RC 16776 (#74079) 4/16/91
Computer Science 31 pages

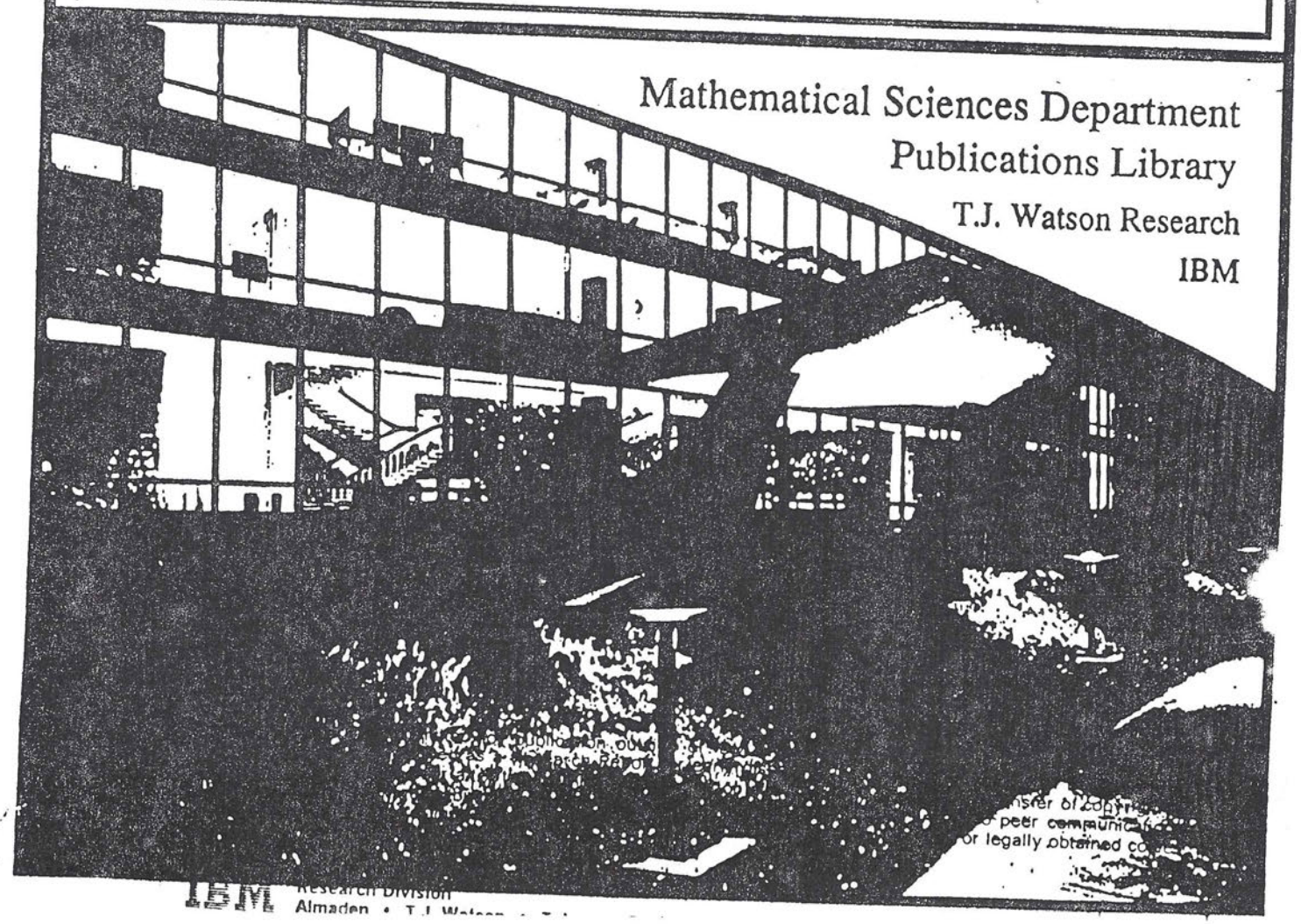
X2

Research Report

Graphics as Visual Language

Ted Selker and Art Appel

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598



Graphics as Visual Language

Ted Selker Art Appel

IBM Thomas J. Watson Research Center
Yorktown Heights, NY

Abstract

Visual language is the systematic use of visual presentation (graphic) techniques to convey meaning. Current computer user interfaces make extensive use of graphic techniques in many areas. This chapter defines structural elements of visual language as a framework for describing computer graphics.

Graphics presentations can be described as having the following visual elements:

graphical alphabet the set of visual primitives used in a visual language
graphical syntax the composition of primitives which form visual statements
interaction language the set of user to system primitives
structure rules that combine sub-languages to form a language

The classification of the visual elements can be viewed as a linguistic description of visual language.

The chapter explains how computers can create graphical presentations and promotes the utility of graphical presentation of data. A concluding section is devoted to the use of graphics in statistics.

Table of Contents

List of Illustrations

1. Introduction

Computer users cannot avoid being influenced by icons, menus, and computer graphics.

A fruitful way to begin to understand appropriateness of graphical techniques is by enumerating them in a structural classification. We don't know all the answers as to what graphical representations are best; how does a specific graphical user interface reveal the underlying semantics of the program? How does the use of graphics in one interface compare to other interfaces? How does a specific use of graphics affect the understandability of data? What other kinds of data should be presented with such an interface? Questions such as these can illuminate tradeoffs in a specific use of visual language in a user interface [51].

The field of graphic design has generated a set of rules for creating depictive text and graphics on paper. Several books have been written as guides for designing graphic presentations [29, 60]. We should draw from such work to codify and extend these rules to interactive computer systems. People in fields ranging from semiotics to sociology have begun the process by cataloging techniques and describing the value and power of visual languages [4, 8, 15, 30, 40, 54] Mackinlay86'.. This chapter explains concepts of visual elements in interactive computing.

2. Background

Definitions of visual language have taken many forms. Some workers have taken a formal mathematical approach [8, 23]. Mackinlay, for example, used first order logic to define a formalism for working with presentation graphics [36]. The notable works of Card and Reisner use task action grammars to describe interactive systems [12, 44]. Such formal definitions of visual grammars do not necessarily afford a working definition or feeling for visual language.

Chang wrote a tutorial on visual language which used an operational approach to classifying visual languages, describing them relative to what they do. The tutorial draws on a broad range of sources [16]. Chang segments visual languages into:

- languages that support visual interaction
- visual programming languages
- visual information processing languages
- iconic visual information processing languages

The tutorial describes a number of systems that fit into each of these categories.

Selker, et al, built a framework for comparing systems with visual interfaces [51]. The framework guides a person to consider the following aspects of an application:

- the visual elements used in an interface
- their domain of application
- the relationship between the interface and system functionality
- the expressive power of the interface
- the interaction style and mode
- the visual interface's relationship to other visual interfaces

The framework helps describe interactive systems with visual interfaces.

The visual elements described in Selker's framework serve as the basis of a visual language and the topic of this chapter. Bertin made a vocabulary for the elements of Diagram, Network and Map visual languages:

Marks:	points, lines, areas.
Positional:	1D, 2D, 3D.
Temporal:	Animation.
Retinal:	color, shape, size, value, texture, orientation.

This chapter expands Bertin's Graphical Vocabulary for Diagrams, Networks and Maps. Bertin's retinal properties are refined below to correspond to experimental psycho-physical research results. His marks and positional properties will be expanded as well.

This chapter refines the definition [49] that the visual elements of a language are the graphical alphabet, interaction alphabet, graphical syntax, interaction syntax and structure of that language. These elements define the language and must be analyzed as part of its design. [51].

3. Graphical Alphabet

Visual language is the systematic use of visual presentation techniques to convey meaning. A visual alphabet is made up of a set of spatial and temporal "characters" used in a visual language; iconic systems and visual objects have similar definitions [16, 39]. The following visual alphabet taxonomy draws on previous studies [9, 27, 35]. Visual alphabets are made up of images, icons and symbols.

- *Images* are derived from photos, digitization or projections. Chang and Lodding describe representational imagery as images derived from photographs, degraded photographs or drawings [16, 35]. Fred Lakin's "visual parser" allows shapes to be interpreted as meaningful language elements in a visual grammar [34].
- *Icons* are symbols with physical world referents [9, 16, 27, 30, 33], stylized representations of objects or processes. For example, the fork and knife icon connotes a place to eat.
- *Symbols* are drawings of arbitrary design [35]. The character "a" is an example of a symbol. These are labels with no ***a priori*** referent.

Images, icons, and symbols have *Surface Characteristics*, which include the perceptually separable and salient dimensions of color: hue, saturation, value and texture [38].

- *Hue, saturation and value* define dimensions for color. These are often used to convey information; stop and caution are typically depicted by red and yellow, respectively.
- *Texture* can also convey information. It is often used to differentiate obscured items in a menu or windows on a screen. Textures on computer interfaces, like textures on wall paper, are composed of images, icons or symbols (Figure 37). Halftoning, for example, is a process that uses dichromatic texture to recreate gray-scale images. A grid of dots gives the illusion of a gray-scale image (Figure 1). Light areas are depicted with small dots and dark areas are depicted with large dots.

Figure 1. Halftoning

We call the process of applying surface characteristics to alphabetic elements rendering. *Rendering* is the technology and approach used to present a visual language. Techniques for rendering include drawing with with vectors or polygons. Layout languages such as Press [55], PostScript [2] and Movie-BYU [1] are languages for rendering. The techniques and practices of rendering are what we most often think about as Computer Graphics. The importance of these techniques motivate us to devote the following section to its description.

A Guide to Graphical Rendering

Contemporary computing systems, even the smallest, can draw pictures that are sufficient for almost all information display needs. An overview of some various types and styles of computational renderings will be presented as a guide to the possibilities.

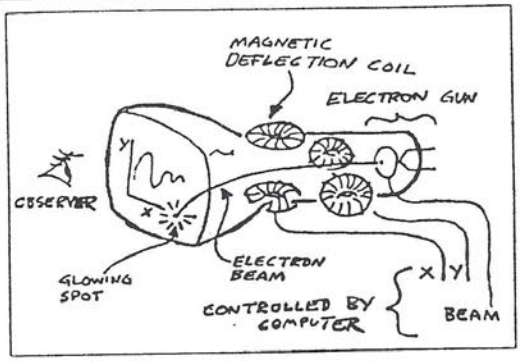


Figure 2. Computer Graphic screen. The screen is bright where electron beams strike fluorescent coatings inside the display tube. The electron beam scan is deflected by magnetic fields, intensity of the beam is set by a control voltage. Computer displays are standardized so that computer users do not have to worry about display devices but only x and y coordinates and the pixel intensity.

Gray Scale Rendering

We introduce rendering by starting with images attempting to mimic photographs. Below is a picture of a mannequin head, Figure 3. A real head would have been too detailed to be shown here. This picture has been derived from a digitized image. A digitized image is the recording of light through a camera where the light intensities are converted to computer readable form. Image data is usually stored in a rectangular pattern where the x and y dimensions are known. The data for Figure 3 is 256 pixels by 256 pixels or 65,536 pixels.

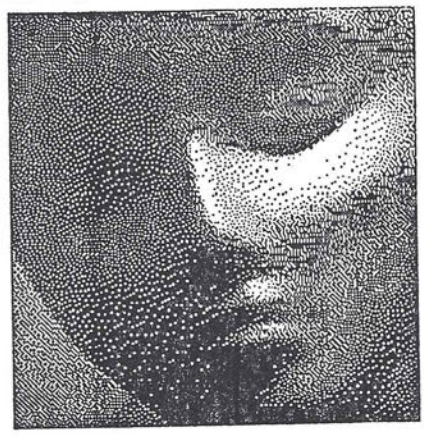


Figure 3. A picture of a human head. This picture was calculated from a digitized image.

A pixel is a picture element and for an image it is the value measured at a single light spot. In order to print the picture of the head it was necessary to calculate the spacing of black and white dots to simulate the variation of light intensity recorded by the digitizing camera. This technique of representing grayness is called dithering [61]. The computer program that calculates the spacing of the black dots does not know anything about the subject of the image. If the result is recognizable it is the the human mind wanting to recognize something and in some way making sense out of the gray pattern.

Figure 4. Recording image data. A video camera is focused upon the subject. The image detected by the camera is converted to computer readable numbers by a digitizing processing board in the computer. These numbers are written directly into computer memory. The pixel values are stored by the control program in a rectangular pattern corresponding in size and arrangement to the camera view. X is the width of the image and Y is the height. After digitizing the image data can be processed like any other data stored in a computer.

While we see and conceptually and emotionally understand images [28], most often real world images are too rich for analysis. One way of reducing the information is to make a thresholded image. To produce such an image, any pixel with a measurement above a threshold value is set to white, or the maximum value, and all others are set to black, or the minimum value. The next picture, Figure 5 is a typical result. The border between the black and white areas shows where the data is at the threshold value.

Figure 5. Thresholded image data. In some places the border between black and white areas is smooth and in other places the border is rough. Texture in the subject of the image together with noise in the recording equipment caused this "roughness".

Even thresholded images may be too complex for mathematical analysis. The next picture, Figure 6 on page 6 shows the result of a rather simple image operation, edge enhancement, which further simplifies the data. The contour of a region can be found by replacing pixel values with the differences of pixels and their neighbors. Any pixel surrounded by pixels of the same color has no differences and are set to zero or black.

The following picture is a typical result. The broad contour line can be further processed to produce a sharper outline. This extraction of lines or curves from an image is called edge detection [5]. By such techniques the amount of data that might be processed is reduced and organized.

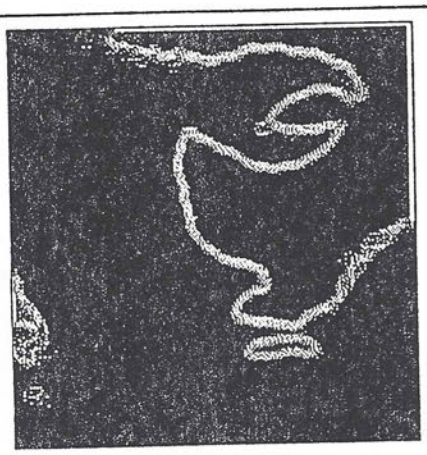


Figure 6. A contour curve on a thresholded image.

Rendering Computer Generated Data

Images can originate not only from digital camera measurements but can be generated from an arbitrary graphics algorithm or data set. The next image is the result of a calculation in an imaginary three dimensional space. A mathematical model of a shape is assumed, the source of light illuminating light at every image pixel is then determined. To do this, take each pixel one at a time. Find which surface lies behind this pixel and exactly what point on the surface lies behind the pixel. Then find how much light falls on this spot in space. If desired, it is also possible to determine how much light was reflected upon this spot in space. If the surface is assumed to be on an object of some transparency, then the refracted light can be calculated. The picture shown below, Figure 7, is of spheres in space including calculations for incident and reflected light. Such pictures can be thought of as artificial photographs. Because the history of light is explicitly calculated along lines or light rays in space, this process is called ray tracing [21].

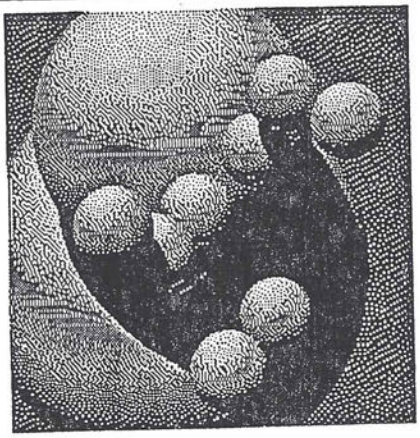


Figure 7. A ray tracing. An artificial photograph is calculated from the illumination by following the reverse history of light to the light source.

A question may be raised about ray tracing: Why follow the reverse history of light from the viewer to the object? Would it not be more direct to follow light rays and their illumination effects from their source? For moderately ambitious ray tracing, it is more economical to trace the history of light in reverse since very few light rays actually contribute to the artificial image, but for the most realistic renderings where subtle illumination effects are to be shown, the complete history of light in the virtual world is needed.

In the printed picture of a human head, Figure 3 on page 4, dithering was undertaken so that the picture can be printed. Images are best shown on computer displays capable of drawing spots of different grays like a television receiver, but dithered pictures of some sort are needed for printing grays with black ink.

The strategy of producing the renderings shown so far was to take an array, a rectangular pattern of values, and convert this array to a dithered pattern of black and white dots. The pattern could then be printed for the purpose of reproducing this document, or displayed on almost any computer screen. This dithered pattern, or any rectangular pattern which is a complete displayable entity is one of many possible

bitmaps or pixelmaps. The word bitmap originates from the engineering of computer displays where every display spot corresponds to one or more storage bits in computer memory. The word pixelmap, sometimes abbreviated to pixmap, comes from the notion that a computer display shows a pattern of pixels and that this pattern can be treated as a total entity in computational operations. An image is a rectangular arrangement of measurements or values, and is shown to an observer by converting it to a bitmap or pixelmap. Frequently the nouns "image", "bitmap", and "pixelmap" are used synonymously. Except for those instances when the conversion process is the subject, this should cause no problem.

A bitmap is converted to a pattern of glowing points shown on the display screen. The overall process is to convert computer memory to a picture. When the memory is changed, the picture is changed. The change of display pictures by changing computer memory is called bitBlt [22]. Aesthetic and perceptual goals motivate designers to make the bitBlt operations run fast and efficiently. The viewer should not be aware that the memory is actually changed bit by bit, memory location by memory location. Satisfactory bitBlt operations such as move, copy, and change color, could not be used until computer circuits and memory control programs attained sufficient speed. With high speed bitBlt, the impression is given that unified rational sections of the screen change.

A common bitmap operation is the reversal of a bitmap for printing. Generally, a bitmap for computer screen display has a maximum value, maybe just 1, for those pixels which should be white, and a minimum value for those pixels that should be black, always 0. For black ink printing, black pixels are always 1, and white pixels are 0. Reversal of a bitmap then is the conversion of the bitmap values. A second bitmap operation is pixel replication. The illustrations in this chapter were generated and displayed on a computer, the pixels on such displays are rather large and have the characteristic that white pixels look bright. Printed pixels are comparatively small. The black pixels that dominate. In order to protect the quality of the illustrations the printed versions replicate pixels, each display pixel is replaced with a pattern, usually 2 by 2 or 2 by 3, with the same value as the original pixel. The new pixmap is now much larger than the original pixmap computational operations on images, Highly interactive displays, such as video games, where bitmaps are manipulated, look extraodinary but have special hardware support to "render" these computations tractable. Workstations with multiple windows and interactive application displays are essentially controlled by the bitBlt screen management programs that keep track of what set of which bitmaps are to be controlled at what time or in response to what user action. The text shown on a screen consist of small bitmaps that have the shape of character fonts.

Rendering Data Sets

Images need not be digitized from life or a virtual world. The next picture, Figure 8 on page 8 shows a typical image which is the result of a two dimensional calculation. In this image, three sine wave functions were added together, and after rescaling to a range from zero to 100, the two dimensional pattern of values was used to produce the dithered picture shown here. In this picture, the two horizontal cycles and the two vertical cycles can be easily seen superimposed over the more complex pattern originating from the lower right corner. This demonstrates that for large two dimensional data fields, data image displays are very effective.

As with images from real life, sometimes a grayscale presentation is too complicated to be easily understood. Simplifying the image can make it easier to interpret. The following image was made with false coloring. Image values were converted from a large range of values to a smaller range of values or maybe just a different range of values like the thresholded picture shown in Figure 5 on page 5. In Figure 9 on page 8 image values below 34 were set to black (new value 0), image values between 34 and 66 were set to gray (1) and image values above 66 were set to white (2). After

dithering, the final bitmap was obtained as shown. The result consists of a contour map which is easily interpreted.

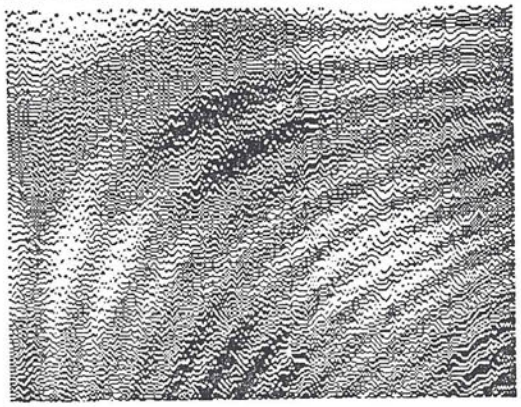


Figure 8. **Calculated data image,** in this case the sum of three two dimensional sine waves.

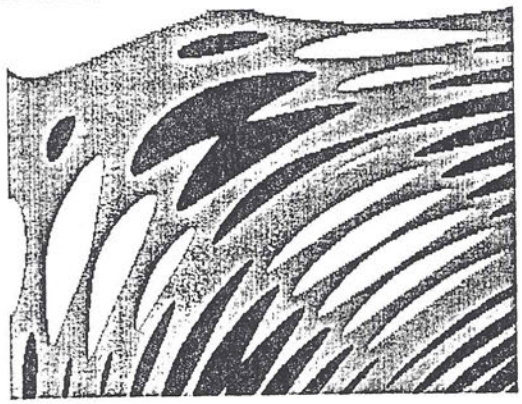


Figure 9. **A false color image** derived from the previous illustration. In this picture, pixels within specific ranges of value were rendered white, gray, or black.

Rendering To Enhance Interpretation

Digitized real world images are difficult to analyze. When we look at a real world picture, such as Figure 3 on page 4 of a human head, the subject is recognized, contours can be deduced and maybe we can estimate the direction in which surfaces are orientated, but it is very difficult. The subject of Figure 10, a cube was especially designed to facilitate this exercise in image processing; the faces were covered with white tape and the edges were painted black. After digitizing the cube, the image was thresholded, and the white background was removed resulting in the image shown in Figure 11 on page 9. It was easy to remove the background since it was separated from the cube by the black edges painted on the cube. Edge detection in image data is the topic of the field of image understanding [5].

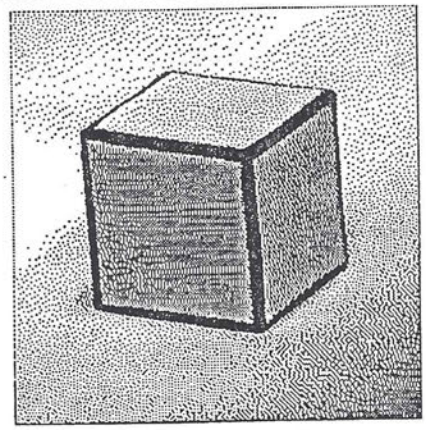


Figure 10. **Black painted edges on white cube.**

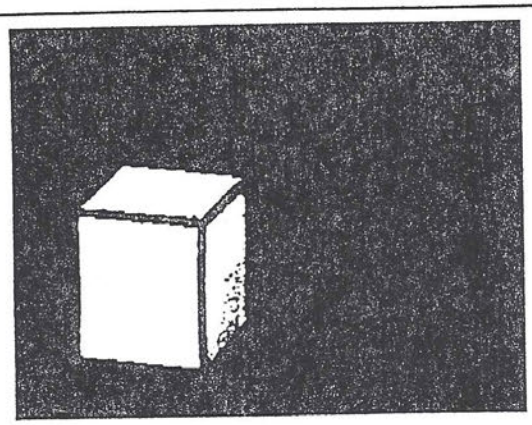


Figure 11. **A thresholded display** of an image similar to Figure 20 on page 12. The difference in proportion is due to the difference in aspect ratio of the computer display x and y pixels and the digitizing camera pixels. The background was removed to allow the faces and front edges of cube to remain.

The preceeding image which shows the isolated frontal edges, contains some noise, but the edges can be identified and the lengths of the three edges can be estimated. If the projective distortion is ignored, the angles of orientation can be taken to be a function of the lengths of the three frontal edges. Figure 12 shows the estimated end points of the frontal edges with small circles. Obviously, the calculation is not perfect. In order to verify the qualitative result, a small picture of a cube has been drawn at the estimated orientation. The value of Figure 12 is not that the problem of determining orientation has been solved, but that the process can be observed and evaluated, and possibly improved. The screen display becomes a graphical workbench: the data of the experiment is on the left, and the result of the calculation is on the right.

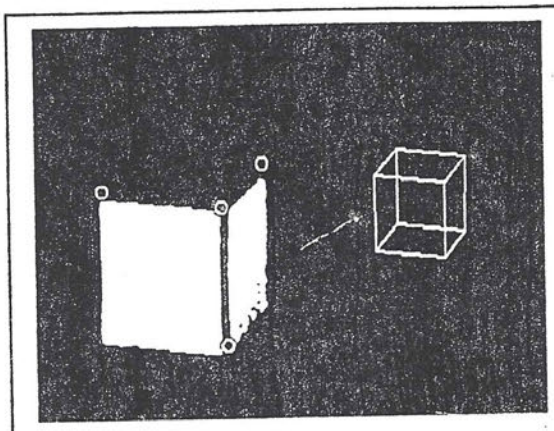


Figure 12. **Measurement of edges.** Circles show the calculated extents of the frontal edges. The ratio of the lengths of these edges allows the orientation of the cube to be calculated. A line drawing of a cube has been made for comparison with the original picture.

Drawing Planes in Space

A classic problem of computer rendering is to show structures. Such objects, in the simplest possible topology, consist of bounded planes in space. The planes may have no thickness but these bounded planes, like the spheres shown in Figure 7 on page 6, are assumed to emulate real material in space. The data structure required consists of a vertex list, that is a table of labeled points in space with their **x,y**, and **z** values, and a topological map; that is how these points are connected together to form planes defined by straight lines in space.

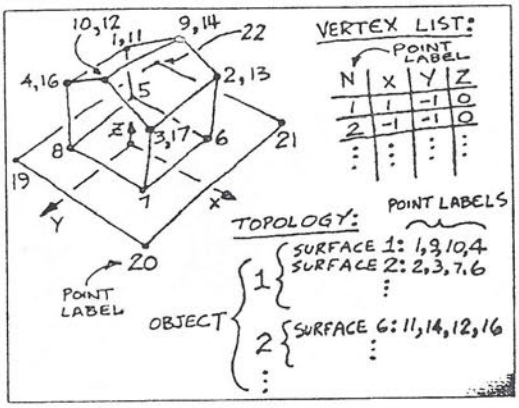


Figure 13. Three dimensional scene creation. A sketch is prepared showing vertex points. A vertex list is made where the three dimensional coordinates are listed for each point. The topology is specified by indicating what points form a polygonal boundary, and which surfaces compose an object.

A perspective projection can now be made of this subject. Examples are shown in Figure 14. In these pictures the position of the viewing eye has been changed to obtain different projections. The style of rendering is called a wire frame because the lines which outline the bounded planes resemble glowing wire filaments in space [43].

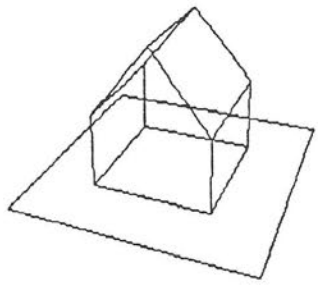


Figure 14. A wire frame rendering.

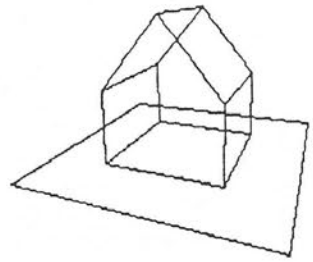


Figure 15. Closer view of the wireframe house. Notice the increased divergence of lines.

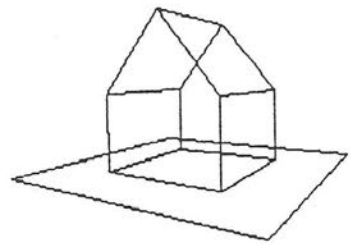


Figure 16. Lower angle view of the wireframe house.

This next picture, Figure 17 on page 11, shows the scene when an effort is made to increase the sense of solidity. The artistic and computational device used assumes that the surfaces that compose the scene are opaque and hide anything that is behind them. The procedure of determining which lines should not be drawn is called hidden line elimination.

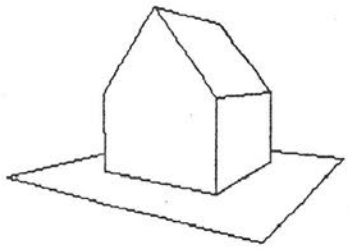


Figure 17. Hidden lines removed on view of the wireframe house as if the surfaces of the structure are opaque.

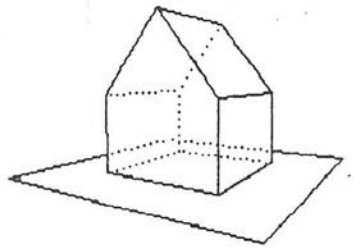


Figure 18. Hidden lines dotted on view of the wire frame house improving the sense of spatial dimension.

Hidden line elimination can be accomplished in many ways. The popular technique used here is called painter's algorithm [46] because it duplicates what happens when a painter colors over a previously painted region. The new paint covers the old. This method works by drawing surfaces, one at a time, from the farthest to the closest. Immediately after a plane is drawn, it is filled in with blackness. Any previously drawn line segments behind a projection of a plane are erased. Figure 18 illustrates a variation of hidden line elimination: after all bounded planes have been drawn with proper visibility, another wire frame drawing is made with dotted lines. The result consists of two types of lines, solid to indicate visibility and dotted for lines that are hidden by a plane. The sense of space and structural content is improved.

This data structure allows individual scene components to be manipulated. For example, in the following picture, the points that define the roof were translated in three dimensional space vertically, thus raising the roof.

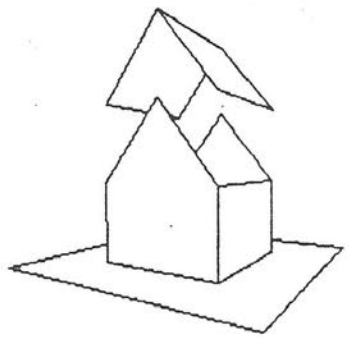


Figure 19. Roof defined as a separate object. It can be distinguished by vertical translation.

Showing the dotted lines usually improves perception. This series of pictures illustrates that a line drawing can define structure as well as position. These pictures of a house were drawn using perspective projection, which emulates the geometrical distortion of natural vision. The further an object is from the viewer, the smaller it will

appear [24]. Unfortunately perspective line drawings distort everything. If the subject of the last few pictures were not a familiar object, (a house), the pictures might not be understood.

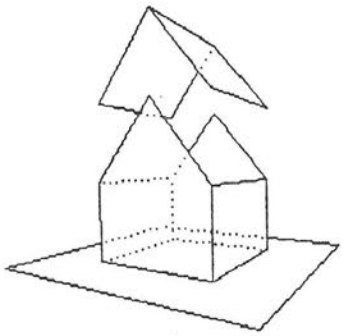


Figure 20. Hidden lines dotted in a house with a separate roof.

Floating Horizon Rendering for Surface Patches

The next seven illustrations demonstrate a very important rendering tool called floating horizon. This technique is probably the most common method of establishing spatial order in three dimensional pictures. It is computationally fast and easily programmed [65].

Assume some data has been generated or measured that can be organized along a two dimensional order such as is shown in Figure 21. This figure shows the data in an isometric projection, which will be described later, with no obvious structural or depth clues. The data is organized on a checkerboard pattern, I, J, where I increases along the x direction, and J increases along the y direction. It helps considerably if the data is plotted as patches in space such as in Figure 22 on page 13 but the absence of depth clues makes this picture ambiguous.

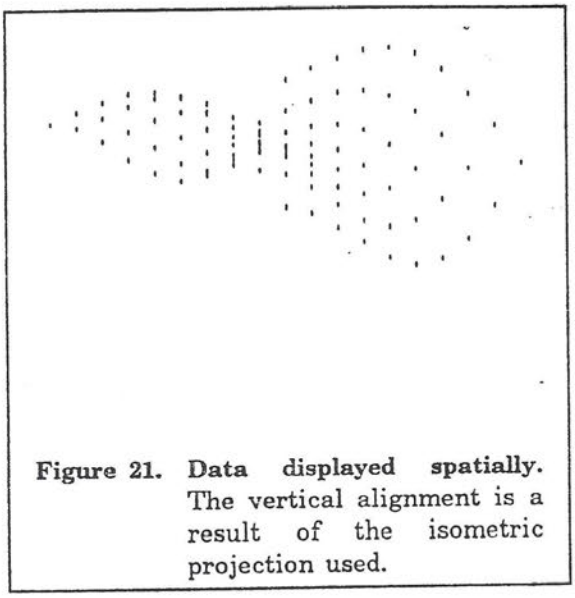


Figure 21. Data displayed spatially. The vertical alignment is a result of the isometric projection used.

Isometric projection is very easily drawn by hand or computer [25]. The calculation goes like this: let px represent the screen's horizontal coordinate and py represent the screen's vertical coordinate. Let x,y, and z be the three dimensional coordinates of a point, let cos be the cosine of 30 degrees (.866) and let sin be the sine of 30 degrees (.5). To calculate px and py:

$$px = \cos (x - y)$$
$$py = \sin (x + y) + z$$

Generally a dot can be drawn on the screen at the location px,py to represent the point x, y, z. To do so on most display systems, however, requires scaling to fit the screen coordinate system. This scaling of data is usually done by software provided with the computer system.

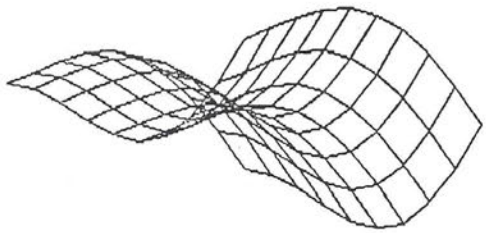


Figure 22. **A wire frame drawing** of the same data. Lines connect points along the I and J order or along x = constant and y = constant curves.

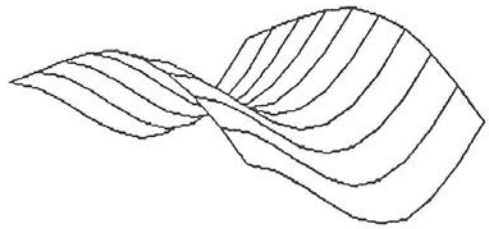


Figure 23. **Floating horizon rendering** of same data. The silhouette envelope consists of a J ruling with sections of an I ruling as end connectors.

The technique of floating horizon requires that a scene be drawn in closed patches, just like one of the patches in Figure 22, and that these patches be drawn in increasing depth. Data such as is shown in Figure 22 is easily drawn in increasing depth because the data indexing is inherently so ordered. The data is organized along I and J addressing so that if the data is accessed by increasing values of I and J, it is fetched in increasing depth. As each patch is drawn, an obscuring silhouette envelope is constructed such that any point that falls within this silhouette will not be drawn. Any point that is drawn must be outside, above or below the silhouette, and upon this point being drawn, the silhouette is updated. Since patches consist of lines, these lines must be drawn as individual dots. Each dot must be compared to the maximum and minimum values of py for each dot value of px. When the picture is started there are no (or some) default values of minimum py (lower horizon) or maximum py (upper horizon), but as the picture progresses, these tables are continually changed. All that is needed is a function to decompose a line into individual screen pixels and a table lookup and update. Once such a function is available there is some artistic freedom for making the picture.

The next five illustrations were all drawn using floating horizon. The only difference between them is the design of patches or obscuring envelopes. If it is desired to draw the surface from differing viewpoints it may be best to reorganize the data because the I and J indexing is used to provide depth information.

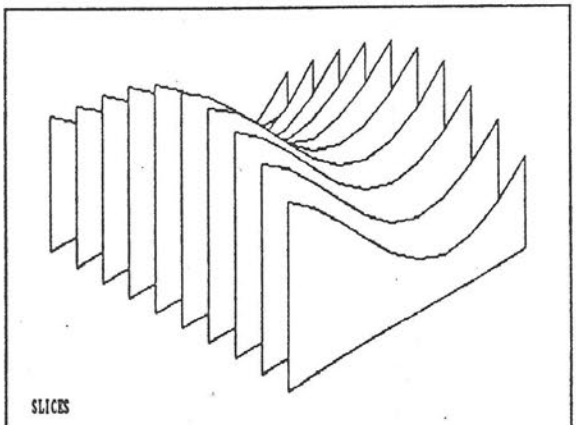


Figure 24. **Sliced rendering.** The horizon envelope is an imaginary slice bounded by a J ruling.

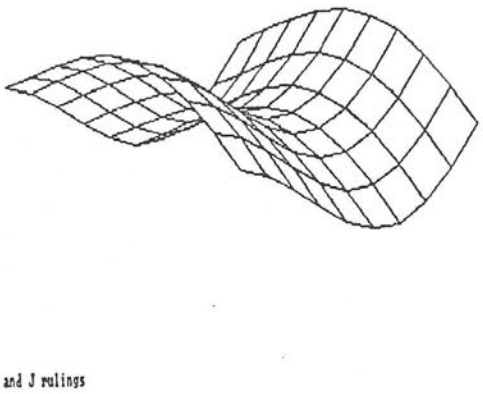


Figure 25. Surface rendering. The horizon envelope is a patch formed by I and J rulings.

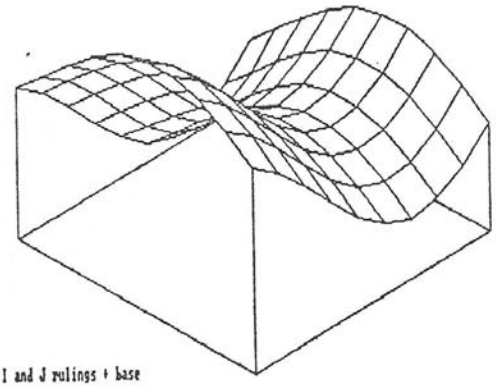


Figure 26. Base added to the same view.

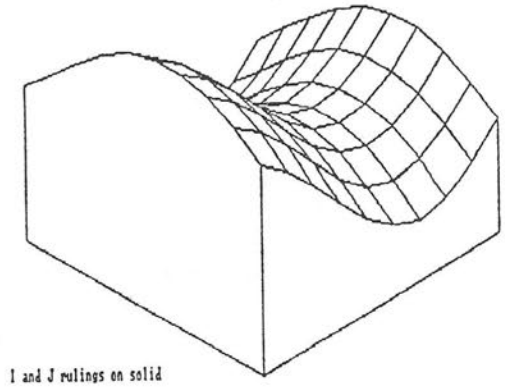


Figure 27. Side horizon envelopes on the same view to create the illusion of a solid.

Isometric projection has some well known advantages. When such a projection is plotted by hand, x coordinates can be drawn or measured along a thirty degree line on the paper, y can be drawn along 150 degrees, and z can be drawn vertically at 90 degrees. All three coordinates can have the same scale factor. Calculation of isometric projection has been shown above. Since the days of hand drawing and continuing into our time of computer rendering, isometric pictures have been very popular because distortion is minimal, regular, and well understood. A simple means exists to actually measure the picture, and depth is easily perceived.

Floating horizon can be used with perspective projection or any other projection. Painter's algorithm could just as well have been used to prepare the last five illustrations. Floating horizon and isometric projection are usually used for complex surfaces that are derived from a large amount of data.

Rendering to Emphasize Structure

All the renderings made so far assume some sort of overall structure or order for the data being displayed, either indexing or topology. This is not always possible; the next series of pictures demonstrate what can be done when no obvious structure exists. Assume that some x, y, z data has been provided. Figure 28 on page 15 might be the first graphical attempt. The points are plotted above a base plane with no depth clues. This is not a bad idea since we might first have to make up some scale factors or at least program some automatic scaling functions. Rendering something quickly is useful when drawing unfamiliar data. The projection scheme used is called oblique [26]. It is suitable for those instances where we know very little about the data.

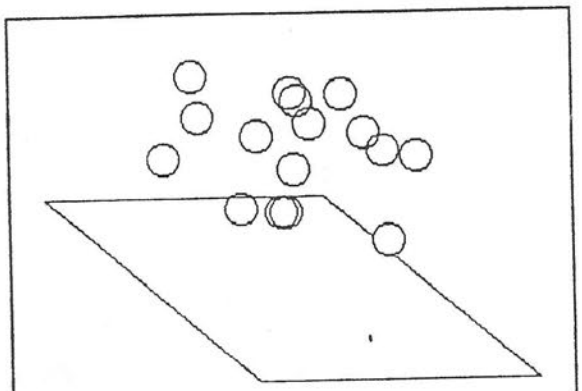


Figure 28. Data floating above a base.

As a first effort to make sense of things we could sort the data points in space and perform a minor painter's algorithm by doing a selective area erase within each plotted circle as it is drawn. Now we can present depth cluing. In an oblique projection the depth sorting need only sort along the dimension that goes into the picture, in this case the y coordinate.

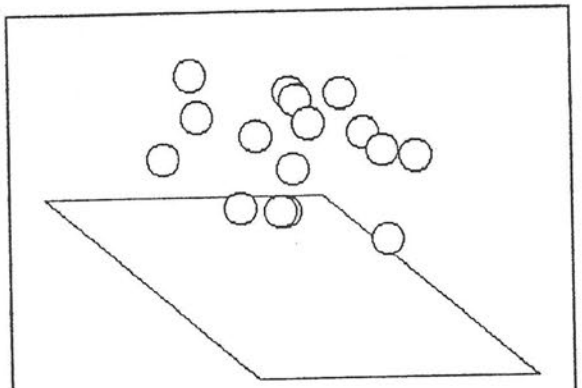


Figure 29. Data drawn in reverse order of depth, each circle occludes space behind it.

In order to provide some structure and to make the depth of the circles more obvious, lollipop sticks can be drawn as in the following picture.

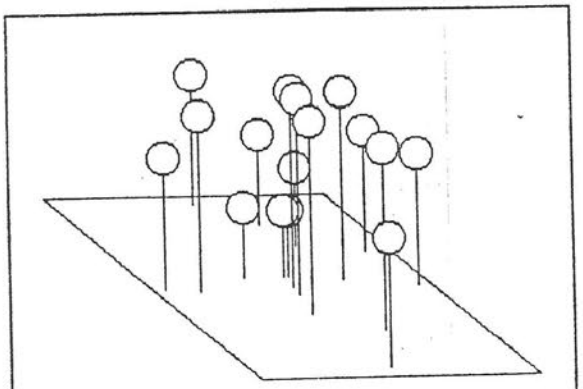


Figure 30. Lollipop sticks to indicate the projected x and y coordinates.

The next embellishment used to enhance depth is called haloing, and is very simple [3]. To halo a graphical element is to outline it in the background color. This emphasizes the difference in depth nicely if the graphical elements are sorted in depth. The circles in the following picture are outlined by black circles, then the lollipop sticks are drawn as three lines: two lines in black surrounding one in white. Notice how the haloing sacrifices a bit of resolution for the improved illusion. The preceding lollipop plot is probably sufficient for a quick look or a rapid presentation of unstructured data. The next few illustrations demonstrate efforts to make sense of the data or find a unifying structure. The structure most commonly hoped for is a mathematical equation. If an equation that fits points can be found, the data can be summarized with the equation and possibly a fundamental truth can be asserted. In the next picture, Figure 31 on page 16, it is assumed that the data lies on the surface of an oblate spheroid. If a series of curves on the spheroid were to be plotted in the oblique projection, the picture would appear as shown.

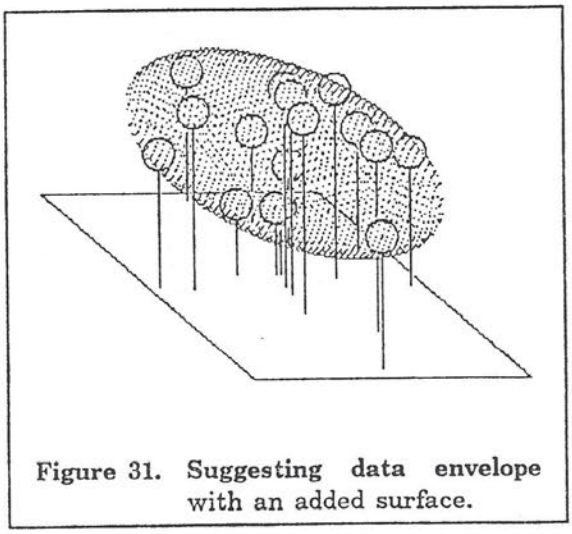


Figure 31. Suggesting data envelope with an added surface.

By drawing the surface, the envelope of data and the surface seem to coincide. Next, individual curves can be drawn to see if they pass through any of the data points. A special result might take place as shown in Figure 32, where one curve passes through four data points.

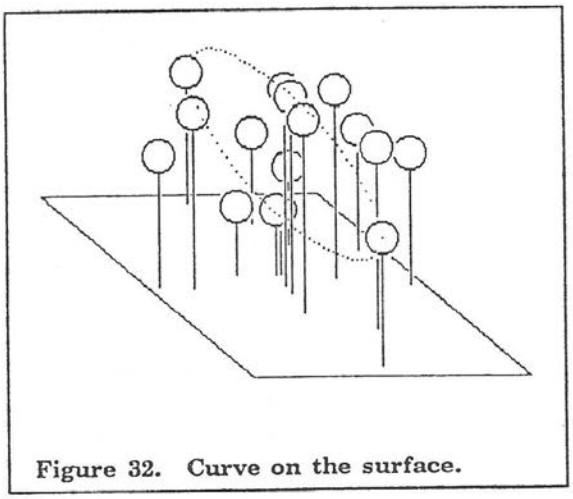


Figure 32. Curve on the surface.

Finally, the entire surface may be drawn as a set of curves subject to the haloing of the lollipops. This is organizationally more complex, but is not difficult. It can be seen that the surface of the oblate spheroid passes through the centers of several data points and on a particular section drawing of the spheroid, three data points in particular lie on a curve of the spheroid.

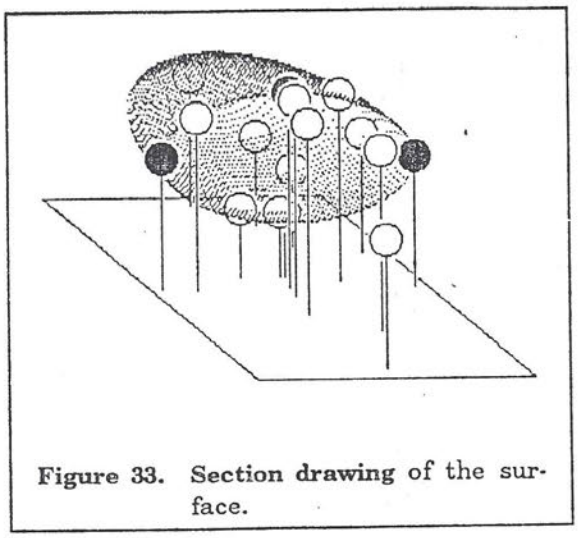


Figure 33. Section drawing of the surface.

This picture presents highlights the relationship of the data to a surface. The data points are well defined in space, the surface passes through the data points, the section is well defined and the lollipop sticks define the x and y coordinates. The value of oblique projection is also used; an oblique projection offers no distortion on planes parallel to the screen, bringing out the true shape of curves parallel to the screen or in the x,z planes. In particular, the last picture shows the shape of the elliptical cross section.

It is unfortunate that it is not possible to cover all topics or techniques of graphical display. This section on rendering has chosen to give examples covering major techniques of presenting pictures graphically. For a more complete description of the field refer to [21]

4. Visual syntax

To interpret a picture or graphical presentation rendered, visual alphabet "characters" are parsed for their meaning by the viewer or program. It is evocative to draw parallels between visual syntax and natural language syntax. The inherent serial nature of natural language relative to the inherently spatial structure of visual language makes the notions of parsing and statements quite different. Natural language relies on articles, prepositions and punctuation for delineating structure and context. Visual language, on the other hand, quite naturally uses spatial structure to represent the context in which information is interpreted. Linguistic terms are used in this chapter solely as points of reference.

Visual parsing distinctions can be separated into syntactical categories:

- positional
- size
- time
- rule

Positional - Positional syntax refers to the use of spatial relationships between visual elements which convey information to the observer. These relationships are recognized in the parse of the language. These include:

- **Relative Positional** relationships:
  - *Sequential* - The spatial sequence of objects denotes an order in which visual elements should be considered. Written languages use sequential syntax, words follow each other left to right and top to bottom. Sequential languages can use space differently. Cartoon frames are often arranged on a line or column (Figure 34).
  - *Metrical* - Measurement is used to denote relative value. Plotting a function uses metrical syntax. Plots using polar, cartesian, projected 3-D and logarithmic scales are examples of metric coordinate spaces. A metrical syntax could use position on a B-spline as its metric. Graphs and bar charts use a cartesian coordinate system to relate two dimensions of information (Figure 35).
  - *Orientation* - Angular relationships between visual objects can drive a visual parse as well. In the case of three dimensional models for example, rotation presents a view of objects or a scene. A rotation of 180 degrees along the horizontal or vertical axis would result in a view of the far side (rear) of the object. For an analog clock, the orientation of the hands with respect to the dial indicates the time of day.

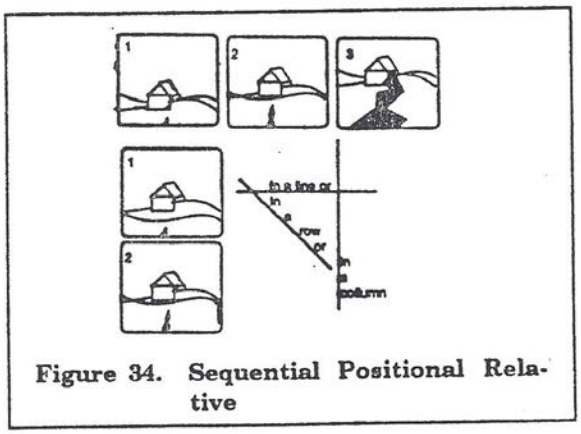


Figure 34. Sequential Positional Relative

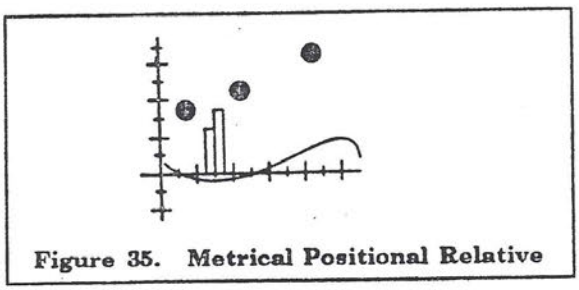


Figure 35. Metrical Positional Relative

- **Interacting Positional** relationships:
  - *Embedded* - Spatial enclosure or geometric containment is often used to define a visual statement. Examples include text balloons in cartoons (Figure 36). VennLisp is a programming language using an embedded notation [34]. A program is defined by positioning hollow language key symbols inside of each other. Evaluation in VennLisp pro-

ceeds from the outside of the diagram inward.

Figure 36. Embed Positional Interacting Syntax

- *Intersecting* - Partial or total spatial overlap of visual objects is another popular way of presenting visual statements. In electrical circuit diagrams, the intersection of straight lines at right angles indicates continuity in the circuit. Venn diagrams use intersecting regions, as well as embedded syntax, to express relationships (Figure 37). Overlapping window systems often use intersection (occlusion) syntax to indicate which windows are active.

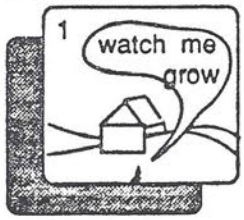


Figure 37. Intersecting Positional Interacting Syntax.

- *Shape* - Spatial relationships, like those found in puzzles, allow syntax to prescribe how things can be connected together (Figure 38). Learning Research and Development Center's Bridge system has pieces which fit together like a puzzle to make programs.

Figure 38. Shape Positional Interacting Syntax.

- **Positional Denoted** relationships: Specifiers, such as lines or labels, indicate relationships between objects.
  - *Connected* - Arcs specify connectivity between visual objects. These connections specify semantic information. Node/arc notations are used in computer flow charts, electrical wiring diagrams, organization charts, etc. Stella is a visual programming language that allows a user to draw several styles of nodes and arcs to specify a program [45].
  - *Labeled* - Spatial relationships can be maintained by symbols. In large circuit diagrams, spatial relationships are not always seen on a single drawing. Labeled wires are used to refer to other drawings (Figure 39).

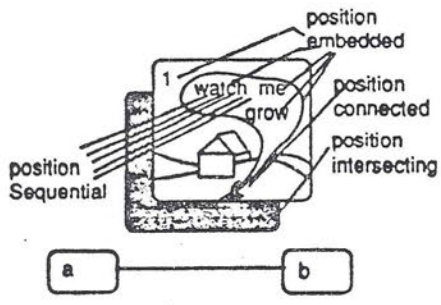


Figure 39. Labeled Positional Denoted

Size - Relative size of objects can influence how and when component items are interpreted (parsing order). Catalogs often contain large pictures of expensive objects, smaller pictures for the less expensive things. Ancient petroglyph languages on stones and cave walls sometimes use size as

the organizing feature. Importance of people and things in these languages are expressed by relative size.

Temporal - Many visual languages use time as an organizing dimension. Computer drawing programs use temporal information to communicate in several ways. For example, the order of spatial overlap can be used to determine recency. Blinking draws the attention of the reader.

Rule - Visual statements can be defined by arbitrary algorithmic visual relationships. Cursors in text editing environments are often designed to stop at the end of a line, or wrap around to the next line.

Visual syntaxes can be combined in relationships using constraints. JUNO [42] and ThingLab [10] use "constraint" languages to define the spatial relationships between displayed objects. Things can "attract" nearby objects. Peridot depends heavily on rules [41]. Peridot defines rules as pairs of one way relationships, rather than mutual constraints.

The Gauges system also uses one way constraints [56]. The system updates a gauge when a variable changes. Moving a gauge needle does not change the variable it represents.

5. Interaction

To talk about interactive systems, we must describe how a visual language's alphabetic primitives are placed and moved in a visual utterance. The user input language, system's visual output language, and application (semantics) are what a user interacts with. Prior discussions of interaction have ranged from interactive vs. batch [40] to the use of direct manipulation [52]. Our classification defines "direct manipulation" as *closed-loop* feedback, *transparent mapping* between action and consequence, and *temporally indistinguishable* response time. Rouse discusses issues of appropriate feedback and response in some detail [47].

Input Syntax is the sequence of actions for communication from a user to the system. The output visual language described above needs to be matched by some set of input "utterances". The input language syntax ranges from concrete keyboard button selection to the abstract shape or gesture.

- *Keyboard Entry* includes text entry and menu selection. The keyboard entry device may be similar to a typewriter and includes variants such as chord keyboards [20].
- *Point and Pick* is a menu selection paradigm. Selection can be made with a mouse, joy stick, digitizing tablet, touch screen or other direct manipulation technique. A menu allows a user to select items from a list, labeled buttons, a set of pictures, or a structured spatial display [53].
- *Point and Move* is a paradigm where an object is selected and moved. Many windowing systems allow the user to point to an object and drag it to a new location on the display. This technique was first demonstrated in the Sketchpad system [57]. Many systems (video games, CAD systems, etc.) use gesture, shape and temporal movement to dictate the "parse" of an image.
- *Point and Draw* is the general drawing technique which leaves a mark wherever the cursor has been. MacPaint relies heavily on this technique [32]. Grail was the first system to demonstrate the use of drawing in an interface [19].
- *Gesture* is the use of motion through space and time. Some mouse drivers (i.e., X-windows [48]) use the speed with which the mouse is moved to accelerate cursor movement on the screen. Handwriting recognition systems can use gesture recognition to interpret characters and editing commands [59, 63, 64].[1]

Input language scenarios can be distinguished by feedback, response time, and interactivity abstraction:

- *Feedback* - A system can be open loop or depend on feedback to support interaction. An open loop system might simply delete a file in response to a user issuing the erase command without indicating success. At the other extreme is the case where a mouse responds immediately to mouse movements or a text editor responds to each keystroke.
- *Response time* is how quickly the system responds to the user. Traditional operating system literature distinguishes real-time, interactive-time and batch-time as definitions for computer response. Response time can vary from being temporally indistinguishable to temporally distant from an action. Fast response in a time-shared computing environment improves an individual's productivity [17].
- *Interactivity abstraction* - A mapping exists between the user's action and the system's response. One extreme is a transparent mapping between user action and system response. A relatively direct correspondence between action and response exists when a person is using a screen based editor. The computer echoes the symbols the person types and moves the cursor around the screen when the arrow keys are pressed. The other extreme is a discon-

[1] Handwriting can also be analyzed through image interpretation techniques.

tinuity between user actions and system responses. This is typified by batch and command line interfaces. An indirect correspondence exists in a text editor when a person types on a command line and the computer responds with a message elsewhere on the screen.

6. Visual Language Composition

We tend to think of visual interfaces as homogeneous. In reality, these interfaces are composed of heterogeneous interacting visual languages. Menus are often augmented with direct manipulation techniques such as text entry. Recognizing how languages are used for different purposes in a visual interface is part of visual language design. Use of separate visual languages in a system can segment function, perspective on information, type of manipulation or system structure.

Structure pertains to the segmentation of activities and commands. A typical text editor has several visual languages in its interface (Figure 40). The text area uses direct manipulation for positioning characters, words and blocks of words with a cursor. A ruler at the top of the window (borrowed from the typewriter) changes appearance as a user types in the text area. A set of commands can be accessed via a menu or command line.

The highly moded structure of the text editor can be held in contrast to the program interpreter for the language BASIC. A BASIC interpreter typically has only one visual language and one set of commands and functions in its interface.

Coverage of the language is described by Shu as one of three dimensions for classifying visual languages [54]. Coverage ranges from special to general purpose. A special language might be a notation for describing organic chemical bonds in a interactive chemistry set program. Other aspects of the interface may include menus and drawing tools. Conversely, a calculator shows all the system's functions with a single language and interface.

The LOOPS system uses a structured icon to present all of its functions [56]. By choosing various parts of the icon, a user accesses various subsets of its capabilities (Figure 41). An instrument is chosen to view and change a variable in a program. When an instrument, such as the pitcher, is chosen, the coverage of the instrument is one variable, not the whole program.

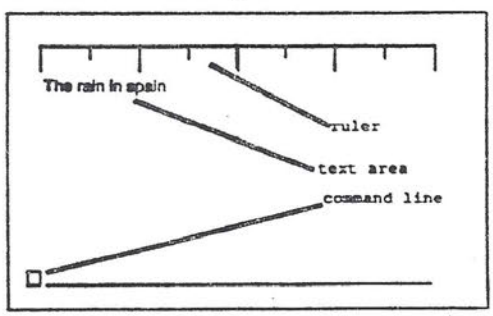


Figure 40. Text editor

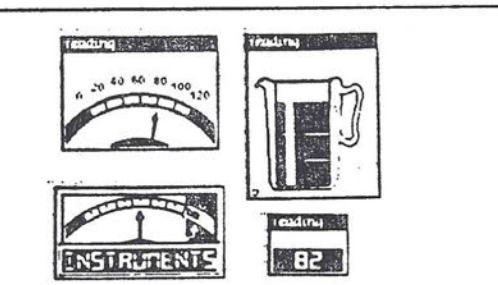


Figure 41. Instruments

7. Visual Language Revisited

By separating elements of visual language embodied in computer graphics, this chapter has set forward a framework for understanding the parts and use of computer graphics. Any use of computer graphics has the following parts:

Visual Alphabet - building blocks of a visual language.
- **Images** photos
- **Icons** caricatures
- **Symbols** designs
  - **Surface** color, texture
  - **Rendering** presentation techniques

Visual Syntax - creating graphic statements
- **Positional** spatial graphic relationships
  - **Relative** sequential, metrical, orientation
  - **Interacting** embedded, intersecting, shape
  - **Denoted** connected, labeled
- **Size** size graphic relationships
- **Temporal** time graphic relationships
- **Rule** defined graphic relationships

Interaction - input language matching the presentation language
- **Input syntax** keyboard, point & pick, point & move, point & draw, gesture
- **Feedback** system response, response time
- **Abstraction** relationship of user input to system's output language

Structure - graphic sub-languages, interaction sublanguages.

This structure can be used as a basis for creating graphic computer interfaces.

8. Computer Graphics Aids For Statisticians

As the elements laied out suggest, a complete use of graphics integrates many utilities to simplify the user's work. The alphabet in which data is rendered, the syntax of how it is to be interpreted and, the interaction language the user relies on all work together. Available software accomplishes various integrations of tools. This section is a brief view of applications, graphics techniques and software which statisticians can use to do their work.

Graphing

Graphs in cartesian coordinate spaces are traditional and powerful analysis tools. Computer software packages enable workers to plot data sets to quickly see trends without having to look at each particular datum. Since they can be generated quickly, if the statistician does not like a particular type of graph he may choose to look at another type, such as cartesian plots, scatter plots, bar charts or pie charts. Even business personal computer spreadsheet systems like Microsoft's Excel or Javalin [7] make these tools available. Tools are available that allow one to view their data as a bar chart, pie chart, or scatter plot, as well as many others.

Rendering data three dimensionally as described above allows the statistician to view data from various angles. Many sophisticated modeling and visualization software packages are available. Programs like MacSpin [18] allow a user to view a set of data convolving and to separate parameters into spatial dimension. For more information, see "The Elements Of Graphing" [13].

Colorising

Colorising differentiates data. To compare two or more sets of data one might plot the graphs of them on the same coordinates in different colors. This would allow a quick visual determination of the possibly subtle differences among these sets. Colorising is a common mechanism for representing data. For example color is used in maps, seismic data, and landsat pictures. Mathematicians use color to represent a third dimension on a two dimensional plot when viewing phenomena such as chaotic functions.

Mathematics + Graphics = Analysis

Integrated analysis packages like MAXIMA[58] and SAS[31] are examples of early successful analysis packages which use analysis to facilitate understanding of data. The more modern MATHEMATICA and S packages integrate graphing as an elementary analysis technique. The 'S' Language is a a programming environment for data analysis and graphics which includes many commands to make graphics easy for the statistician [6]. For example, S provides commands such as PLOT (CORN.RAIN,CORN.YIELD) which produces a scatter plot of the two sets of data CORN.RAIN and CORN.YIELD. If the statistician is not satisfied with the results of this graph she might try PLOT(CORN.RAIN,CORN.YIELD,LOG="Y") which yields a graph with a logarithmic transformation of the y axis.

Many statistics software packages exist even in the inexpensive PC computer market. A comparison of these can be found in PC Magazine [37]. A less market oriented source is "Graphical Mehtods for Data Analysis" [14].

Alternative Presentations

Books like Tufte's "The Visual Display Of Quantitative Information" [60], report achievements using specific views of data for gleaning specific knowledge. Some graphing packages include features which allow comparison of various presentation techniques and alternative presentation views.

Graphics Packages

Two dimensional and three dimensional graphics packages such as the early Movie-BYU have been available for some time [21]. These can be used to display and analyze data sets that don't necessarily need to be physical world data. Modern rendering engines are starting to come equipped

with visualization software like Stellar's AVS system [62].

Graphics Aids Statisticians Can Look Forward To

Interactive Realtime Microworlds.

Computers are now capable of manipulating complex images at speeds which allow interactivity and animation. Proposed visualization techniques allow data to be displayed as an image that can be explored as though it were a place.

New computer scenarios will allow users to interact directly with such physical models. They will be able to change things possibly by pressing on a (virtual) wall or moving a support, and see the stress or other impact interactively.

Seeing large Data Sets

Mechanisms for searching large and multi-dimensional data sets are emerging such as the Xerox ROOMS metaphore [11], and IBM Room With a View metaphore [50].

Automated Presentation

A wealth of representations are becoming available. As the value of the methods in the framework described in section one becomes rationalized, we will come to have software which can aid us further in presenting data. A first step in this direction could be an Artificial Intelligence approach to Automatic Design of Graphical Presentations [36]. In the future systems will describe trade-offs in presentation schemes. They will choose and modify prsentations to enhance analysis. We can look forward to highly interactive three dimensional stereo environments in which the computer will work with us to cull and understand data.

9. Acknowledgments:

Special thanks to Larry Koved for excessive help on previous versions of this chapter. Thanks to Ellen Shay, Kevin Goroway and Chandelle Vuolo for important editing work. Thanks all the attenders of the Visual Representation and Epistemology of Presentation Seminar Series.

10. Bibliography

[1] ACM, editor. Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH. in ACM, editor, *Computer Graphics*, 11(3), Fall 1977.

[2] Adobe. *PostScript Language Reference Manual*. Addison-Wesley Publishing Company, Inc., 1985.

[3] A. Appel, F. J. Rohlf, and A. Stein. The Haloed Line Effect for Hidden Line Elimination. *SIGGRAPH*, 13(2), ACM, NY, August 1979.

[4] R. Arnheim. *Visual Thinking*. University of California Press, 1969.

[5] D. H. Ballard and C. M. Brown,. *Computer Vision,*, pages 143-146. Prentice-Hall, NJ, 1982.

[6] R. Becker, J. M. A. Chambers, and A. R Wilks. *The New S Language*.. Wadworth &Brooks/Cole, CA, 1988.

[7] J. Bernoff, E. Brout, and J. Waldron. *Javelin*. Javelin Software Corp., 1985.

[8] J. Bertin. *Semiology of Graphics: Diagrams, Networks and Maps*. The University of Wisconsin Press, 1983.

[9] C. Bigelow and D. Day. Digital Typography. *Scientific American*, 249(2):106-119, August 1983.

[10] A. Borning. *Thinglab – A Constraint-Oriented Simulation Laboratory*. PhD thesis, Stanford University, 1979.

[11] S. K. Card and D. A. Henderson. Catalogues: A Metaphor For Computer Application Delivery. *Interact'87*, pages 959 -- 964, North Holland Amsterdam, September 1987.

[12] S. K. Card, T. P. Moran, and A. Newell. *Psychology of Human Computer Interaction*. Lawrence Erlbaum Associates, 1983.

[13] J. M. A. Chambers. *The Elements Of Graphing*.. Wadworth &Brooks/Cole, CA, 1983.

[14] J. M. A. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods For Data Analysis*.. Wadworth &Brooks/Cole, CA, 1985.

[15] S. Chang. *Visual Languages*. Plenum Press, 1987.

[16] S. Chang. Visual Languages: A Tutorial and Survey. *IEEE Software*, pages 29-39, 1987.

[17] W. J. Doherty and B. Pope. Computing as a Tool for Human Augmentaion. IBM Research, Yorktown Heights, NY 11622, Jan 1986.

[18] D2Software. *Macspin*. d2Software, Austin TX, 1985.

[19] T. O. Ellis and W. L. Sibley. The Grail Project verbal and film presentation, 1966.

[20] W. K. English and D. C. Engelbart. A Research Center for Augmenting Human Intellect. *Proceedings of the National Computer Conference*, IFIPS, 1968.

[21] J.D. Foley, A.Van Dam, S. K. Feiner, and J. F. Hughes,.
Computer Graphics Principles and Practice, Second Edition,, chapter 14 and 16..
Addison-Wesley, MA, 1990..

[22] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes,.
Computer Graphics Principles and Practise, Second Edition,, chapter 14 and 16..
Addison-Wesley, MA, 1990..

[23] K. S. Fu.
Languages for Visual Information Description.
1984 IEEE Computer Society Workshop on Visual Languages, pages 222-231, December 1984.

[24] F. E. Giesecke, A. Mitchell, and H.C. Spencer.
Technical Drawing,, chapter 17.
The Macmillan Company, NY, 1958..

[25] F. E. Giesecke, A. Mitchell, and H. C. Spencer.
Technical Drawing, chapter 15.
The Macmillan Company, NY, 1958..

[26] F. E. Giesecke, A. Mitchell, and H. C. Spencer.
Technical Drawing, chapter 16.
The Macmillan Company, NY, 1958..

[27] D. Gittins.
Icon-based human-computer interaction.
International Journal Man-Machine Studies, 24:519-543, Academic Press, London, 1986.

[28] R. L. Gregory.
The Intelligent Eye,.
McGraw-Hill Paperbacks, NY, 1970..

[29] D. Huff.
How to Lie With Statistics.
Norton, 1954.

[30] W.H. Huggins and Doris R. Entwisle.
Iconic Communication.
Johns Hopkins University Press, 1974.

[31] SAS Institute Inc.
SAS User's Guide: Basics.
SAS Institute Cary, NC, 1982.

[32] Carol Kaehler.
MacPaint.
Apple Computer, Inc., 1983.

[33] R. R. Korfhage and M. A. Korfhage.
Criteria for Iconic Languages.
in S.K. Chang et al, editor, *Visual Languages*, Plenum Press, 1987.

[34] F. Lakin.
Visual Grammars for Visual Languages.
AAAI '87 Proceedings, Vol. 2, pages 683-688, 1987.

[35] K. N. Lodding.
Iconics - A Visual Man-Machine Interface.
Proc. Nat'l Computer Graphics Assoc., 1:221-233, NCGA, Fairfax, VA., 1982.

[36] J. Mackinlay.
Automatic Design of Graphical Presentations.
PhD thesis, Stanford University, 1986.

[37] PC Magazine.
Statistical Analysis.
PC Magazine, 8(5):103--315, Ziff-Davis, NY, 1989.

[38] A. Marcus.
Tutorial 18: User Interface Screen Design and Color.
ACM/SIGCHI, 1986.

[39] Fanya S. Montalvo.
Diagram Understanding: The Intersection Between Computer Vision and Graphics.
MIT AI Lab. Memo 873, November 1985.

[40] B. A. Myers.
Visual Programming, Programming by Example, and Program Visualization: A Taxonomy.
CHI '86 Proceedings, pages 59-66, 1986.

[41] B. A. Myers.
Creating Dynamic Interaction Techniques by Demonstration.
CHI '87 Proceedings, pages 271-284, 1987.

[42] G. Nelson.
Juno, a Constrain-Based Graphics System.
ACM SigGraph Proceedings, 24, 1985.

[43] W. M. Newman and R. F. Sproull,.
Principles of Interactive Computer Graphics, pages 237, 246-255..
McGraw-Hill, NY, 1973..

[44] P. Reisner.
Human Factors Studies of Database Query Languages: A Survey and Assessment..
Computing Surveys, 13, March 1983.

[45] B. Richmond, P. Vescuso, S. Peterson, and N. Maville.
A Business Users Guide to Stella.
High Performance Systems, 1987.

[46] D. F. Rogers.
Procedural Elements for Computer Graphics,.
<Missing journal>, pages 272-280.,
McGraw-Hill, NY,
1985..

[47] W. B. Rouse.
Human-Computer Interaction in the Control of Dynamic Systems.
Computing Surveys, 13(1), March 1981.

[48] Robert W. Scheifler.
X Window System Protocol, Version 11.
MIT, 1987.

[49] T. Selker and L. Koved.
A Formal Analysis of Visual Language.
IBM Research, Yorktown Heights, NY, 1988.

[50] T. Selker and L. Koved.
Room With A View (RWAVE): A Metaphore Interactive Computing.
IBM Research, Yorktown Heights, NY, 1990.

[51] T. Selker, C. Wolf, and L. Koved.
A Framework for Comparing Systems with Visual Interfaces.
Interact '87 Proceedings, North Holland Amsterdam, September 1987.

[52] B. Shneiderman.
Software Psychology.
Little, Brown and Company, 1980.

[53] B. Shneiderman.
Designing the User Interface.
Addison-Wesley, 1986.

[54] Nan C. Shu.
Visual Programming Languages: A Dimensional Analysis.
Proceedings of the International Symposium on New Directions in Computing, Trondheim, Norway, August 1985.

[55] B. Sproull, W. Newman, and J. Maleson.
The Press File Format.
Xerox PARC, December 1979.

[56] M. Stefik.
The Loops manual.
Xerox PARC, 1985.

[57] I. E. Sutherland.
Sketchpad: A Man-Machine Graphical Communication System.
Spring Joint Computer Conference Proceedings, 23, 1963.

[58] Symbolics, Inc.
Maxima Symbolic Mathematics Manipulation Package..
Symbolics, Inc., 1986.

[59] C. C. Tappert, J. M. Kurtzberg, and Paula S. Levy.
Elastic Matching for Handwritten Symbol Recognition.
IBM Research, Yorktown Heights, NY 9988, May 1983..