Pixels or Perish

Brian Hayes

Drawings and pictures are more than mere ornaments in scientific discourse. Blackboard sketches, geological maps, diagrams of molecular structure, astronomical photographs, MRI images, the many varieties of statistical charts and graphs: These pictorial devices are indispensable tools for presenting evidence, for explaining a theory, for telling a story. And, on top of all that, they *are* ornaments; they entice and intrigue and sometimes delight. A magazine like *American Scientist* would be impoverished without them.

Methods for producing scientific illustrations—and for *re*producing them in publications—have been changing. Printing plates for figures were once engraved by hand, then made by a photographic process, and in recent years have been created by digital techniques. Now we are about to turn the page—if not close the book—on yet another chapter in publishing history. After centuries of reading and writing on paper, we seem to be headed for a world where most documents will be distributed online and viewed on a display screen of some kind. How will this transition to a new medium affect the practice of scientific illustration?

Print publishing has a centuries-long tradition and a rich culture. Generations of illustrators have developed technical knowledge, artistic sensibility and a highly refined toolkit. There's a huge body of existing work to serve as example and inspiration. In digital publishing, this kind of intellectual infrastructure is only beginning to emerge

Yet the new computational media offer new opportunities for the exercise of creativity, especially in quantitative graphics, where illustrations

The art of scientific illustration will have to adapt to the new age of online publishing

are closely tied to data or mathematical functions. On the computer screen, graphs and diagrams can become animated or interactive, inviting the reader or viewer to become an explorer. I find this prospect exciting. But I'm also mindful that we don't yet have deep experience with the new graphical methods.

Pyramid Scheme

I offer the illustration on the opposite page—along with the corresponding digital version on the *American Scientist* website—as a case study. Population pyramids are a well-established tool in demography. In this case the pyramids show the age structure of the global human population over a 150-year period, according to estimates and projections published by the United Nations.

Tracing change over time is the main point of the illustration, yet this is notoriously hard to do in a static picture. Snapshots at 50-year intervals give some sense of the overall outcome: What begins as a pyramid evolves into an onion dome. But it's not so easy to see how and why the shape is changing. One thing that's not made explicit is how cohorts (groups of people born at about the same time) move upward through the age categories as time passes. Consider the bar at the base of the pyramid in 1950, which measures the number of people who were less than 5 years old in that year. The survivors of this group reappear in the 50-to-54-year-old bar in 2000, and a tiny sliver of centenarians remain in 2050. There's nothing in the structure of the diagram to remind you that those three bars represent the same people.

No doubt a clever illustrator could improve the graphs in ways that would more clearly convey the basic facts of life: that births affect only the bottommost bar, and deaths shape all the rest. Showing more intermediate stages would certainly help. However, space on the page is always at a premium in a printed magazine.

The version of the same illustration in the Web edition of this column suggests some of the possibilities of more-dynamic visual media. Instead of looking at preselected snapshots, you can move through time, forward or backward, and watch the pyramid change shape as a result of births and deaths. Animated transitions emphasize the continuity of the human population, as cohorts migrate through the decades. With higher temporal resolution (5 years per step, rather than 50), it's easier to spot noteworthy moments of transition. For example, it appears there was a sharp drop in worldwide fertility in about 1990; that's when the sides of the pyramid grow noticeably steeper. And another landmark comes in about 2050, when each successive group of 0-to-4-year-olds ceases to be larger than the preceding cohort, so that the base of the "pyramid" becomes pinched. (Note: I am deeply interested in these demographic trends, but my aim here is to discuss the effectiveness of graphic presentations, not to debate the meaning or validity of the data.)

Gains and Losses

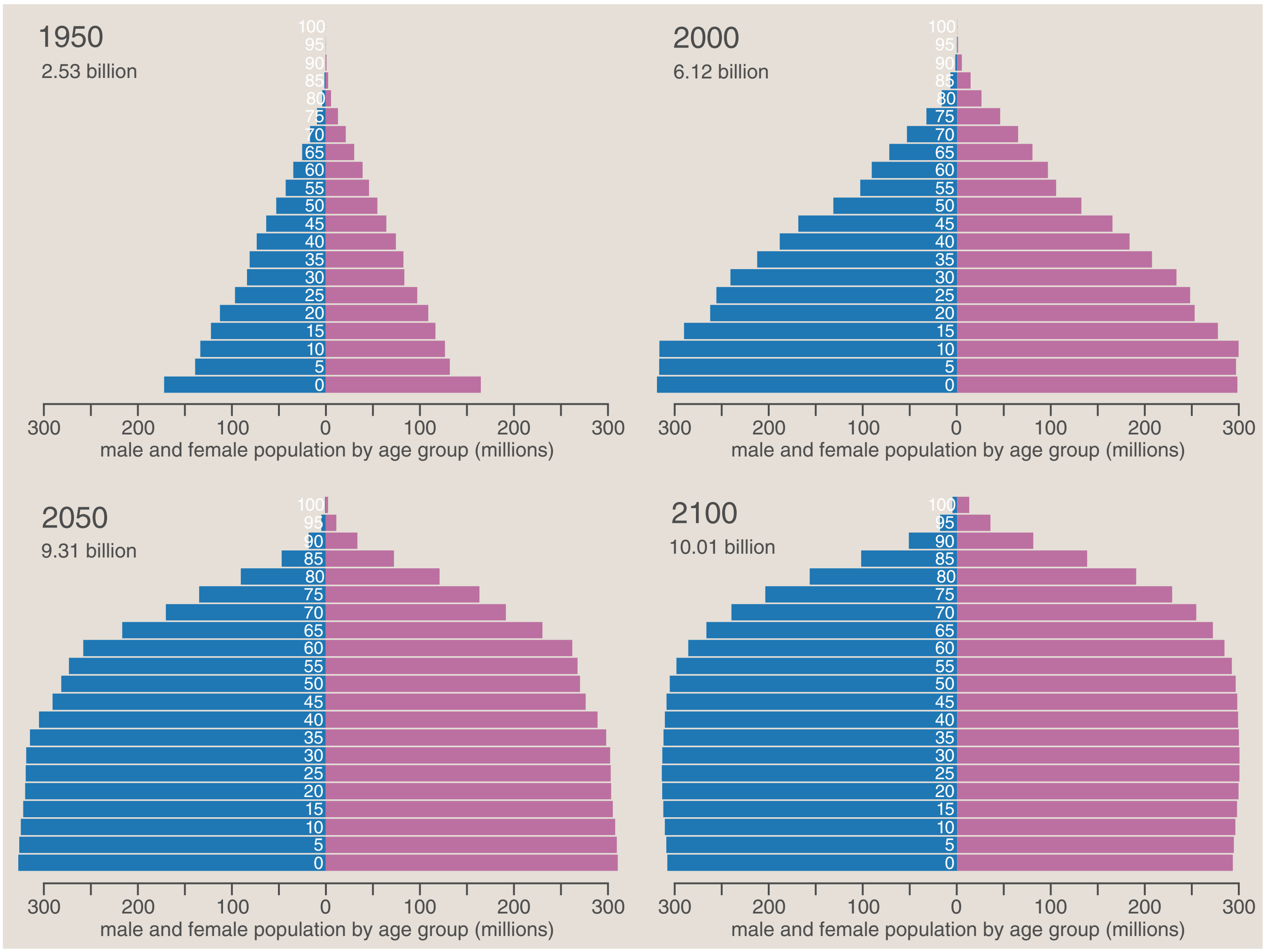
Interactive gadgets like the Web version of the population pyramid tend to be put in a category apart from the illustrations that appear on the pag-

Brian Hayes is senior writer for American Scientist. *Additional material related to the* Computing Science *column appears at http://bit-player.org. Address: 11 Chandler St. #2, Somerville, MA 02144. E-mail: brian@bit-player.org*

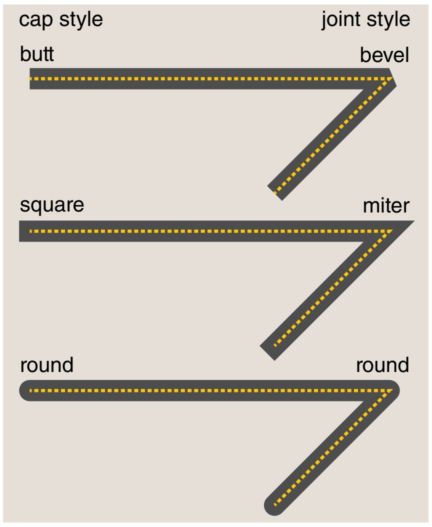
es of a magazine or journal. They are classified as supplemental material, or maybe educational software, and are not seen as an integral part of the publication itself. Years ago, many publishers segregated photographs and certain other kinds of illustrations in an analogous way. They were printed on special paper and bound in a separate section of "plates." That practice ended with improvements in printing technology. Likewise, when publications are distributed over the network and read on a computer screen, active graphics can be integrated into a document in the same way that ordinary photographs and drawings are. There's no reason to keep them out of the mainstream.

What do we stand to gain in going from paper to pixels? Animation—adding a time axis to a graphic—is the most obvious possibility, but there are many other ways to exploit the power of computation. For one thing, we are liberated from the fixed size of the printed page. Computer displays also have bounds, but when a figure is too large to fit, we can roam about in it by scrolling or by "panning and zooming." (Think of Google Maps.) When a diagram is too intricate for the reader to see all details clearly, we can offer tools to magnify selected regions. In a cluttered graph, we can highlight and label data points when the reader selects them, or else hide distracting features from view. We can offer the reader options, such as changing the scales of a graph from linear to logarithmic, or choosing a subset of the data. Three-dimensional graphics are easier to understand in a medium where the reader can rotate a diagram or change the point of view.

Of course good old-fashioned paper also has advantages, starting with the fact that everyone knows how to use it. No one needs an instruction manual for reading a magazine. No one needs any special hardware or software, either. Authors and publishers can be reasonably certain that all readers will see the same words and pictures; there's no need to worry that Internet Explorer will sow one thing and Firefox another. And the printed



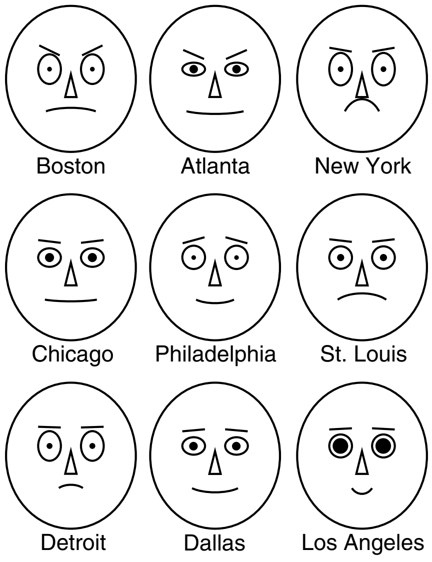
Evolution of the human population provides an example of a concept that seems difficult to convey effectively with a static image on the printed page; an animated, interactive graphic may be better suited to the task. The four population pyramids show the age structure of the world population at 50-year intervals, according to estimates and projections made by the United Nations. From such snapshots it can be difficult to perceive how groups of people flow through the age categories, determining the shape of the distribution from year to year. For example, the 0-to-4-year-olds in the bottom bar of the 1950 pyramid went on to become the 50-to-54-year-olds in 2000 (though slightly diminished by deaths). An interactive version of the illustration on the *American Scientist* website allows the trajectory of individual cohorts to be followed more clearly. Animated transitions show each group rising through the pyramid over the course of a lifetime, while a new generation enters at the base. The interactive version is built with SVG (Scalable Vector Graphics) and a JavaScript library called D^3, created by Michael Bostock.



Joinery that master carpenter Norm Abrams would approve is a feature of the PostScript graphics language. Where a stroked path makes a turn, the joint can be beveled, mitered or rounded. Another setting controls the style of the end caps. The same fastidious attention to small details appears in the more recent language called Scalable Vector Graphics.

page still offers a level of resolution and typographic refinement that cannot be matched on the electronic display screen.

At a deeper level, the producers and consumers of printed graphics have had many decades to develop conventions about various graphic devices and what they mean. For example, arrows are variously used to show the flow of material or time or interconnections between parts. Line graphs and bar charts have an elaborate semantics that is not obvious but is widely understood. Much of this knowledge and lore will transfer directly to new computational media, but we'll doubtless also need some new graphic metaphors, and it may take time for them to emerge.



Cartoon faces whose features encode measured aspects of life in several cities were generated by a PostScript program, which both computed the mapping from data to facial geometry and produced the graphical output. Eyebrow angle corresponds to walking speed, pupil size to the speed of bank transactions. The original illustration appeared in "The Pace of Life," by Robert V. Levine, *American Scientist*, September–October 1990.

Picture Perfect

Scientists make pictures for many purposes. Doodles and sketches in a lab notebook might serve a strictly private function; many graphs and charts are created in a process of exploratory data analysis, and are soon discarded. Here I want to focus on more formal illustrations—those destined for publication, perhaps in a journal or an *American Scientist* article, perhaps in a textbook or on an educational website. And because my interests are computational, I'm going to emphasize quantitative graphics.

Any account of publication-quality computer graphics has to begin with PostScript, the "page description language" developed in the 1970s and 1980s by John Warnock and Charles Geschke, the founders of Adobe Systems. Warnock and Geschke are computer scientists, but they worked closely with graphic artists, typographers and the printing trade, and the language reflects this influence.

PostScript is primarily a language for "vector" graphics, where objects are constructed from geometric lines and curves, rather than "raster" graphics, where an image is a rectangular array of discrete pixels. PostScript operators with names such as *moveto*, *lineto* and *curveto* construct a path in a two-dimensional coordinate system of almost unlimited precision, so that the geometry of the drawing is independent of the resolution of the output device. Paths can be built from straight line segments or from curves called Bézier splines, defined by cubic equations. The operators *stroke* and *fill* can then be applied to create a visible graphic object. Some aspects of the language seem almost comically fastidious, such as the elaborate specifications of beveled, mitered or rounded joints between stroked lines; but it turns out such fussiness makes a real contribution to the visual quality of the finished artwork.

PostScript has another distinctive property: It is not just a notation for describing drawings but a complete programming language, with features such as conditional expressions, iteration and named procedures. In this way PostScript blurs the distinction between drawing a picture and writing a program.

An illustration published in *American Scientist* in 1990 offers an example. Robert V. Levine of California State University, Fresno, had written an article on "The Pace of Life," measuring quantities such as walking and talking speed in 36 cities. As an aid to understanding this multivariable data, I experimented with a visualization technique invented by Herman Chernoff of Stanford University. The illustration mapped Levine's measurements to various features of a cartoon face. (Part of the array of faces is shown below at left.) The PostScript file that generated this figure did not specify the coordinates of the various lines, arcs and ellipses in each of the 36 faces; instead, it had a single face-drawing procedure, which was invoked 36 times on 36 rows of raw data. Thus the illustration didn't exist, even as an internal data structure, until the program was run.

Virtual Paper

In 1990 the only way I could run a PostScript program was to send it to a laser printer or a typesetting machine; I had no way to see output on the computer screen. (The debugging cycle consumed reams of paper.) Today we have PostScript interpreters for the screen, but the language remains closely tied to its ink-on-paper origins and is useless for any kind of active illustration, where objects move or respond to events. In PostScript, all art is still life.

A later variant called Display PostScript was meant to bring the same elegant and precise drawing model to interactive graphics, but it never caught on. What *did* gain traction was PDF, or Portable Document Format, which takes a step in the opposite direction, away from programmatic graphics. PDF is essentially "flattened" PostScript; it's what's left when you remove all the procedures and loops in a program, replacing them with sequences of simple drawing commands.

From the outset, PDF aspired to be virtual paper—to re-create on the computer screen the experience of reading a printed document. It succeeds brilliantly. Layout and typography are carefully preserved; you get everything but paper cuts and inky fingers. This is a laudable achievement, but I also see it as a sad waste of resources. When I read a PDF on my laptop, I'm using a powerful and versatile computing engine to imitate a mere sheet of paper. The machine could do much more.

One remedy for this situation would be to re-engineer PDFs to make fuller use of the available computing capacity. Many of the necessary facilities, such as scripting languages, are already present in the PDF specification; they're just not used much. That could change. In the meantime, though, lively ideas for active graphics and scientific visualization are coming from another direction—from the world of HTML, the language of the Web.

The Web Playground

In some respects the Web is an unlikely source for innovations in high-quality graphics. It began as a text-only service, and when graphics were first introduced—through the <img> element of HTML—the only acceptable formats were raster images. Proposals for including vector graphics in Web pages were discussed all through the 1990s, and standards were drafted soon after. Nevertheless, vector formats have become a convenient and practical option for Web graphics only in the past few years.

In spite of this long struggle to bring drawing to the Web, the medium has attracted a community of talented programmers, designers and artists, who find it a friendly place for experimenting with new ideas and showing off the results. By its nature, the Web is a very open system, where anyone who can view a page can also see the code that created it.

We now have *two* widely supported schemes for drawing on Web pages. (Two is not necessarily better than one.) The <canvas> element of HTML is closely analogous to the <img> tag but accommodates vector graphics. Scalable Vector Graphics, or SVG, introduces an entire sublanguage similar in structure to HTML.

Even before these additions, the Web was already a polyglot nation. Web browsers have to speak at least three languages: HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) and the JavaScript programming language. These are, respectively, the nouns, the adjectives and the verbs of the Web. HTML sets forth the basic structure of a document (paragraphs, headings); CSS provides guidance on how to present the various elements (colors, fonts, margins); JavaScript encodes actions (responding to mouse clicks and other events).

The <canvas> element, as the name suggests, is a blank rectangular surface for drawing on. It has a fixed size and can be placed anywhere in a document. The dimensions are measured in screen pixels, so this is not a device-independent graphics protocol; however, coordinates can be specified with precision finer than the pixel resolution. The JavaScript methods that draw shapes on the canvas include procedures named *moveTo*, *lineTo*, *stroke* and *fill*, with an explicit nod to the PostScript heritage. There are even the same options for line caps and joints.

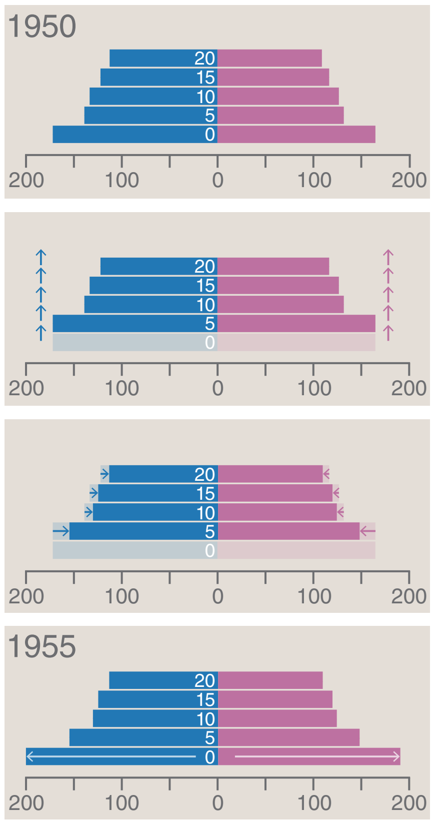
SVG works a little differently. Instead of setting aside a rectangular region that isolates the drawing from other elements of the document, SVG incorporates the drawing commands into the same data structure (called the Document Object Model, or DOM) that holds all the HTML. Indeed, the SVG language is a close cousin of HTML, with a similar syntax based on tags enclosed in angle brackets. And, like HTML, SVG is a noun language; but the nouns are different, defining lines and curves rather than paragraphs, tables and lists.

Whereas drawing on a canvas is done procedurally—by invoking JavaScript functions—SVG drawing elements can be defined in a declarative manner, simply by listing their coordinates and properties, in much the same way that text is entered into an HTML file. This declarative style might suggest that an SVG drawing is a static object, defined in advance and never changing. But it is brought to life by JavaScript, which can inject new drawing elements, remove old ones, rearrange objects or alter their style attributes. JavaScript programs have access to the entire DOM, so operations on drawings use the same basic mechanisms as operations on text.

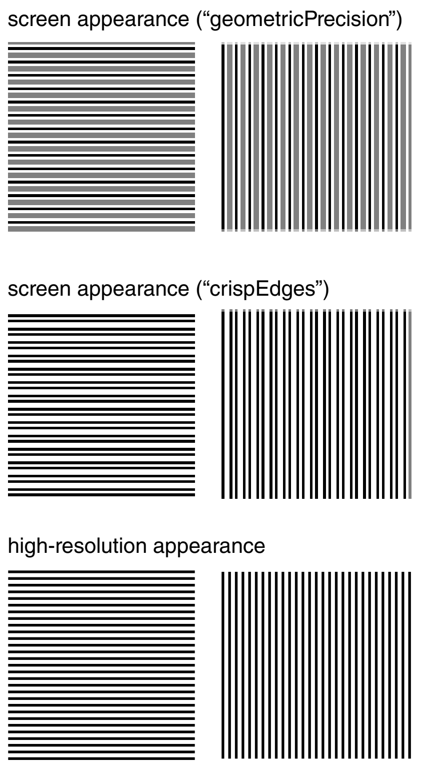
SVG also borrows heavily from PostScript (including the line caps and joints). And in this case the drawing space truly is device-independent and capable of very high precision. Anything displayed on the screen must ultimately be mapped to a finite number of pixels, but SVG drawings take maximum advantage of the available resolution, just as PostScript figures do.

Data-Driven Documents

The interactive version of the population pyramid that I discussed above is an SVG graphic—although, as it happens, I did not write a single line of SVG code when I created the illustration. All of the SVG structures are generated by an embedded JavaScript program, which reads in a data file and constructs the corresponding bars for male and female population. (The architecture is the same as that of the



Comic-strip animation, where successive stages in a process are shown in separate panels, is one of several conventions for indicating the passage of time in a static medium. Here the panels show stages in the operation of the interactive population pyramid as the year advances from 1950 to 1955. Three kinds of arrows suggest movement; transparent "ghosts" represent objects that are about to appear or disappear. Such visual metaphors will not be needed in a medium where objects really can move or appear and disappear.


screen appearance ("geometricPrecision")
screen appearance ("crispEdges")
high-resolution appearance

The Heisenpixel principle states that the position and the thickness of lines cannot both be shown accurately in a medium with finite resolution. On a computer screen, parallel lines placed two-and-a-half pixels apart are nonuniform either in thickness *(top)* or in position *(middle)*. The rulings appear correct in a high-resolution printed pattern *(bottom)*. In the print version of this column, the computer-screen gratings are represented by images captured at actual size; in the Web version, the screen patterns are drawn in SVG, and the high-resolution pattern cannot be shown.

1990 PostScript program that read a data file to generate Chernoff faces.)

The pyramid is initialized to the state of the world population in 1950. When a mouse click advances the date to 1955, all the bars shift upward by the width of one bar (representing the aging of the population by five years), and then the bar lengths are adjusted to account for deaths in each age group during the five-year interval. Finally a new bar enters at the bottom of the stack, representing net births into the youngest age category. Each of these transitions is animated over 750 milliseconds. It's worth emphasizing that all of the computations are done "on the client side," by the JavaScript interpreter built into a Web browser; nothing is precomputed by the Web server.

The pyramid visualization is built on a JavaScript library called D^3 (for "data-driven documents") described in a 2011 paper by Michael Bostock, Vadim Ogievetsky and Jeffrey Heer of Stanford University. (D^3 is an open-source software project managed by Bostock, who is now with Square, Inc.)

At the heart of the D^3 framework is a simple but general mechanism for creating or modifying elements of the DOM based on supplied data. For example, in the pyramid figure, the length of each bar is determined by an entry in a table that lists population by age, gender and year. When the year changes, each bar length is relinked to a different entry in the table. The updating of the display and the animated transitions are handled behind the scenes by the D^3 library.

In designing my population pyramid I was inspired by several examples and tutorials on the D^3 website (https://github.com/mbostock/d3/wiki) and I borrowed snippets of code from them. There are at least two more population pyramids among the examples, and many other delightful tools and toys worth exploring.

Info Vis

The D^3 project is one of many to come from a thriving creative community that works under the banner of *info vis* or *data vis* (with close connections to those who do *stat vis* and *sci vis*). Michael Friendly of York University in Toronto has described the present era as a new golden age in data visualization. The *old* golden age was the 19th century, when William Playfair, Florence Nightingale, Charles Minard and a few others perfected many of the graphic devices (pie charts, line graphs) that are now standard apparatus throughout the sciences. The modern revival has brought us new forms of quantitative graphics suited to an age when considerable computational power is available even in a Web browser.

I am enthusiastic about the prospects of the info-data-stat-sci-vis biz. It has the potential to make science communication at all levels—from schoolbooks to scholarly journals—more effective and more fun. But worrisome problems remain.

First, creating active graphics takes a lot of work—and a lot of *wonk*, too. If this is an art form only for JavaScript gurus, it will not spread widely. The impact will be greater when the ideas from a program such as D^3 filter into software environments such as R and gnuplot, MATLAB and Mathematica, or even Powerpoint and Excel.

Second, the quality of graphic output is not yet up to the highest publication standards. One reason is simply the low resolution of most computer screens. This will doubtless change, but in the meantime we have to cope with issues such as the Heisenpixel problem *(see illustration at left)*.

Finally, there's the nagging anxiety about entrusting the literature of science to digital formats that are not directly accessible to the human senses. Will we still be able to see those fancy JavaScript graphics in 100 years? In 10 years? As a matter of fact, they don't work reliably even now unless you choose the right combination of hardware and software to view them.

Souvenirs of the Web

Another question arises from the choice of graphic formats whose native environment is the Web. My hope is to see these new forms of illustration become enhancements to scientific publishing, but the Web is not where scientists publish. It is a major channel for *distributing* science publications, including the 1,000 journal titles at JSTOR, for example, or the 700,000 preprints at arxiv.org. But almost all of that material comes in the form of PDFs rather than HTML documents. It's available *through* the Web, not *on* the Web. Even the conference papers and journal articles that describe the D^3 system are not HTML documents with D^3 illustrations; they are PDFs with still images.

Why do authors and readers prefer PDFs for this kind of publication? One factor may be this: A PDF is something you possess. You download it from a server, give it a name, store it in a folder. It's yours; it stays put. A website built out of HTML has a different character. It's not a thing you own but a place you visit. You can't take it home with you—although perhaps you can send a postcard or keep a small souvenir in the form of a bookmark.

Perhaps someday, when all information lives in the cloud, readers will give up their acquisitive desire for thinginess in publications. If not, documents created in the HTML/SVG/JavaScript ecosystem are at a disadvantage, because they cannot readily be turned into self-contained packages for downloading and safekeeping.

For the purpose of getting those nifty D^3 graphics into science publications, there would seem to be two plausible approaches. We could open up PDF to accept a wider range of graphics formats. I'm told this is technically feasible; the challenge is making PDF a more attractive working environment for the young programmers who come up with the cool new graphics tricks. It's worth noting that an active community works on embedding three-dimensional graphics in PDF, with impressive results.

The alternative is to seek a better way to encapsulate all the bits and pieces that constitute a Web application, so that it can be distributed in the same way as a PDF. Something resembling encapsulated HTML already exists; it's the basis of several file formats for electronic books.

In J. K. Rowling's *Harry Potter* books, newspapers for wizards are ink-on-paper publications, but the pictures on their pages spontaneously come to life. It's the best of both worlds—the familiar physical form of reading matter we've known since Gutenberg, but no longer lying still on the page. Out here in the land of Muggles we may never quite attain that kind of magic, but we could come remarkably close.

Bibliography

Adobe Systems. 1990. *PostScript Language Reference Manual*. Second edition. Reading, Mass.: Addison-Wesley.

Bostock, M., V. Ogievetsky and J. Heer. 2011. D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17:2301–2309. (Preprint online at http://vis.stanford.edu/files/2011-D3-InfoVis.pdf)

Dahlström, E., et al. (eds). 2011. Scalable Vector Graphics (SVG) 1.1 (Second edition). World Wide Web Consortium Recommendation 16 August 2011. http://www.w3.org/TR/SVG11/

Friendly, M. 2008. The golden age of statistical graphics. *Statistical Science* 23:502–535. (Available online at www.datavis.ca/papers/index.php#methods)

Heer, J., and M. Bostock. 2010. Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics* 16:1036–1043. (Preprint online at http://vis.stanford.edu/files/2010-Protovis-InfoVis.pdf)

Heer, J., M. Bostock and V. Ogievetsky. 2010. A tour through the visualization zoo. *Communications of the ACM* 53(6):59–67.

United Nations Department of Economic and Social Affairs. 2011. World Population Prospects, the 2010 Revision. http://esa.un.org/unpd/wpp/

Wilkinson, L. 2005. *The Grammar of Graphics*. Second edition. New York: Springer-Verlag.