visited, its associated node in the visualization will thicken, but only up to a certain threshold. Not much additional information is learned therefore there is no need to allow the node to grow to enormous proportions. This homeostasis rule causes the node to settle to a certain size until changes in its use occur.

The Anemone experiment looks at how structural information can be juxtaposed with less structured usage patterns. Figure 4.5.4 is an image of Anemone with two layers. The top layer is the directory structure of the site (depicted with branches), determining a hierarchy for where individual web pages are located. The layer beneath represents the paths taken by users as they visit the site.

The paths can follow the directory structure, which is closely related to the link structure, as in the case of the web site being visualized here. The paths can also have disparate jumps to various areas of the site. Looking at the two layers together show interesting trends in the paths that large quantities of users take through the site. For instance, a large number of paths can be seen that connect the home page of Professor Maeda into other areas of the site. This is because

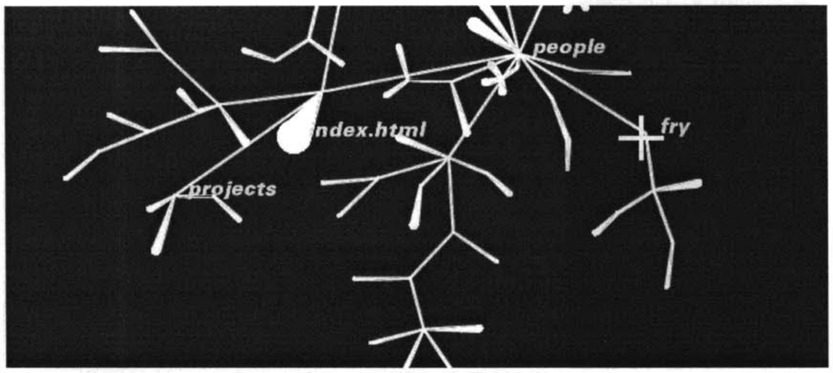
4.5.4 *the two layers of Anemone, juxtaposing structured hierarchy with nonlinear usage*



many external links point to his page. In addition, a large number of visitors arrive from search engines having queried for his name.

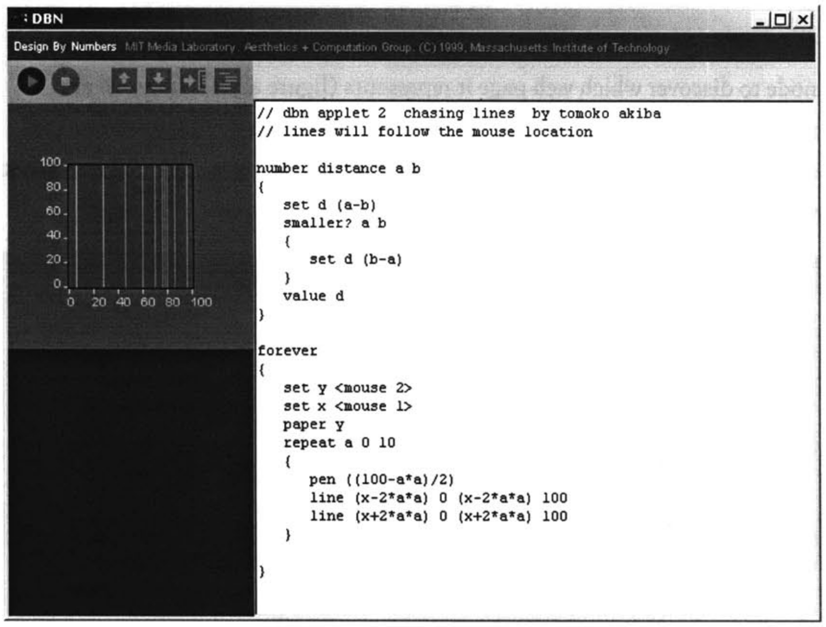
User interaction with this visualization is important. The viewer can click a node to discover which web page it represents (figure 4.5.5). They can also move nodes around as a way to peek inside the data set and take a closer look at what's happening. Nodes can be dragged about the screen, pinned down, and watched in relation to other parts of the structure.

4.5.5 *user interaction, dragging nodes with the mouse*



The graphics in this project are two dimensional, unlike the previous project. I wanted to experiment with 2D because 3D doesn't necessarily add an additional dimension, as some would believe. The human visual system doesn't let us truly 'see through' things, so if the output device is a flat surface, the viewer really only perceives 2D and occlusion. More advanced use of occlusion in this work is still being investigated.

*Visualizing computation*–Anemone has also been applied to the visualization of computational processes. Figure 4.5.6 is the coding/editing environment for Design By Numbers (DBN), a simple programming language developed by John Maeda to teach non-programmers (artists and designers in particular) about computation [Maeda, 99]. On the right-hand side is the editing area where a program is entered; on the left is the imaging area, controlled by a running DBN program. About a year ago, I completed a rewrite of the inner-workings of DBN, which deal with how text written by the user is first parsed and then executed by an interpreter engine. Because this process is so opaque and rarely understood in common use, even by experienced programmers, I decided to develop a visualization of how this process takes place.



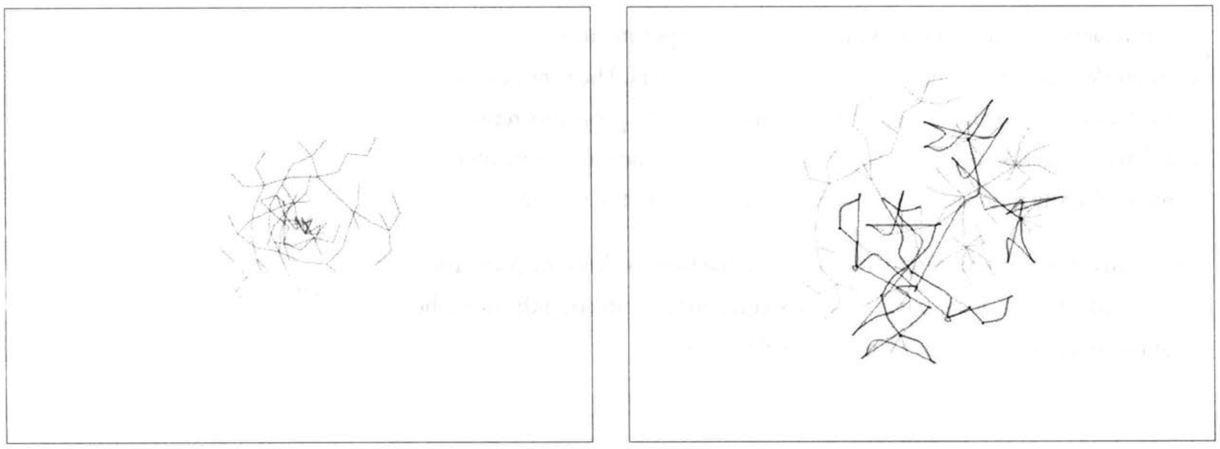
DBN
Design By Numbers MIT Media Laboratory Aesthetics + Computation Group (C) 1999, Massachusetts Institute of Technology
100
80
60
40
20
0
0 20 40 60 80 100

```
// dbn applet 2  chasing lines  by tomoko akiba
// lines will follow the mouse location

number distance a b
{
   set d (a-b)
   smaller? a b
   {
      set d (b-a)
   }
   value d
}

forever
{
   set y <mouse 2>
   set x <mouse 1>
   paper y
   repeat a 0 10
   {
      pen ((100-a*a)/2)
      line (x-2*a*a) 0 (x-2*a*a) 100
      line (x+2*a*a) 0 (x+2*a*a) 100
   }

}
```

4.5.6 *example of a simple mouse-based program running in the DBN environment*

The concept of a parse tree is central to any programming language. A parser converts the program typed by the user into a tree-shaped structure. To run the program represented by the tree, the tree is traversed visiting the branches in a well-defined order, starting with the root of the tree.

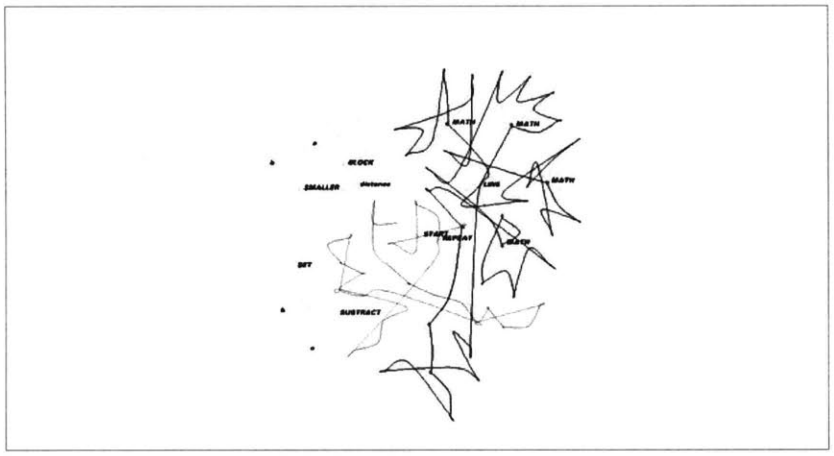
The top layer of branching lines is the parse tree of the program typed by the user. After parsing this program, it is output to Anemone using a simple text-based protocol. The branches have simple layout 'intelligence' and are aware of their neighbors. This causes them to avoid overlapping one another, and



4.5.7 *growth of the parse tree*

produces a concise layout. This process is about halfway finished at the left of figure 4.5.7.

The right-hand side of figure 4.5.7 shows the parse tree once it has neared an equilibrium state for its layout. A darker path is beginning to appear, illustrating the flow of the execution engine (interpreter) as it walks through the parse tree of the program code. Figure 4.5.8 shows a more advanced stage of this process, with an area of the program becoming very bright its use increases. This piece of code is inside the main loop of the simple mouse-based program in figure 4.5.6. The visual provides a impression of how a program is run. In addition, it provides insight into the process of how software is run by a machine.



4.5.8 *the visualization after several execution cycles, the path of the interpreter has become very prevalent*

User interaction is the same as with the previous experiment that used the Anemone Visualization Engine with web site usage data. The user can click a particular node in the tree to see what it relates to in the program's representation. This interaction is useful in the sense of making the piece more literal, and therefore slightly more accessible for users just learning the system.

A more advanced system might allow user interaction to affect the program while it runs. By removing parts of the visualization, or directing the flow, the program could be modified in a more visual manner.

# 5 Analysis

This section describes the successes and shortcomings of this thesis work, a taxonomy for reading and observation, guidelines for implementation, and several directions for future work, including a discussion of improvements to a design environment in which Organic Information Visualizations (OIVs) can be built.

## 5.1 Successes

*Qualitative representations*–Organic Information Visualizations are effective qualitative representations. Developing immediate impressions from the first instant a user looks at a piece, or within a short time span, is extremely useful. The systems become useful tools for tracking instantaneous trends in dynamic information sources. Visualizations of such data sources can exist in the background, as an ambient component of one's environment, and still be effective.

*Exploratory visualization*–The visualizations excel when applied to exploratory data visualization, as an interactive way to learn about a data set. The example in section 4.4 of trying to understand word usage in a large text explains this in further detail.

*Demonstrative systems*–Rather than giving specifics about the content that they represent, the visualization demonstrates it visually. The viewer is engaged in parsing of the display, mentally comparing the current state with previous states. This process is implicit, however, in the human perceptual system. The mind continually contrasts what is currently being seen against related, stored images. The strength is the speed and accuracy with which this can be done, especially without it being an explicit mental task. A weakness comes from only being able to be certain that these comparisons can apply to broad trends. Unless the viewer is more actively engaged, this ability won't get much further.

## 5.2 Shortcomings

It seems that OIVs are potentially less useful for quantitative analysis. While it is possible to support a mixing of quantitative specifics into the representation (like the example shown in section 1.2), this has not been pursued to much depth.

Addressing highly noisy or irregular data is another issue that could use further exploration and testing. The keyboard input example in section 4.4 is a simple explanation of a broader difficulty. Finding appropriate and repeatable ways to make OIVs self-regulate to bring the useful features out of noisy data is potentially a very difficult problem.

Information lacking structure is a related area. Consider a stream of sampled sound data. It would require many components to construct a visualization that would do something useful with this data as input. This is entirely possible under the model of OIV, but tests in this area are far from substantive.

The use of OIVs for prediction has not yet been explored. Although it is possible to construct a visualization out of neuron elements which would be capable of sophisticated prediction, it has not been fully examined in the context of information visualization.

## 5.3 Reading and Observation

This research points to multiple ways to read an abstract visualization and their relationship to current visual information processing theories.

1. *Innate understanding*–rely on one's own graphical sensibilities, using Bertin's model of an innate model of graphics based on charts, maps, etc. [Bertin, 83] Perhaps only a weak link to this work, as it pursues alternative representations that are not necessarily intuitive.

2. *Comparing deviance*–first determine what the 'normal' state looks like, and then watch how the changing data signal causes the visualization to deviate from that state. This is probably the closest model for the current set of experiments. The normal state is continually updated as the visualization evolves in time.

3. *Comparing multiples*—an interesting area of research would be comparing several versions of the same system, each with slightly different rules determining its organization and representation. By observing the relative differences between the iterations, deeper understanding can be gleaned. There are two ways this can be done. First, by looking at streams of data, and running them through the systems based on the same sets of rules. Second, by looking at the same data set and changing the rules used to display it. This method is related to the notion of *small multiples*, discussed in [Tufte, 90].

4. *Information dissection*—by doing queries within the rules to see what kinds of things are happening, the viewer can engage in a kind of detective work. Sometimes reading a data will be a difficult process that requires the viewer to take an active role in learning about the data. Information design is concerned with making data accessible, not dumbing it down or limiting it to what is instantly digestible.

## 5.4 Guidelines for Rules

Having applied the Organic Information Design process to a set of experiments, the following guidelines become apparent.

*Structure*—it is essential to construct these systems in ways that will cause the individual parts to aggregate together. The experiments presented used simple clustering and grouping, but did not use employ additional rules to govern the grouping of those first-level clusters.

*Appearance*—these are complicated systems, and it is easy (and therefore tempting) to make them very beautiful in their complications.. This is a weak exercise, however, because one can take the simplest thing and make it needlessly complicated but beautiful. Ostentatious or wildly complex things can be interesting, if they're done in the appropriate contexts and executed in a creative manner. But in general, it is more important to simplify for the sake of the user.

*Movement*—it is important to avoid extraneous movement in the composition that is not related to the task at hand. Movement eye-catching and quickly distracts the user. A visualization can entertain and hypnotize with movement, but the viewer loses touch with the content of the representation.

## 5.6 Randomness

There is a danger in employing any sort of randomness when representing a system, particularly in the kind of representations discussed in this thesis. Layered interrelationships can cause a small bit of randomness to propagate and amplify throughout the system. Initial experiments used a small amount of randomness to provide minor perturbations where needed. For instance, if two nodes were too close together or occupying the same space, a small amount of force would be applied to make them repel from one another. Unfortunately, this introduced an artificial kind of novelty that was not motivated in any way by characteristics found in the data set.

Running a visualization that contains randomness twice with the same set of data produces a slightly different result. As horrifying as this might sound to a stastician or scientist, it is not as inherently evil as it might seem. The very basic aspects of the qualitative impression (being the primary view pursued by the visualization) are preserved. It is not sufficient to be content with maintaining just this minimal part of the qualitative impression. Randomness is a significant issue that needs to be addressed.

In later experiments, my goal was to remove randomness by using a (sometimes arbitrary) feature of the data to flip the coin of chance. This has the advantage of causing a visualization of the same data set (using the same set of rules and parameters) to produce equivalent and repeatable results. This is an improvement over the previous outcome, but is still imperfect. There is a kind of randomness in the decisions that the designer makes to remap some semi-arbitrary feature to provide the 'answer' previously determined by a random number. This errs on the side of being overly deterministic and artificial with respect to the pieces themselves or the method of their representation.

More work needs to be done in determining a more ideal solution for this issue. OIVs maintaining equivalent qualitative character, in spite of small amounts of randomness, seems the correct direction. But determining where randomness is permissible, and to what degree, is a difficult task.

## 5.7 Platform and Programming Language

All of the experiments described in this document were built using C/C++ and OpenGL. This is not in any way the preferred language, as programming and tweaking a project built in C/C++ requires an edit-compile-test-terminate process that can be very tedious. It is far from effective as an environment to do any kind of sketching, tweaking, and iterating: all necessary components of a design process.

On the other hand, C/C++ was the simplest model for me to use with maximum performance, as the implementation of systems involving many small interacting components are extremely performance-sensitive. The data being modeled is voluminous, and the behaviors applied to individual nodes are computationally intensive. A software developer could spend a great deal of time simply tweaking for speed and performance gains, but it would be time wasted, with regards to the current stage of this research. There are more pressing issues to be addressed.

Without regard for the performance issues, an improved environment would use an embedded interpreter, providing a means for modifying and updating the rules in real time and viewing the outcomes based on the changes. Manipulation by simple programming then becomes part of the observation process. It's a step away from how one toys with numbers in a spreadsheet to 'do some figures' or using a piece of software like MatLab to take a look at a data set or signal.

An additional improvement would be the replacement of C/C++ with a high-level language in which data types have an inherent knowledge of their neighbors and their relationship to them. Once this is in place, it's simply a matter of building ways to manipulate those relationships, and combining that with a visual representation that expresses those relationships. Data structures like resizable lists, trees, and various flavors of graphs are well understood in computer science. They should be simple to use as data types but are often more difficult to deal with, more tedious than they should be, or simply more tedious than would be expected.

## 5.8 Additional Future Research

In addition to an improved development environment and addressing the shortcomings of the current pieces, a number of content areas could have potentially interesting representations if built using the Organic Information Design process.

GENOMICS–The human genome will be completely mapped within six months. A model for representing the 600 million elements will be needed, but more importantly, a method for relating these elements to gene expression.

ECONOMICS–An economy is an extremely complex system involving many parts. Experiments that examined various aspects of such economies could prove interesting. For example, the interactions of corporations and conglomerates across multiple industries, and how these affect one another and other parts of the economy.

AUDIO PROCESSING–Understanding less structured data, or using organic properties to build structure out of a generic signal could provide valuable insights in how to improve the process of Organic Information Design.

COMPLEX ADAPTIVE SYSTEMS–How do complex systems of rush hour traffic resolve themselves? Simulations with large numbers of actors and variables such as these point to the application of Organic Information Design to a large body of research.

GAME THEORY–Visualization of the rules involved in game theory and their relationship to how a game is actually played could help bridge understanding in this area.

SOFTWARE AND COMPUTER ARCHITECTURES–During the course of this work, several small attempts were made at developing representations of various computational processes. This work could be expanded further, with experiments that attempt to visualize more advanced software projects, or an entire operating system.

# 6 References

Ahlberg, Christopher. Shneidermann, Ben. *Visual Information Seeking: Tight coupling of dynamic query filters with starfield displays*. ACM CHI '94 Conference Proceedings. (Boston, MA, April 24-28, 1994) 313-337.

Anscombe, F. J. *Graphs in Statistical Analysis*. American Statistician, 27. February 1973, 17-21. Cited from [Tufte, 92] pp. 14-15.

Bertin, Jacques. *Semiology of Graphics: Diagrams, Networks, Maps*. Madison, Wis.: University of Wisconsin Press, 1983.

Braitenberg, Valentino. *Vehicles: Experiments in Synthetic Psychology*. Cambridge, Massachusetts: MIT Press, 1984.

Card, Stuart K. Mackinlay, Jock D. Shneidermann, Ben. *Readings in Information Visualization: Using Vision to Think*. San Francisco: Morgan Kauffman, 1999.

Coveney, Peter. Highfield, Roger. *Frontiers of Complexity*. New York: Fawcett Columbine, 1995.

Flake, Gary William. *The Computational Beauty of Nature*. Cambridge, Massachusetts: MIT Press, 1998.

Frisch, U. Hasslacher, B. Pomeau, Y. *Physics Review Letters*. 56 (1986) 1505.

Gershenfeld, Neil A. *The Nature of Mathematical Modeling.* New York : Cambridge University Press, 1999.

Hardy, J. de Pazzis, O. Pomeau, Y. *Molecular Dynamics of a Classical Lattice Gas: Transport Properties and Time Correlation Functions*. Phys. Rev. A, 13 (1976), 1949-61.

Holland, John H. *Hidden Order: How Adaptation Builds Complexity*. Reading, Massachusetts: Addison-Wesley, 1995.

d'Humières, Dominique and Lallemand, Pierre. *Complex Systems.* 1 (1987) 599-632.

Kilian, Axel. *Defining Digital Space Through A Visual Language*. M.S. Thesis. Massachusetts Institute of Technology, Department of Architecture. 2000.

Lamping, John. Rao, Ramana. *Laying out and visualizing large trees using a hyperbolic space*. Proceedings of UIST '94, pp 13–14, 1994.

Logo Foundation. *What is Logo?* http://el.www.media.mit.edu/logo-foundation/Logo/Logo.html

Maeda, John. *Design By Numbers*. Cambridge, Massachusetts: MIT Press, 1999.

Rao, Ramana. Card, Stuart K. *The table lens: merging graphical and symbolic representations in an interactive focus+context visualization for tabular information*. ACM CHI '94 Conference Proceedings. (Boston, MA, April 24-28, 1994) 222.

Resnick, Mitchel. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Cambridge, Massachusetts: MIT Press, 1994.

Reynolds, Craig. *Flocks, Herds, and Schools: A Distributed Behavioral Model*. Computer Graphics 21(4). July 1987, pp. 25-34.

Shneidermann, Ben. *Dynamic Queries for Visual Information Seeking*. IEEE Software, 11(6) 1994. pp. 70-77.

Sims, Karl. *Evolving Virtual Creatures*. Computer Graphics, Annual Conference Series, (SIGGRAPH '94 Proceedings), July 1994, pp.15-22.

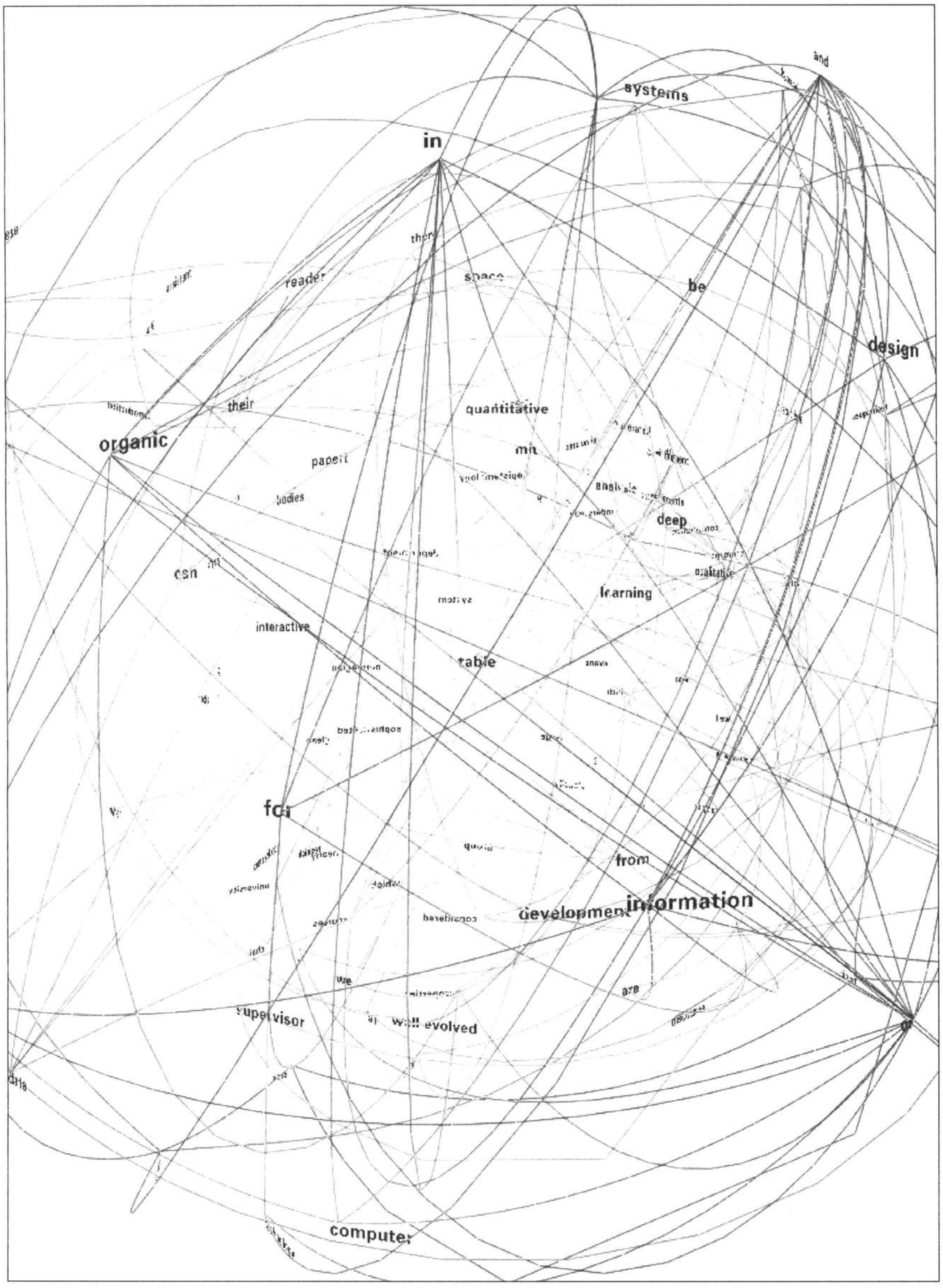
Strausfeld, Lisa. *Embodying virtual space to enhance the understanding of information*. M.S. Thesis. Massachusetts Institute of Technology, Program in Media Arts & Sciences, 1995.

Tufte, Edward R. *Envisioning Information*. Cheshire, Conn.: Graphics Press, 1990.

Tufte, Edward R. *The Visual Display of Quantitative Information*. Cheshire, Conn.: Graphics Press, 1992.

Ville, Claude A. et al. *Biology*. New York: Harcourt Brace Jovanich, 1989.

Wurman, Richard Saul. *Understanding USA*. New York: Ted Conference Inc, 1999.


and
systems
in
be
design
reader
space
their
quantitative
organic
mit
paper
bodies
analysis
deep
can
learning
interactive
table
for
from
information
development
are
supervisor
well evolved
computer
data
of