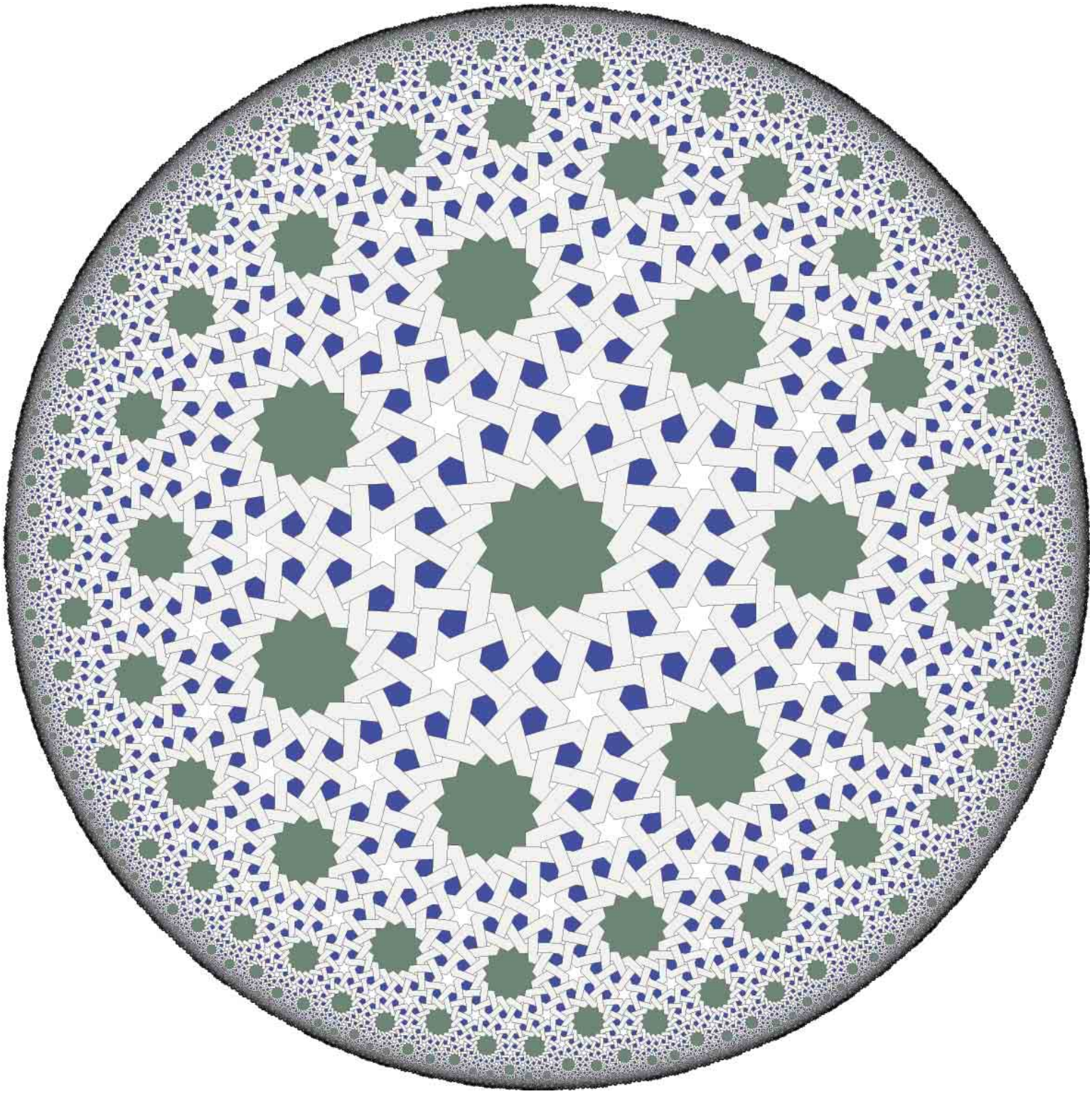


An Islamic star pattern realized in the hyperbolic plane, generated using custom software described by Craig S. Kaplan and David Salesin in "Islamic Star Patterns in Absolute Geometry" in *ACM Transactions on Graphics* (Apr. 2004).

ADAPTIVE DOCUMENT LAYOUT

How to automatically reformat, resize, and paginate electronic text and graphics so documents look as good on displays of any size as they do on paper.

Quality layout and design are hallmarks of paper publications, giving each magazine and newspaper its own unique branding and visual style. You can spot *Time* magazine from 20 feet away because of its distinctive look. One of the most important tools designers use to achieve these effects is an underlying grid to help order the visual elements on each page into a coherent and pleasing display. Grid-based design has its roots in the work of Dutch artist Piet Mondrian and Swiss architect Le Corbusier in the 1920s to 1940s and further development after World War II in Switzerland. After spreading worldwide in the 1950s and 1960s, it has since become the standard for paper-based publication design [3, 5]. Today, a number of successful software systems support grid-based page design, including QuarkXPress—the print industry standard—and Adobe PageMaker and Microsoft Publisher, both typically used for desktop publishing.

Online publication design is a different story. During the past few decades, even though computers have become much more powerful, display technology approaches the readability and convenience of paper, and the profusion of electronic networks makes distribution of electronic documents quick and cheap, the design of online publications has lagged. When presented electronically, grid-based page design is typically in a "frozen" form like Adobe's Portable Document Format (PDF). This inflexibility can lead to a poor online reading experience, since the

By Charles Jacobs, Wil Li, Evan Schrier, David Bargeron, and David Salesin

size and resolution of many computer monitors, laptop screens, and PDAs require users to scroll around in order to see a whole page. Meanwhile, articles presented in HTML usually provide some limited adaptability that, unfortunately, comes at the expense of a quality grid-based layout. Although it is certainly possible to produce grid-based page design using HTML, Web-page designers are more likely to use fixed-size HTML tables that do not adapt to the viewer's display.

Two main reasons explain why online publication design is so poor. First, traditional design techniques—even those employing software and computer hardware—are highly manual and do not scale well. Quality layout and design is simply too expensive for the enormous amount of content available on the Internet, especially at the dizzying speed it is being distributed and updated. Second, advances in display technology have tended to fracture the market into a variety of viewing devices, each with its own characteristic sizes and resolutions. Creating separate layouts for each different form factor would only compound the complexity and expense of quality publication design.

As a result, many of the visual cues that help brand paper publications are woefully absent in their online editions, and the online readability of these publications suffers for it. To bring online publication design up to par with its paper-based counterpart, we propose using grid-based design principles that automatically adapt content to appealing page layouts and that match the displays on which they will appear.

THE CENTRAL IDEA IS TO DEFINE A LAYOUT "STYLE" AS A SET OF ADAPTIVE PAGE LAYOUTS INDEPENDENT OF ANY PARTICULAR CONTENT, THEN RELY ON THE COMPUTER TO LAY OUT DOCUMENTS USING THESE DESIGNS.

Defining Layout Style

To address the shortcomings of the current online reading experience, we've developed an adaptive document layout system to display documents on a variety of devices while retaining the ability to produce quality grid-based layouts. Our solution leverages and extends previous research on adaptive page layout, including constraint-based layout [2] and style-content separation [1]. The central idea is to define a layout "style" as a set of adaptive page layouts independent of any particular content, then rely on the computer to lay out documents using these designs. The layouts themselves can adapt to a range of display sizes, and different layouts can be created to handle displays that vary even more widely in size and shape. Because the system separates style and layout from content, the same layouts can be reused repeatedly for new content.

The system represents page layouts through adaptive page templates that provide a general framework for the layout of the elements on the page while allowing arbitrarily complex arrangements. The locations of the elements are described in terms of a set of constraints, enabling them to adapt to a variety of different-size pages. Each template is designed to adapt to a range of display dimensions, as well as to other types of viewing conditions (such as degrees of font magnification). The document's content is then selected and formatted dynamically to fit the viewing situation at hand, as well as user preferences and the display device being used. The pagination and layout system chooses the best sequence of page templates to use, given a particular document and viewing situation, then presents the resulting layout on the screen. Using simple "first fit" pagination, the system is fast enough to support real-time interactive window resizing on desktop computers [4].

Each template is responsible for defining the grid-based layout for a single page containing a particular set of content, including, say, two figures, a sidebar, and some body text. Each template is valid across a range of page dimensions, and a collection of these templates together constitutes a particular layout style. The document content itself is divided into a set of logically independent streams that are laid out sequentially. For example, an article might have a "body text" stream containing the main text of the article and a "figure" stream containing the illustrations and photographs accompanying the text. The system also supports a range of modern page-design ideas (see Figure 1).

Each template divides a page into a set of regions, or elements, where content is to be placed. To specify which content to use, each element specifies a source stream from which its content is drawn. If multiple elements consume content from the same stream, then the stream's content flows from one element to the next. For example, a layout with two columns of text would have an element for each column, as read from the body-text stream in sequence.

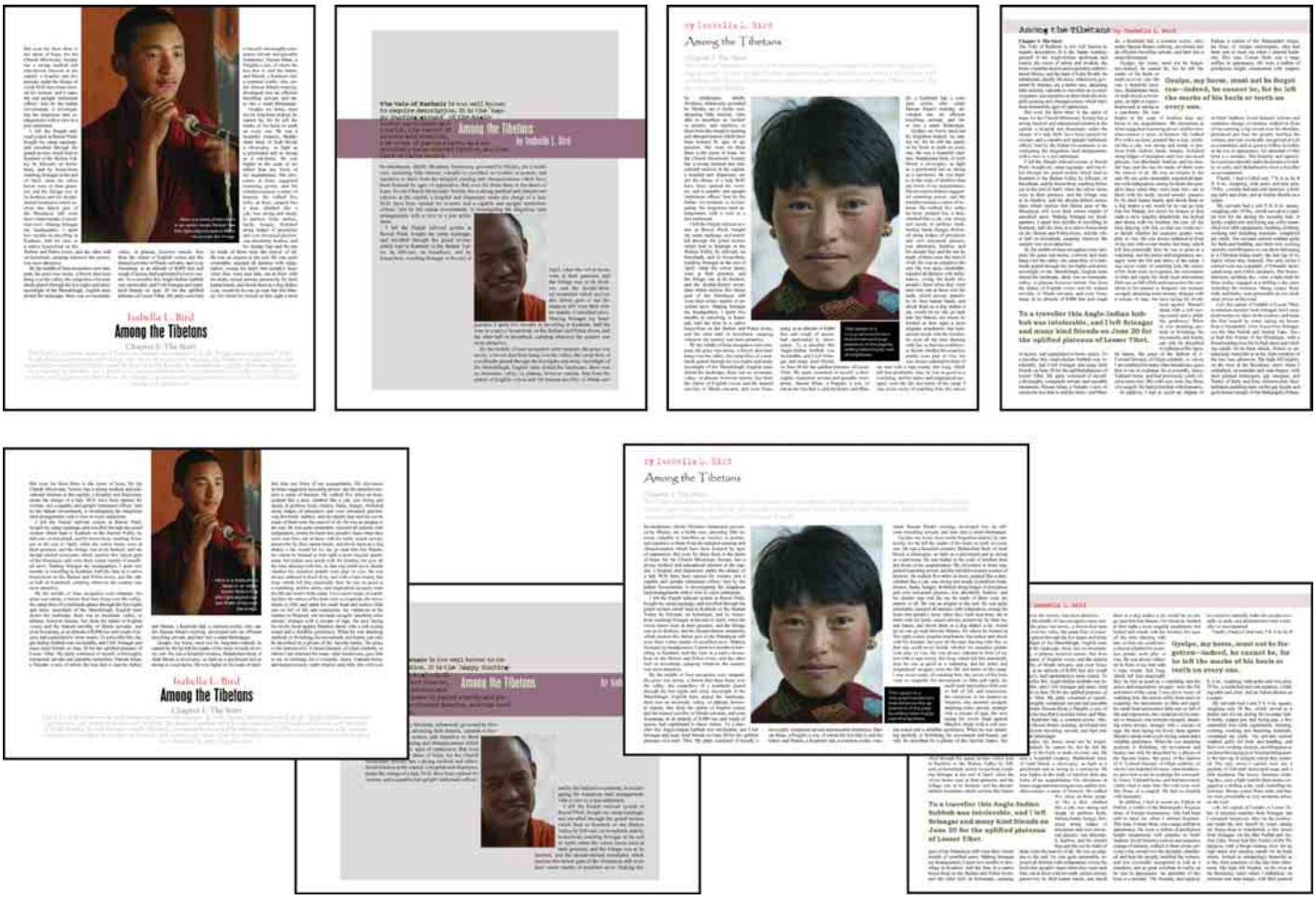


Figure 1. Adaptive grid-based designs at two different page sizes. Note how each one automatically adapts to portrait and landscape orientations by changing the configuration of layout elements.

Grid-based page designs often include overlapping elements, or regions that appear to be cut out of other elements. For example, a common type of page element is a "runaround" figure, where the text flows around the figure. The system achieves this effect by allowing each element to specify its place in an element z-order. Elements higher in the z-order sit atop the lower elements, and the area of the higher elements is "cut out" of the lower elements.

Several individual pieces of content sometimes combine to form a single conceptual unit. For example, a photographic image and its associated caption are typically laid out together as a single figure. The system handles these collections of content by allowing nestable subtemplates that format one of these compound collections of content as a unit. An element in the page layout specifies a subtemplate to lay out these pieces of content within the space reserved for the figure. The layout infrastructure is thus fully recursive, supporting everything from simple figure/caption combinations to more complex embeddings.

The system adapts page layouts to the attributes and characteristics of the target display and the content at hand by dynamically determining the size and placement of the elements in a template. It does this by evaluating the interdependent constraints among the elements of the template. Consider, for example, a simple page template consisting of a title element above a body-text element. The title is constrained to begin at the top of the page and to be as tall as required in order to fit the title text. The body element is constrained to begin at the bottom of the title element and end at the bottom of the page. How tall the title element actually turns out to be and how much space is available for body text depend on the content that is to be laid out, along with the page size.

The constraint system comprises a pool of con-

Figure 2. This document is adaptive over a range of page sizes. Note that for a certain range of sizes, the same basic layout (expressed in a single page template) is reused, though once the page size changes too dramatically, the system automatically switches to a new template.

straint variables, whose values are determined by a mathematical expression in terms of the other constraint variables, and is known as a one-way constraint system. Information about the display context (such as the physical dimensions of the page) is reflected in read-only constraint variables. The table on the right-hand page shows a schematic view of a sample template and the constraints that determine the extents of each of the elements on the page. By expressing the locations and extents of the page elements, the system resizes the template to fit the given content at any given display size. However, if the page size changes too dramatically, a single layout may not adapt appropriately. Instead, the system may need to switch to a new layout (such as one defined by a different template). Figure 1 shows how a variety of page layouts might adapt to a wide-aspect page.

In order for the system to be able to choose from the multitude of templates in a given layout style, each template must include preconditions to express when it is valid. Preconditions might indicate the amount of content from a given stream that must be present, as well as the range of values within which a given constraint variable must lie. With this mechanism the page designer might specify that, for example, a particular template is valid only if there are exactly two figures available to put on the page, and only if the page dimensions fall somewhere between a European A4- and a U.S.-letter-size page. The system relies on preconditions to determine the range of page sizes for which each template is valid and to help determine which set of templates to use for laying out a document's first page, as first pages often require special treatment. Figure 2 illustrates how a single template might adapt to a range of page sizes but switch to a different layout when the page is too wide.

BECAUSE THE SYSTEM SEPARATES STYLE AND LAYOUT FROM CONTENT, THE SAME LAYOUTS CAN BE REUSED REPEATEDLY FOR NEW CONTENT.

Two or more distinct templates might be able to accommodate similar content and meet the same preconditions. Each template therefore also includes a scoring variable, the value of which is determined by evaluating the template's constraints against the content to be laid out. The scoring variable influences the choice of template when paginating a whole document. For example, two different templates may be able to accommodate content that includes a single figure, but one of the templates may be more appropriate for a larger figure, while the other may be more appropriate for a smaller figure. If the system is trying to lay out a large figure, the first template will produce a higher score than the second template, and the system will choose it to lay out the content.

Desired Pagination

Any document that is to be presented to readers must first be paginated. That is, the system must decide which template to use for each page and how much content to put into each template. In order to produce a desirable pagination, the system's paginator must account for several factors, including: aes-

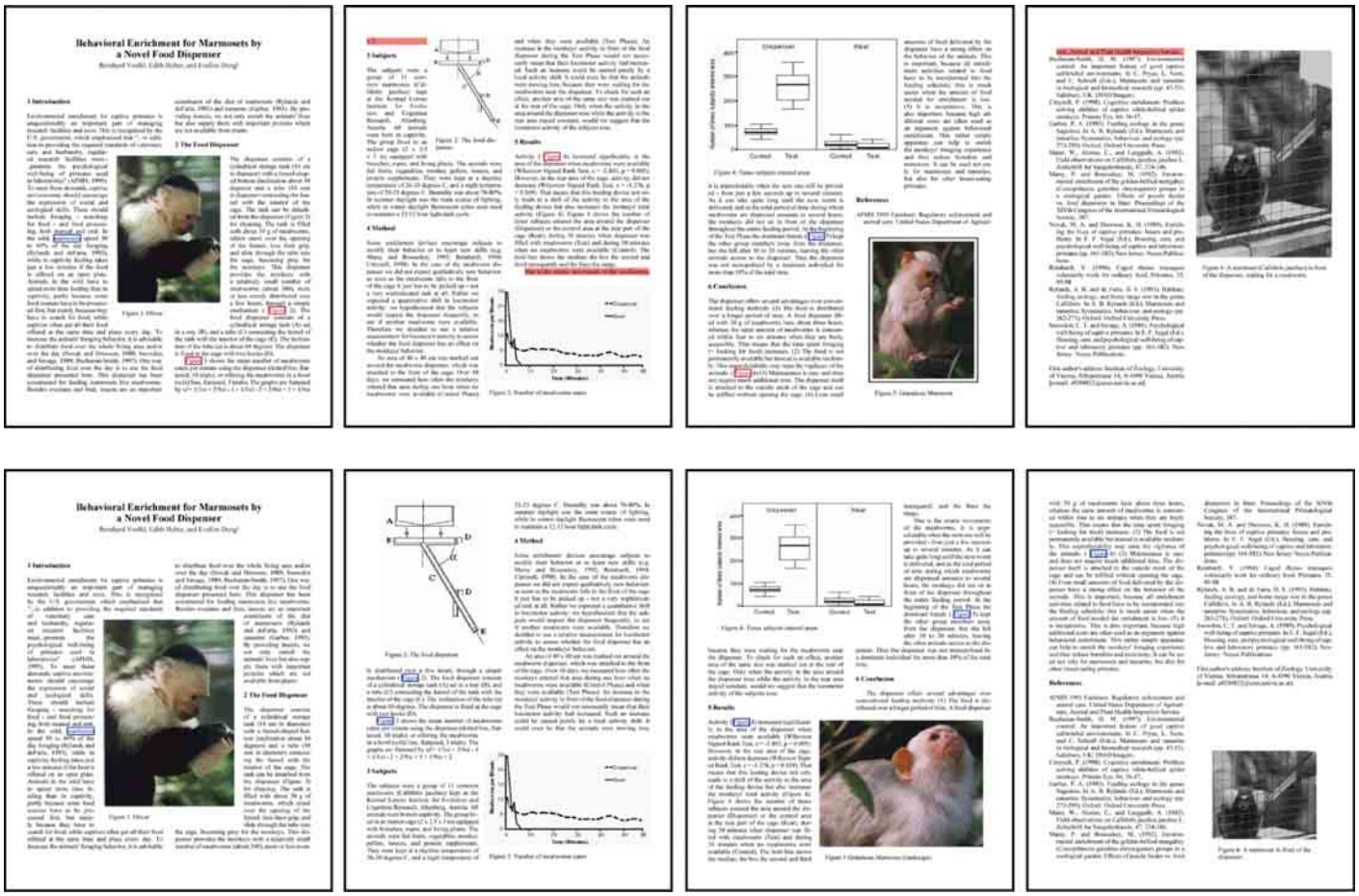
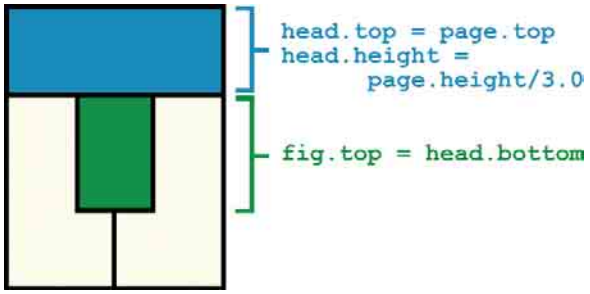


Figure 3. Top row is a "first fit" pagination. Undesirable layout choices (such as figure references not placed on the same page as the referenced figure), as well as widowed and orphaned lines of text, are highlighted in red. References on the same page as their referents are outlined in blue. The bottom row shows how the pagination algorithm improves the layout by adding more templates and versions of images.

thetic and typographic measures (such as the presence of widowed and orphaned lines of text); proximity of figures to the places in the text that reference them; and the fullness of the document's pages. The page templates themselves may also influence pagination through evaluation of their scoring variables. These factors all go into a quality score for a particular pagination. The paginator tries to find a sequence of pages that produces the highest score. This automatic procedure allows publishers to reuse a given layout style for many different documents. It also allows the reformatting of documents for different display sizes without additional design labor and without modifying the document source.



A page template and the constraints used to size its constituent elements.

When a document includes only text, superior pagination can avoid widowed and orphaned lines by slightly under- or over-filling pages and moving page breaks to more desirable locations. When one or more additional types of content (such as figures and tables) are present, the paginator may choose from a variety of page assignments that may change the positions of page breaks in the text. Multiple template choices may also be available for the same content assortment. These options mean that finding a desirable pagination is non-trivial.

To achieve an optimal pagination, a measure of "badness" must be defined to reflect the various properties described earlier to calculate a numeric score. In general, the more flexibility available to the pagination algorithm, the better the optimal solution is likely to be. Our pagination

WE CAN EXPECT EVEN GREATER PROLIFERATION OF DIFFERENT-SIZE READING DEVICES, ALONG WITH THE NEED FOR DOCUMENTS THAT ADAPT TO THEM.

algorithm is a variation on a dynamic programming-based approach [6] that builds up a solution to a problem by combining the results of more limited subproblems. First, it computes all solutions for the first page, then computes the solutions for the second page using the previously computed first pages as starting points. We have enhanced the efficiency of this solution by pruning the number of subproblems the system must evaluate to find the optimal solution. One such pruning strategy uses a fast, approximate method to derive an upper bound on the cost of the optimal solution. If a subproblem's cost exceeds this bound, the algorithm builds no further solutions on that subproblem. Another pruning strategy starts with a near-perfect threshold and iteratively relaxes that threshold until it finds a solution.

The system might also introduce optional or "filler" content by employing templates that consume content from optional streams. If these templates are included in the layout style and if optional content is available in the document, the paginator will include them in the solution whenever they improve the optimal pagination. We have found that having a small number of templates with optional content vastly improves pagination quality (see Figure 3).

Although constraints are a powerful means for specifying flexible layouts, increased expressiveness also places more responsibility on the template designer, who must now consider how a page must look over a range of aspect ratios and sizes, rather than at just a single static size. To make this potentially difficult task more manageable, we have developed a constraint-based graphical authoring tool that allows the template designer to draw and arrange layout elements, specify how they adapt to different page sizes, and preview this adaptation interactively. The system also supports the authoring of template preconditions and adding style attributes to elements.

Conclusion

Computer display hardware is approaching the point—at least from an image-quality standpoint—where the on-screen reading experience rivals the one on the printed page. Moreover, there is every reason to believe that once documents look as good on a screen as they do in print, the on-screen reading experience will surpass that of reading on paper. Computers provide many opportunities for customization of style and content—as well as for navigation, animation, and interactivity—that are bound to make the on-screen experience superior. It is also reasonable to assume that as high-resolution displays become cheaper and require less power, we will see even greater proliferation of different-size reading devices, along with the need for documents that adapt to them.

Oddly, one of the most significant impediments to achieving the decades-old vision of a paperless world may turn out to be a deceptively simple 2D computer graphics and user interface problem. How might grid-based design adapt elegantly and seamlessly to all viewing conditions? We've taken some significant first steps toward an answer, though our system is not the final word on adaptive grid-based page layout. We feel it does, however, open up a lot of new and interesting directions and hope others will join us in exploring them. C

References

1. Adler, S. *Extensible Stylesheet Language (XSL), Version 1.0.* W3C Recommendation, 2001; see www.w3.org/TR/xsl/.
2. Borning, A., Lin, R., and Marriott, K. Constraint-based document layout for the Web. *Multimedia Syst. 8,* 3 (Oct. 2000), 177–189.
3. Hurlburt, A. *The Grid.* Van Nostrand Reinhold Co., New York, 1978.
4. Jacobs, C., Li, W., Schrier, E., Bargeron, D., and Salesin, D. Adaptive grid-based document layout. *ACM Transact. Graph. 22,* 3 (July 2003), 838–847.
5. Müller-Brockmann, J. *Grid Systems in Graphic Design.* Hastings House Publishers, New York, 1981.
6. Plass, M. *Optimal Pagination Techniques for Automatic Typesetting Systems,* Tech. Rep. STAN-CS-81-870, Department of Computer Science, Stanford University, 1981.

Charles Jacobs (cjacobs@microsoft.com) is a research software design engineer in the Document Processing and Understanding Group at Microsoft Research in Redmond, WA.
Wil Li (wilmotli@cs.washington.edu) is a Ph.D. candidate in the Graphics and Imaging Lab at the University of Washington in Seattle.
Evan Schrier (evans@cs.washington.edu) is a Ph.D. candidate in the Graphics and Imaging Lab at the University of Washington in Seattle.
David Bargeron (davemb@microsoft.com) is a research software design engineer in the Document Processing and Understanding Group at Microsoft Research in Redmond, WA, and a Ph.D. candidate in the Graphics and Imaging Lab at the University of Washington in Seattle.
David Salesin (salesin@cs.washington.edu) is a professor in the Department of Computer Science and Engineering at the University of Washington in Seattle and a senior researcher in the Document Processing and Understanding Group at Microsoft Research in Redmond, WA.