Graphical Style

Towards High Quality Illustrations

Richard Beach and Maureen Stone
University of Waterloo and Xerox PARC

Abstract

If there is to be widespread acceptance of computer generated images in areas traditionally served by graphic artists, these images must meet a high standard of quality. Document preparation systems are an application area that is gaining maturity in providing high-quality computer typeset documents. These systems exhibit a trend towards specifying the formatting information for a document separately from the body of the text. The goal is to have the document format designed by someone with expert knowledge of typography. Writers can then apply a format to their own work simply by indicating the semantic content of their text, such as the headings, paragraphs, or footnotes. The result is that a writer can produce properly typeset documents without learning the esthetics of typography. This paper extends this idea to encompass the illustrations in the text. We have developed a prototype system that uses a set of graphical style rules to define the design guidelines for the illustrations. The rules, called a *graphical style sheet*, can be used to control a uniform "look" over a set of illustrations, or to change the appearance of a particular illustration to reflect different publishing styles or different media. The prototype coordinates with an existing document preparation system and the combined systems were used to produce this paper. We conclude that this is a viable method for controlling image style for at least one class of illustrations. This approach contributes to image quality by providing a method for capturing knowledge of graphic arts standards, and for ensuring a consistent appearance of related illustrations within technical documentation.

Categories and Subject Descriptors: I.3.3 **[Computer Graphics]:** Picture/Image Generation – *Display algorithms;* I.3.4 **[Computer Graphics]:** Graphics Utilities – *Picture description languages;* I.3.6 **[Computer Graphics]:** Methodology and Techniques – *Device independence;* I.7.2 **[Text Processing]:** Document Preparation – *Format and notation; languages; photocomposition;* J.5 **[Computer Applications]** Arts and Humanities – *Arts, fine and performing;*

Additional key words and phrases: Graphic arts, graphic design, graphical style sheet, illustration, integrated text and graphics

Introduction

If there is to be widespread acceptance of computer generated images in areas traditionally served by graphic artists, these images must meet a high standard of quality. Increasingly, we see examples such as chart-making systems or spectacular special effects where the quality is defined by the traditional graphic arts standards for print, video or film media. This paper describes the development of tools to improve the quality of technical illustrations.

The inclusion of graphic images into computer-typeset documents is an area of current research and development, for example, PIC [7], IDEAL [18], JANUS [5], Etude [12], and the Xerox Star [10,17]. Typical illustrations which we wish to include are line art and shaded images. Frequently the composition systems which create them are text formatters extended to handle the higher quality output and flexibility available with typesetters and laser printers.

Computer graphics has evolved along two fronts towards quality images: the introduction of new output devices and the development of new rendering algorithms. New devices have higher resolution and more color capability making it possible to render images more precisely. New algorithms that generate smooth curves and more realistic shaded surfaces provide a way to produce high-quality images. Now, artists and designers can expect to find opportunities for creative expression within such systems.

The *style of a document* is a phrase that conveys several meanings, all related to quality. The word *style* in a thesaurus refers to the ideas of fashion, method, beauty, class, and expression. To a graphic designer, the phrase *house style* refers to the customary way that a particular publishing house handles typesetting or illustrations.

Traditionally to a graphic designer, a style sheet communicates to the compositor how to render a document or image [16]. A style sheet, such as the one in Figure 1, provides typographic parameters for typesetting text and guidelines for achieving certain visual effects. The purpose of a style sheet is to ensure consistency and design discipline within a project, and to provide a rapid and effective means for specifying that discipline.

Content		Parameters	Typeface			Typesize		Type-weight					Slope		Leading		Lettering			Color		Column-width		Type-of-column			Indent	
			1.	2.	3.	1.	2.	light	medium	bold	semi-bold	extra-bold	regular	italic	solid	points	all-caps	all-lower	initial-cap	black		pica	justified	unjustified	centred	flush-right	points	
Title	main																											
	subtitle																											
Heading	1st level																											
	2nd level																											
	3rd level																											
	4th level																											
Text	quotations																											
	normal																											
	footnotes																											
	captions																											
Tables	captions																											
Numbering	folios																											
	illustrations																											
	footnotes																											

Figure 1. TRADITIONAL STYLE SHEET for specifying typographic parameters. The rows indicate the parts of the document to be treated specially. The columns indicate the typographic parameters which the compositor uses when typesetting this job. Entries in the matrix are either checkmarks or numeric values.

Computer typesetting systems have provided style mechanisms for text composition through formatting macros or style sheet databases. The '-ms' macro package supplied with TROFF is an example of formatting macros that implement document style [11]. SCRIBE uses *document types* to establish the formatting details for a variety of document styles [15]. The Xerox Star uses *property sheets* to select parameters to control the appearance of selected items of text and graphics in documents [10]. In each of these systems, some mechanism is provided to manipulate the content of a document separately from the appearance of the document. Thus consistency and design discipline can be achieved. This is accomplished without forcing the author to become a graphic designer while the graphic designer can supply specialized knowledge to create the document style.

Extending these formatting style techniques to include graphic images is a natural evolution. In the following sections we describe our concept of graphical style for illustrations and also the artwork-rendering prototype that we integrated with an existing text composition system. The illustrations we consider will be line drawings, although a provision for continuous tone images will also be described.

Examples of Graphical Style

To motivate our concept of graphical style, we present two examples taken from traditional graphic arts productions. The first example presents observations on some stylistic aspects of *Scientific American* illustrations. The second example describes how a consistent style was achieved in producing a book having many line drawings.

Scientific American Illustration Style

Scientific American has established a reputation for the clarity and effectiveness of its illustrations. In Figure 2 taken from the recent article, 'Artificial Intelligence,' in the October 1982 issue of *Scientific American* [19], we can observe several aspects of the *Scientific American* style. Lines are generally thin, although with different weights to convey detail, arrowheads are always the same open design, lettering is always 8-point Helvetica capitals, shades of grey and colors are used only when needed and then only to convey essential meaning, and the design is clean and carefully crafted. While the author supplies the concept sketch and the illustrator renders the image with great skill, it is the art department that ensures that the traditional style of *Scientific American* illustrations is maintained [4].

Traditional Illustrated Book Production

A recent textbook for introductory computer science co-authored and typeset by the first author of this paper [6] required a large number of illustrations. Many of the illustrations were listings of computer programs and their output. With a suitable typeface and the text files containing the original programs and computer output, these figures were easily composed directly into the main body of the book. In contrast, there were over 150 line drawings that were hand-drawn by a draftsman. The production of these illustrations presented a considerably different problem.

Illustration guidelines were written to establish the desired style. The authors and the book designer described how various details were to be handled by the draftsman for each type of illustration: mathematical graphs, Pascal syntax diagrams, data structure diagrams, and simple line drawings.

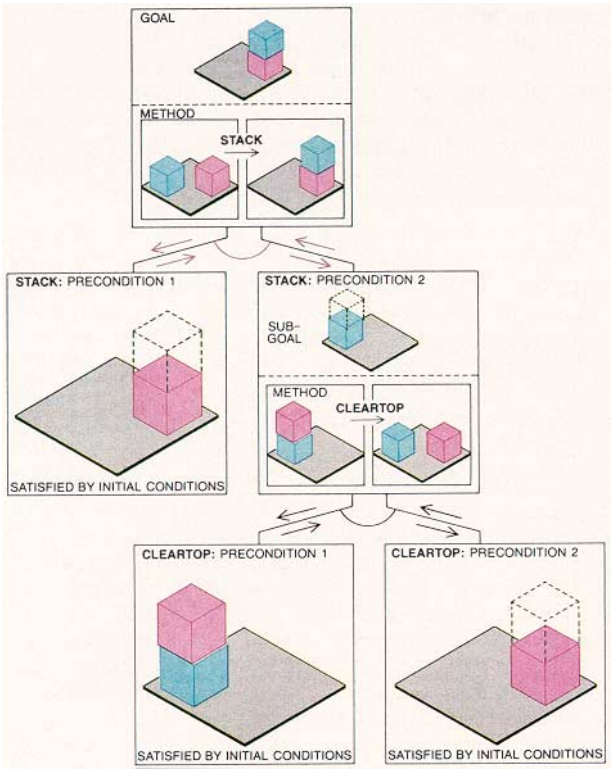

Figure 2. SCIENTIFIC AMERICAN STYLE for illustrations is evident in this one from David Waltz' article 'Artificial Intelligence' [19] in the October 1982 issue, page 122. Several stylistic aspects can be noted: the thin line weights, the open arrowhead design, the use of color and shading, and the caption typography. (Used with permission of W.H. Freeman & Co.)

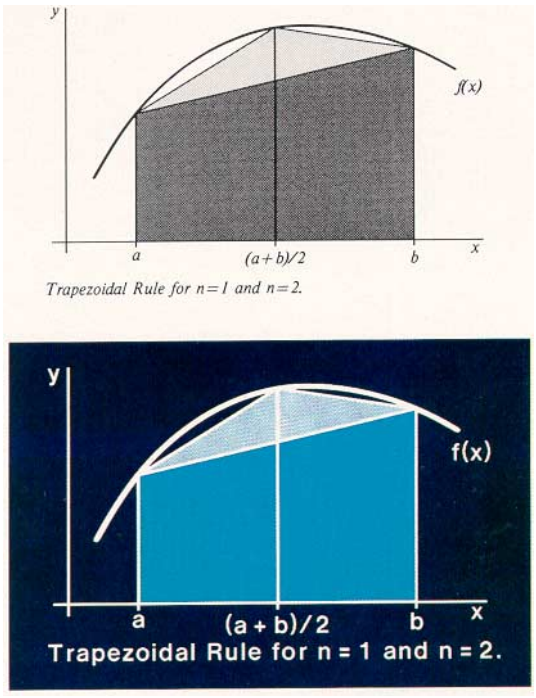

Figure 3. TRAPEZOIDAL RULE FIGURES demonstrate the use of graphical style to produce two very different visual effects. The top illustration is adapted from *Computing* [6] by redrawing it with the Griffin illustrator. The style is faithful to the hand-drawn original. The bottom illustration uses the same picture file but with a style appropriate for a 35 mm. color slide. The slide has the preferred format with light detail on a dark background, thicker lines in white, a larger, bolder and simpler typeface, and the caption is formatted to the slide width. (Used with permission from Reston Publishing Co.)

For instance, the guidelines specified the different line weights for the axes and curves in graphs, the typography for labels on graphs and syntax diagrams, the shading technique for areas in graphs and simple line drawings, and the treatment of arrows in syntax and data-structure diagrams. These guidelines were organized by illustration categories, and then by illustration components. Thus the guidelines for graphs specified the treatment of axes, curves, areas, intersection points, axis tick marks, axis labels, and curve functions; and the guidelines for syntax diagrams specified the treatment of terminal symbols, nonterminal symbols, and grammar rules. Unfortunately, there was no computer support available for drawing the illustrations that could produce sufficiently high quality output or that could implement the various guidelines.

The book's illustrations were also used to produce overhead transparencies for lectures. The computer programs and output could easily be reformatted with a larger type size suitable for projection by specifying a different text formatting style. The hand-drawn artwork had to be enlarged photographically. When an illustration was scaled to transparency size, much of the detail in the illustration was often too small to be read in a large lecture hall. A style substitution facility for graphical images, similar to the one available for text, with the ability to change the proportions of line weights, to use different shading, and to request larger, bolder, text captions would have been invaluable.

Graphical Style Sheets

A graphical style sheet is a way of describing the design guidelines for an illustration. A set of figures produced with the same style sheet should "look" related. It should also be possible to dramatically change the appearance of a figure by specifying different styles, as shown by the book style and transparency style for the Trapezoidal Rule illustration in Figure 3. This implies that there is some separation between the style sheet and the specifics of a particular illustration. The style sheet should contain rendering information specified in some semantic way. For example, there might be a rule that specifies how all axes for graphs should be rendered. We need to determine, therefore, how to separate an illustration into content versus format, and how to specify the format in terms of rendering attributes.

Content versus Format in Illustrations

To apply the *content versus format* discipline found in text composition systems to illustrations, it is necessary to define the content of an illustration separately from its format. The illustration's content is analogous to the author's rough sketch given to a draftsman, such as Figure 4, and its format is analogous to the appearance of the finished artwork rendered by the skill and craftsmanship of the artist, for example, Figure 3. The sketch is described by geometrical objects and their positioning, while the artwork rendering is described by design guidelines and drawing techniques.

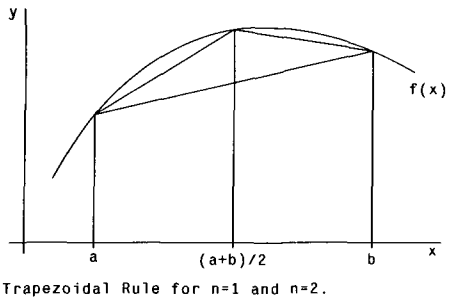

Figure 4. SKETCH OF THE ILLUSTRATION for Figure 3 represents the basic geomety of the picture. All the rendering information for the figure has been reduced to drawing thin lines and using a typewriter-like typeface. The three examples of the Trapezoidal Rule have all been produced by using the same TiogaArtwork file but with appropriate differences in the style rules.

In a manner analogous to the style mechanisms of text formatters, there must be an additional means of including semantic notions in an illustration. Just as not all three-space indentations are paragraph indents, not all thin lines are axes on a graph. Therefore, graphical style must include mechanisms for capturing the intent of the author/illustrator. One way to do this is to provide a level of indirection which names the semantic parts of the illustration. The rendering attributes and guidelines associated with these names are defined separately in a graphical style sheet. If this level of indirection is available, then it is possible to render the same illustration in quite different ways by changing only the graphical style definitions. For example, Figure 3 shows two graphical styles with thin, clean lines for a typeset book and with wider, bolder lines and colors for a color transparency.

Rendering Attributes

A graphical style sheet must express how an illustration is to be rendered. Basic drawing attributes supported in most graphics packages are obvious candidates for specifying how an artwork rendering program should produce an illustration. Examples of such attributes appear in the Griffin illustrator [3], the GKS standard workstation attribute model [2] and in the Xerox Star basic graphics feature [10]. These examples suggest that at least line weight, line patterns, color specification, and caption typography parameters be included in any graphical style sheet.

Graphic designers frequently rely on mechanical aids and transfer sheets to obtain consistent or special effects, suggesting other sources of rendering attributes. A standard reference for transfer sheet designs is the Letraset Catalog [1]. Additional rendering algorithms can be created to produce some of those effects. For instance, a line in an illustration sketch might be rendered by specifying an arrow design in the graphical style sheet to be drawn along that line. Similarly, borders or texture patterns might be rendered from details provided in graphical style sheets.

The Prototype System

Overview

To experiment with these ideas of graphical style we developed a prototype suitable for a class of technical illustrations The system, named TiogaArtwork, was designed to coordinate with the Tioga document preparation system and an interactive illustration program called Griffin [3]. The procedure for generating a figure is to make a draft using Griffin and then convert the figure to a special kind of Tioga document. Once the figure is in document form it is possible to adjust both the style and the content using a combination of Tioga and TiogaArtwork. All of the figures in this paper, except the published example in Figure 2, have been produced using this technique.

The TiogaArtwork system was developed in the Cedar programming environment [14], which is a research project at Xerox PARC. Cedar is both a language, derived from Mesa [13], and a computing environment. The hardware for this environment is a Dorado processor [9] with a 1024 by 808 pixel bi-level display and, optionally, a 640 by 480 pixel color display. A range of medium- to high-resolution printers is available. The highest-quality printers produce digitally half-toned images at phototypesetter-compatible resolutions either in black and white or as color separations.

The design of the prototype is based on features of the Tioga document preparation system, so it is necessary to briefly describe Tioga before going into the details of TiogaArtwork. The current implementation of Tioga consists of an interactive text editor and a batch-oriented typesetter. Documents in this system consist of two files: a file of text nodes and a file of formatting style rules. The text nodes in Tioga have a hierarchial structure similar to that of the NLS editor [12]. A document is a tree-structured hierarchy of nodes containing text, for example, in section, subsection, paragraph order. A normal tree-traversal results in the familiar form of the document. Each node in the document can possess certain *node properties.* The principal use of these properties is to associate the formatting style rules with the nodes. Figure 5 represents the Tioga document structure of some surrounding nodes of the document for this paper.

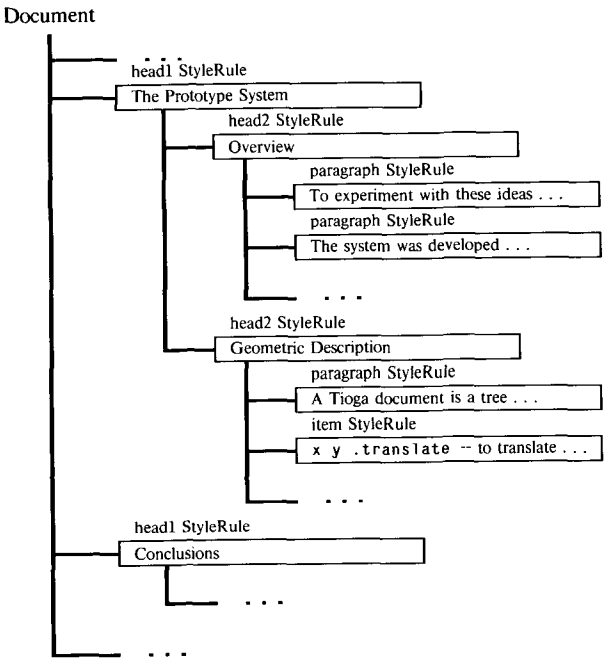

Figure 5. DOCUMENT STRUCTURE provides a hierarchy for organizing a text document. This illustration represents the Tioga document structure for the nearby sections and subsections of this paper. The boxes represent text nodes and the labels above each box represent the node properties. The StyleRule property indicates the formatting attributes for first-level headings, second-level headings, paragraphs, and items within a list.

The style concept in the Tioga formatter is similar to that in SCRIBE [15]. In Tioga, formatting styles are defined by a collection of rules contained in a separate file. The style rules specify formatting parameters such as type family, style, and size, indentation, interline leading, text justification mode, and composition layout parameters. The formatting rules are written in an interpreted language, so it is possible to compute parameters during the formatting process.

We have designed a way to include nodes containing graphics in a Tioga document. These nodes contain a textual representation of geometric attributes such as line coordinates, curve control points, positions, and transformations. The style rules associated with these nodes specify graphical parameters and rendering attributes. Tioga's node properties are implemented as named property lists, so we defined a new property, which we called `ArtworkClass`, for the illustration nodes. The style machinery for Tioga is extensible, so the current set of tools allows us to build on the existing mechanisms for manipulating styles and for adding graphical style attributes. It also means that our system can use Tioga facilities for text formatting.

A document in our system, therefore, is a tree of nodes arranged in a hierarchical structure. Some of the branches of the tree represent paragraphs of text and some represent illustrations. An illustration is a subtree in the document tree-structure with its root node possessing an `ArtworkClass` property. Nodes within the subtree structure may be either graphics or text. Nodes representing subpictures will have the `ArtworkClass` property, while text captions within an illustration will have only the standard Tioga properties. This recursive relationship is important; it means that our system can use all the text formatting features of Tioga for text inside of illustrations. Figure 6 represents the Tioga document node structure for portions of the Trapezoidal Rule illustrations in Figures 3 and 4.

TiogaArtwork is the part of the system that interprets the graphical nodes and style rules. The pictures can be previewed on a display screen using the Cedar graphics package [21] or converted for printing.

The combination of using the Tioga document structure and representing the graphics as text allows TiogaArtwork to interact easily with the Tioga editor and the Tioga typesetter. This is advantageous in a number of ways. One advantage is the ease of creating and editing figures. The Tioga editor does not react to the `ArtworkClass` property, so the text and style properties for a graphics node can be edited in the normal manner. This means that for the prototype experiment we did not have to write a special editor to manipulate the graphics, although we did write a conversion routine which translated from Griffin format to Tioga node structure and automatically generated style rule properties from the Griffin style attributes.

Another advantage of using the Tioga document structure for illustrations is that it permits us to typeset any text in the illustration. This is accomplished by setting up a call-back mechanism between the typesetter program and TiogaArtwork. As the document tree is traversed, the typesetter formats text nodes in the normal fashion. Whenever a node with the `ArtworkClass` property is encountered, that entire subtree is passed to TiogaArtwork. If TiogaArtwork subsequently encounters a text node while rendering the illustration, it passes the text branch back to the typesetter. This recursion is guaranteed to terminate at the end of the tree-path traversal.

Trapezoidal Rule Figure

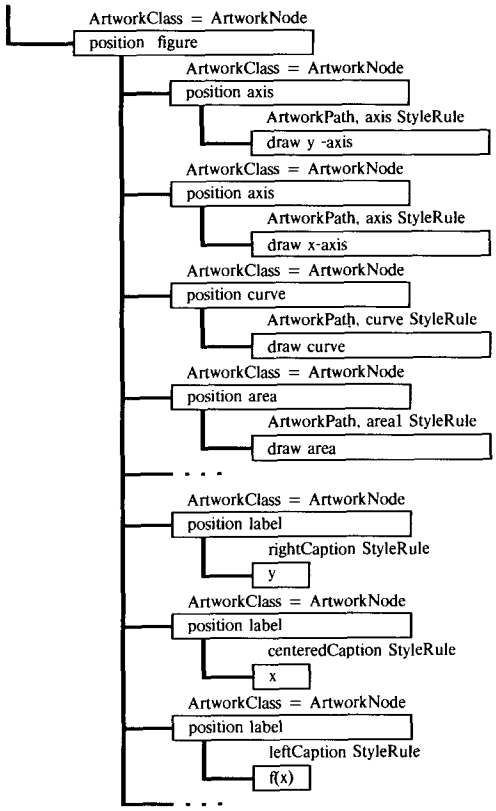

Figure 6. ILLUSTRATIONS in TiogaArtwork also use the Tioga document structure. The boxes represent the artwork nodes which contain the geometrical representation, and the labels above the boxes represent the node properties. Note that StyleRule properties exist on both text and artwork nodes. This structure can be extended to encompass pictures composed of many subpictures and text captions in a natural hierarchy.

The call-back mechanism is also used to pass dimensional information between the two programs. The typesetter provides the dimensional parameters for the formatted text with which TiogaArtwork can layout the caption within the illustration according to the style rule. TiogaArtwork generates the dimensions of the formatted graphics so the typesetter can layout the figures with the paragraphs on the page.

This document structure and system design gives us a great deal of flexibility for experimenting with the best way to represent illustrations in a document. The current structure puts only geometric information in the document body and puts all rendering parameters in the style rules. We found that this representation gives a good semantic description of the picture. The next sections will describe our current implementation in more detail.

Geometric description

A Tioga document is a tree of nodes. Some of the branches of the tree represent paragraphs of text, and some represent illustrations. For text, the levels of the tree represent the section-subsection-paragraph hierarchy of the document. For illustrations, the hierarchy represents a relative set of transformations for subpictures. Each subpicture in the illustration is drawn relative to the coordinate system established by its ancestors. An `ArtworkClass` node,

therefore, may contain a set of coordinate transformations which establish the coordinate system for this node and all of its descendant subpictures:

`x y .translate` -- to translate the origin to <x,y>

`sx sy .scale` -- to scale by x,y scaling factors

`r .rotate` -- to rotate by r degrees

A geometrical shape can be composed of straight lines and curves, and a sequence of these lines and curves is called a *path* [21]. A path can represent a line, an area or a clipping region. The path definition is provided by commands to draw lines and curves:

`x y .moveto` -- establish the current path position <cx,cy> as <x,y>

`x y .lineto` -- draw a line from the current position to <x,y>, and reset <cx,cy> to <x,y>

`x1 y1 x2 y2 x3 y3 .curveto` -- extend the path with a curve which has the four Bezier control points <cx,cy>, <x1,y1>, <x2,y2>, and <x3,y3>, and reset <cx,cy> to <x3,y3>

For the kind of pictures we are working with, the paths and transformations are all that is specified in the nodes of an illustration document. The rest of the rendering information will be supplied by the style rules. Figure 7 is an extract from the geometric description used for the three instances of the trapezoid rule diagrams in Figures 3 and 4:

For reasons which become apparent during the discussion of rendering algorithms, it is necessary to distinguish between transformations and path definitions in the contents of an `ArtworkClass` node. It is also convenient to create artwork nodes which specify the file name of a TiogaArtwork illustration. Continuous-tone images stored as files are another category of illustration that can be accommodated by the `ArtworkClass` property. The values of the `ArtworkClass` property are the following:

`ArtworkNode` -- node contains the textual representation of an illustration which is not a path, such as transformations

`ArtworkPath` -- node contains the geometric definition of a path

`ArtworkImage` -- node contains the name of a continuous-tone image file stored as an array of intensity samples

`ArtworkFileName` -- node contains the name of another TiogaArtwork file

Basic style parameters

Style rules are described in an interpreted language with each format rule expressed as a procedure definition. A typical rule describes a list of parameter-value pairs which will be stored in a global association list. The general format for this in the Tioga editor is:

```
(name-of-style-rule) "commentary
      describing the style" {
       value parameterName
       value parameterName
       ...
       value parameterName
       } StyleRule
```

The `name-of-style-rule` refers to some semantic aspect of the illustration such as axis, curve, or nonterminal. Values are either numbers or keywords and may be expressions. Values which are distances may be expressed in most

```
% TiogaArtwork figure for Trapezoid Rule
  % Cluster 1
  0 0 .translate 1 1 .scale 0 .rotate
    % y-axis
    11 31 .translate 1 1 .scale 0 .rotate
      1 1 .moveto 1 185 .lineto
    % x-axis
    3 39 .translate 1 1 .scale 0 .rotate
      1 1 .moveto 249 1 .lineto
    % curve
    27 87 .translate 1 1 .scale 0 .rotate
      1 1 .moveto
      8 17  15 33  25 49 .curveto
      43 78  71 106  105 113 .curveto
      131 118  161 110  185 97 .curveto
      194 92  201 87  209 81 .curveto
    % area from a to (a+b)/2
    51 39 .translate 1 1 .scale 0 .rotate
      1 1 .moveto 1 97 .lineto
      81 161 .lineto 81 1 .lineto
      1 1.lineto
    % area from (a+b)/2 to
    ...
    % y-axis label
    8 216 .translate 1 1 .scale 0 .rotate
      y
    % x-axis label
    247 36 .translate 1 1 .scale 0 .rotate
      x
    ...
```

Figure 7. GEOMETRIC REPRESENTATION of an illustration in a textual form consists of comments, transformations, and path definitions. The Trapezoidal Rule illustration was first drawn with the Griffin illustrator [3]. It was then automatically converted into a TiogaArtwork document generating the node structure and node properties from the Griffin illustration file. The indentation indicates the node structure. The comments, which begin with percent signs, were added manually by editing the document text.

convenient units. For instance, 2-point line weight can be expressed as `2 pt`, colors can be expressed by keyword color names such as `red`, `darkBrown`, or `lightBlue`, and relative colors can be evaluated as some percentage (such as `75 percent`) of the brightness or saturation of a named color.

The following basic set of drawing attributes were defined as graphical style parameters:

`lineWeight`	the line thickness
`pathType`	the path area/outline type: `filled`, `outlined`, `filled+outlined`
`penType`	the pen shape: `round`, `square`, `rectangular`, `elliptical`, `italic`
`penHeight`	the pen height as a proportion of `lineWeight`
`penWidth`	the pen width as a proportion of `lineWeight`
`penAngle`	the rotation of the pen, in degrees from horizontal
`areaColor`	the color of filled areas: hue, saturation, brightness
`outlineColor`	the color of outlines: hue, saturation, brightness

The following set of style attributes are supplied by the existing formatter but are interpreted by the

artwork-rendering software for illustration captions and labels:

`family`	the type family, such as Helvetica or TimesRoman
`size`	the type size
`face`	the type style: `regular`, `italic`, `bold`, and `bold+italic`
`captionFormat`	the text justification mode: `flushLeft`, `flushRight`, `centered`, or `justified`
`captionAlign`	the vertical text justification mode: `flushTop`, `centered`, `baseline`, or `flushBottom`
`lineLength`	the length of caption lines
`leftIndent`	the left indent for captions
`rightIndent`	the right indent for captions
`leading`	the spacing between lines
`textRotation`	the rotation of the text line, in degrees from horizontal
`textColor`	the color of caption text: hue, saturation, brightness

To demonstrate both the style language and the graphical style attributes, Figure 8 shows the two style definitions necessary for Figure 3.

Rendering the illustration

TiogaArtwork translates our representation of an illustration into a set of calls on the Cedar graphics package [21] . The graphics package implements a full set of transformation and clipping operations. Shapes are described as a set of analytical outlines that are filled either with a flat color or an image.

The geometry in our documents consists of paths and transformations. The transformations translate one-for-one into calls on the graphics package. The path definitions are compatible with the outline description required by the graphics package. Thus, if a node contains a path with the `pathType=filled`, then the path represents a closed area to be colored with `areaColor` and can be easily rendered.

If the `pathType` is `outlined`, then it represents the center-line of a line or pen stroke. A thick line is drawn along this path as defined by the `lineWeight` and pen parameters. Predefined pen shapes are round, square, rectangular, elliptical, and italic. Other style parameters control the aspect ratio and rotation of the pen shapes. The graphics package does not currently support a pen semantic, so TiogaArtwork reduces this description to an outline. This definition of a thick line is similar to that of a stroke in Metafont [8].

Extended style semantics

This basic graphical style machinery can be extended to express more complicated style semantics. The following examples of shadows, arrows, and borders are based on common graphic arts practice.

Shadow Styles

Two-dimensional shadow effects can be created to emphasize an object. Two simple examples of shadow effects are drop

```
% TrapezoidBook.Style

BeginStyle

(BasicGraphics) AttachStyle

(BasicText) AttachStyle

(axis) "x,y axes" {
   black outlineColor
   outlined pathType
   1 pt lineWeight
   } StyleRule

(area1) "dark areas" {
   grey areaColor
   filled pathType
   } StyleRule

(area2) "light areas" {
   lightGrey areaColor
   filled pathType
   } StyleRule

(curve) "function line" {
   black outlineColor
   outlined pathType
   2 pt lineWeight
   } StyleRule

(leftCaption) "..." {
   "TimesRoman" family
   8 bp size
   italic face
   flushLeft captionFormat
   flushTop captionAlign
   0 leftIndent
   } StyleRule

(centeredCaption) "..." {
   ...

(rightCaption) "..." {
   ...

EndStyle
```

```
% TrapezoidSlide.Style

BeginStyle

(BasicGraphics) AttachStyle

(BasicText) AttachStyle

(axis) "x,y axes" {
   white outlineColor
   outlined pathType
   2 pt lineWeight
   } StyleRule

(area1) "dark areas" {
   orange areaColor
   filled pathType
   } StyleRule

(area2) "light areas" {
   lightYellow areaColor
   filled pathType
   } StyleRule

(curve) "function line" {
   white outlineColor
   outlined pathType
   4 pt lineWeight
   } StyleRule

(leftCaption) "..." {
   "Helvetica" family
   12 bp size
   bold face
   flushLeft captionFormat
   flushTop captionAlign
   0 leftIndent
   white textColor
   } StyleRule

(centeredCaption) "..." {
   ...

(rightCaption) "..." {
   ...

EndStyle
```

Figure 8. GRAPHICAL STYLE SHEETS for the two Trapezoidal Rule illustrations in Figure 3 demonstrate the style language and the graphical style attributes. The style on the left produces a typeset book quality illustration and the style on the right produces a colored 35 mm. slide form. Note that the styles differ in the line weights, color selections, and typography parameters.

shadows and offset shadows, shown in Figure 9. A drop shadow appears to give an object depth by extending a slanted shadow line away from the object. An offset shadow gives emphasis by showing an underlying copy of the object a short distance away. The style parameters for shadow effects are the following:

`shadowType`	the shadow effect: `drop` or `offset`
`shadowPathType`	the offset shadow path type: `filled`, `outlined`, `filled+outlined`
`shadowAngle`	the angle of the shadow from the object, in degrees
`shadowOffsetAmount`	the distance that the offset shadow is placed at `shadowAngle`
`shadowDirection`	the direction of the shadow from the object: `upLeft`, `upRight`, `downLeft`, `downRight`
`shadowWeight`	the weight of the drop shadow or outline of the offset shadow
`shadowAreaColor`	the color of the drop shadow or offset shadow area
`shadowOutlineColor`	the color of the offset shadow outline

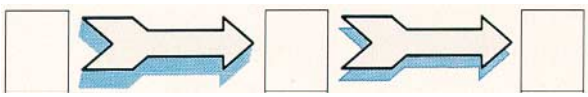

Figure 9. SHADOW EFFECTS can be rendered by a simple algorithm that reuses the geometric path definition several times. The arrow on the left has a drop shadow and the arrow on the right has an offset shadow. Several style parameters are available to control the weight, angle, color, and type of shadow.

Both shadowing techniques can be rendered by first drawing the shadow of the object and then drawing the intended object. Slightly darker shadow colors can be computed in a style rule, for instance, `shadowAreaColor` might be computed as `50 percent` of the brightness of the `areaColor`.

Arrow Styles

Arrows in drawings can be described as paths having a particular arrow style. The style must describe the shape of the arrowhead and tail. One approach is to define a prototype shape on a simple rectangular grid. This model resembles a transfer sheet system such as the one provided by Letraset [1]. To avoid cataloging a large variety of curved arrows, a mapping is used to stretch the prototype shape along any given path. The x-axis of the grid is mapped onto the path directly, and y-values are mapped into distances normal to the path [20]. This scheme permits considerable scope for creative designs by using a simple and easy-to-draw prototype, and then computing the hard-to-draw finished design as in Figure 10.

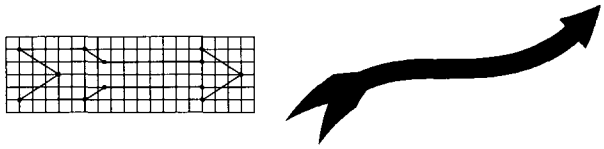

Figure 10. ARROW STYLES can be created by mapping a prototype design, developed on a rectangular grid, along a curved path. The mapping algorithm can be controlled to preserve the arrow head and feather shapes and only stretch the shaft of the arrow. The path can be any combination of lines or curves. The style of the mapped arrow can be filled or outlined in a similar way to other TiogaArtwork objects.

Border Patterns

Arbitrary patterns are a logical extension of the style for arrow heads. These border patterns are slightly more complicated to render because the prototype pattern must be repeated along the path like a wallpaper design. As only an integral number of repetitions is desired, the mapping algorithm must adjust scale factors to ensure an exact number of cycles. Again, the same simple prototype grid is mapped along the path, as shown in Figure 11.

Figure 11. BORDER PATTERNS can also be mapped along paths using the same technique as for the arrows, but using an integral number of repetitions. This Greek Key design consists of a white rectangular area and a blue spiral. The design was created to fit together end to end. The path is a cubic curve.

Conclusions

We conclude that a representation that explicitly specifies the stylistic properties of an illustration provides a powerful way to control document quality. Graphical style sheets can provide design discipline. Using a common style when designing a set of illustrations for a paper, for example, can guarantee a uniform specification of such parameters as color, line styles, and arrowhead shapes. Another benefit is that the additional semantic structure introduced by the style sheet makes it possible to produce those figures over a range of circumstances. Different style sheets can be designed to accommodate not only different publishing styles but also the restrictions inherent in different media, such as the difference between paper and 35 mm slide media. If it is easier to reuse figures, then perhaps more time will be spent producing good ones.

We have successfully implemented a prototype system for manipulating graphical styles which was built to coordinate with the Tioga document preparation system. We have used the combination of Tioga and TiogaArtwork to produce this paper. We conclude from working with this system that it is possible to control some of the stylistic aspects of illustrations using a set of style rules, and that this approach is a useful one for specifying pictures in documents. The particular representation we used was a convenient one for our environment. The important aspect is the separation of the geometry and the style properties, especially the level of indirection introduced by using named style rules. The flexibility inherent in the textual representation and the style language was important in an experimental system. We suspect that a style language is the correct level of abstraction for graphical styles even in fully developed systems.

Representing illustrations as styles plus geometry introduces some interesting issues with respect to rendering algorithms. For example, one natural specification for line style is to specify a uniform width. The specification should also describe the behavior of the line at joints and endpoints; for example, whether the corners are mitered or rounded off. An algorithm that produces an analytical description for the shape of an outline specified in this manner is nontrivial, especially if the path contains parametric cubic splines. This kind of example forces us as graphics system designers to pay attention to useful rendering algorithms, not just convenient ones.

Future Work

It is clear that for many applications one should use an interactive graphics system to design an illustration. For example, we used the Griffin system to generate the illustrations for our paper rather than manually calculate path descriptions. There are many interesting questions about how graphical style should be specified in such a system. Griffin assigns attributes to shapes, but there is little semantic content to the assignment; all objects with the same set of properties have the same style.

In general, we need to learn more about organizing style rules. Our current implementation has a different rule for each node that looks at all different. Often, however, the styles vary in only a few parameters. In fact, the actual semantics may really be: these nodes are the same except for these parameters. In the current TiogaArtwork system,

it is possible to define *generic* style rules, that can be referenced in specific style rules to specify the common set of attributes. Tools are needed to provide an effective interface to this style machinery.

Illustrations are often produced by executing other graphics programs. Frequently these programs have little facility for providing graphical style. Many times the illustrations produced must be redrawn or the programs fine-tuned to generate publication-quality results. Some mechanism for capturing the images produced and for supplying graphical style semantics would be most helpful in incorporating such illustrations in documents.

Another interesting topic is style guidelines which apply to the layout and design of illustrations rather than to simple rendering parameters. It can be convenient, for example, to express box dimensions as a function of the size of the text in the box. The size of the text depends on its font style, so there needs to be some way of specifying this relationship between the style and the geometry. In another example, the actual endpoint of an arrow changes when it is pointing to something which is rendered with a thick line than a thin one. Style rules might also include document layout parameters. For example, the style of an illustration could control the layout so that the figure might have a horizontal orientation when the document is typeset with wide columns, a vertical one when narrow columns are used, or a fixed aspect ratio to suit videotape or slides. A dynamic reconfiguration capability would be helpful in making the illustrations look better in the chosen layout.

Acknowledgements

The sensitivity to graphic arts quality and design issues in this project is due in part to George Roth, a graphic designer who collaborated on a variety of typesetting projects, including the *Computing* text book [6]. With enthusiasm and insight, he often asked "Why can't you program it to do this?" while offering suggestions as to the appearance of the final result.

Both the design and implementation of Tioga document preparation system are critical to our work. Bill Paxton, Michael Plass and Scott McGregor are responsible for Tioga, and we would like to publicly thank them for their contributions to the graphical style project. We also thank John Warnock and Doug Wyatt for the Cedar graphics package and other graphical tools used in our prototype.

One of the principal features of Cedar is that it is an integrated environment. In other words, it is very easy to use pieces of the existing system in a new application. Our work uses this feature extensively, so we were able to build our prototype system in rather short time. Therefore, we will simply thank all of the Cedar implementors for their contributions.

References

[1] anon., *Letraset Catalog*, Letraset USA. (1980).

[2] anon., *Graphic Kernel System (GKS) Functional Description*, ISO TC97/SC5/WG2 N117 (1982).

[3] Baudelaire, Patrick and Stone, Maureen, Techniques for Interactive Raster Graphics, *Computer Graphics* **14**, 3, (1980).

[4] Bell, Edward, Personal communication regarding the production of illustrations within *Scientific American* (1982).

[5] Chamberlin, D., King, J., Slutz, D., Todd, S. and Wade, B., JANUS: An interactive System for Document Composition. *IBM Systems Journal* **21**, 3 (1982).

[6] Dyck, V.A., Lawson, J.D., Smith, J.A. and Beach, R.J., *Computing -- An Introduction to Structured Problem Solving Using PASCAL*, Reston (1982).

[7] Kernighan, Brian W., PIC - A Language for Typesetting Graphics, *Software Practice & Experience* **12**, 1 (Jan. 1982).

[8] Knuth, Donald E., *TEX and METAFONT, New Directions in Typesetting*, Digital Press (1979).

[9] Lampson, Butler W., Pier, Kenneth A., McDaniel, Gene A., Ornstein, Severo M. and Clark, Douglas W., *The Dorado: A High-Performance Personal Computer, Three Papers*, CSL-81-1, Xerox PARC (January 1981).

[10] Lipke, Daniel, Evans, Steven R., Newlin, John K., Weissman, Robert L., Star Graphics: An Object-Oriented Implementation, *Computer Graphics* **16**, 3 (1982).

[11] Lesk, Mike E., Typesetting Documents on UNIX and GCOS: Using the -ms Macros with Troff and Nroff, *Unix Programmer's Manual* **2A**, 7th ed., Bell Laboratories (1979).

[12] Meyerowitz, Norman and van Dam, Andries, Interactive Editing Systems, Parts I and II, *ACM Computing Surveys*, **14**, 3 (1982).

[13] Mitchell, James G., Maybury, William and Sweet, Richard, *Mesa Language Manual*, CSL-79-3, Xerox PARC (April 1979).

[14] Paxton, Bill, *The State of Cedar*, Xerox PARC videotape, V-163 (1982).

[15] Reid, Brian K., A High-Level Approach to Computer Document Formatting, *ACM Symposium on Principles of Programming Languages* (Jan. 1980).

[16] Ruegg, Ruedi and Frohlich, Godi, *Basic Typography*, ABC Verlag Zurich (1978).

[17] Smith, David Canfield, Irby, Charles, Kimball, Ralph and Verplank, Bill, Designing the Star User Interface, *Byte* (Apr. 1982).

[18] Van Wyck, Christopher J., *IDEAL User's Manual*, Computing Science Technical Report No. 103, Bell Laboratories (1982).

[19] Waltz, David, Artificial Intelligence, *Scientific American* **247**, 4 (1982).

[20] Warnock, John, Personal communication regarding the mapping of prototype rectangular grid designs on to curve paths (1982).

[21] Warnock, John and Wyatt, Douglas K., A Device Independent Graphics Imaging Model for Use with Raster Devices, *Computer Graphics* **16**, 3 (1982).