MapShaper.org: A Map Generalization Web Service

Mark Harrower and Matt Bloch
University of Wisconsin-Madison

MapShaper.org is a Macromedia Flash-based WYSIWYG map generalization Web service for real-time generalizing and editing of vector map data. Requiring only a Web browser with the Flash plug-in, it offers line and shape simplification and smoothing. Users can upload map files, perform generalization operations, and export and download their work.

Mapping is one of the earliest forms of information filtering. Because maps are small and the world is large and complex, they require some amount of generalization. Although data reduction and information filtering are important topics of study in visualization today, cartographers long ago developed useful (and even elegant) techniques for the smoothing, aggregation, simplification, displacement, exaggeration, and classification of geographic data.[1] As Arthur Robinson—one of the most influential cartographers of the 20th century—noted, "the act of generalization gives the map its raison d'être."[2]

Generalization, however, is more than shrinking the world down to size (such as a photo reduction); it involves a conceptualization of what is important (selection) and how to employ graphic symbols to stand for other things (symbolization). Maps are powerful precisely because they are representations of reality, not reality itself. For example, a traffic flow map bears little resemblance to actual traffic, yet it helps us understand the world in a way that photographs and satellite images cannot. Its value derives from the careful selection of only the most salient data and making the invisible, visible.

Manual map generalization is a time-consuming process. Even with today's commercial geographic information systems (GIS) software and digital workflow, professional cartographers budget a significant portion of any project to cleaning and simplifying raw digital files that might contain numerous spatial errors and inconsistencies (a local commercial mapping firm reported, "roughly 10 to 30 percent of our budget" goes toward this effort). Example problems include overly detailed river networks (requiring line simplification), jagged contour lines (requiring line smoothing), or the spatial conflicts that can occur when merging data layers created at different spatial scales (for example, a building appearing on the wrong side of the road).

It's no surprise, then, that researchers have spent significant effort in creating robust and flexible automated methods for map generalization, allowing this tedious work to be off-loaded to machines.[3] With the recent and remarkable rise of on-demand mapping services such as Google Maps and MSN Virtual Earth, there is now a commercially pressing need to automate all aspects of map making, including generalization.

Furthermore, our ability to port on-demand mapping services across a range of display devices (for example, cell phones) requires that we develop smart maps that can adjust their look based on the users' needs and display environment.[4] The International Cartographic Association's Commission on Map Generalization is an active group of researchers from universities, government laboratories, and private industry who are working to solve these kinds of automatic generalization problems. We encourage readers to visit the commission's Web site at http://ica.ign.fr/ to learn about their latest efforts.

MapShaper.org

As a step toward those goals, we present MapShaper.org, a Macromedia Flash-based WYSIWYG map generalization Web service (see Figure 1). MapShaper is a simple-to-use tool suite for generalizing and editing vector map data in real time that requires only a Web browser with the Flash plug-in. It offers both line and shape simplification—reducing the number of vertices, or points, that make up a line—and smoothing—reducing the jaggedness of a line (see Figure 2). MapShaper lets users upload map files, perform various generalization operations to those files onscreen, and then export and download their work. Users can also save their work online, allowing them to log in later and continue an editing session. An efficient implementation of the Douglas-Peucker-Ramer algorithm[5] lets users adjust the degree of line and polygon simplification interactively, even for large data sets. Currently, the application supports importing ESRI Shapefiles (.shp) and exporting and converting them to .shp, .ai, .eps, and .as files. These are the most common type of digital vector files used in GIS and cartography; we are working to expand this list to include other data formats.

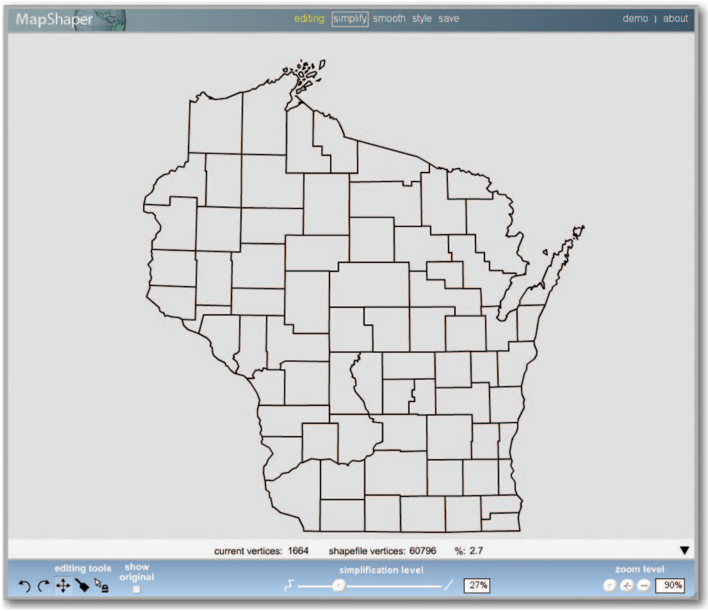
0272-1716/06/$20.00 © 2006 IEEE

Our motivation

MapShaper grew out of our frustration with the relatively limited vector generalization capabilities and the prohibitive cost of commercial GIS software. This frustration is shared by many commercial cartographers who are often forced to nonvisually perform preliminary map generalization with GIS software (for example, using a global line-smoothing algorithm), only to then export their work to graphics software such as Adobe Illustrator where they can manually fine-tune it onscreen. Furthermore, many advances in map-generalization research have yet to find their way out of the laboratory and into the hands of the public. These advances include improved line-simplification algorithms,[6] newer generalization techniques such as snakes,[7] and even holistic agent-based systems that simultaneously generalize all map elements to avoid spatial conflicts (for example, roads, buildings, and rivers always match up and are legible regardless of how much the scale is changed or the map simplified).[8,9] What was needed, we felt, was an integrated visual design environment for vector map generalization.

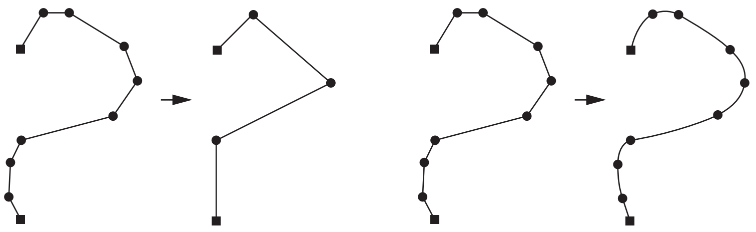
MapShaper is geared toward an audience that ranges from professional cartographers who understand generalization but want to speed up their work to the occasional map maker who might not even appreciate the need to generalize their raw digital data, let alone know how to go about doing that. This commitment to a wide range of users is reflected in our decision to use a simple, step-by-step workflow and to translate needlessly opaque generalization variables that few nonexperts would understand—such as numerical tolerances—into more friendly terms like *more/less smoothing* attached to a slider bar. MapShaper aims to take much of the guesswork out of using generalization algorithms by letting people see and interact with their data in real time.

MapShaper lets the user choose to simplify the whole map, a portion of the map using the generalization brush (described later), or individual features (for example, a single polygon or line). Because we wanted MapShaper to import and export data to popular GIS and mapping software, we had to work within the limitations of common file-format specifications. This prevented us from either creating our own file specification or from implementing features that could not be supported or understood by commercial software that users would already have on their desktop. Above all else, we wanted MapShaper to be a ready-to-use tool that could quickly solve everyday map production challenges, and that was also accessible to the widest possible audience (that is, Web based). Although other map generalization systems exist, notably the exciting work done at the GIScience Center at the University of Zurich and the COGIT Laboratory at the Institut Géographique National (France), many of these systems remain prototypes not available to the public, or are high-end block box solutions designed for a specific client, such as a national mapping agency that needs to create maps on the fly at any scale from its multiscale database (for example, Laser-Scan's Clarity system, http://www.lsl.co.uk/technologies/foundation/clarity/index.htm).[9]



current vertices: 1664 shapefile vertices: 60796 %: 2.7

1 The MapShaper interface walks users through the process and translates potentially opaque terms such as *numerical tolerances* into more friendly terms such as *more/less smoothing.*



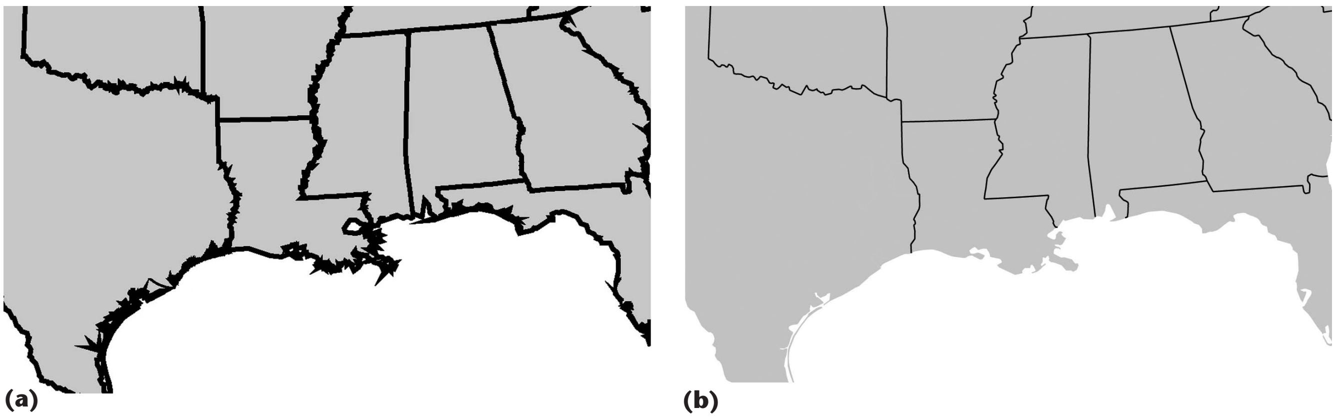
2 MapShaper offers simplification, reducing the number of points in a line, and two kinds of smoothing: (a) vertex displacement and/or (b) conversion to cubic Bezier curves.

Innovations and key features

As a free service, MapShaper.org offers five innovations and key features for vector file generalization.

Critical points preserved

Critical points are "… major inflections along a feature that contribute more overall shape information than most locations do."[10] One of the long-believed chief advantages of the DPR algorithm (which MapShaper is built upon) is its ability to automatically identify and preserve critical points. However, more recent research has shown that this is not always true, that the definition of critical points is problematic, and that DPR can generate visually unacceptable solutions (that is, make ugly maps).[10] To address this, MapShaper allows users to manually identify critical points in a polygon or network that should be locked in place while the simplification and smoothing algorithms work around them. This gives users an uncommon degree of control over the generalization process.



3 **When less is more. (a) The digital linework here is too detailed for a map of this scale, causing small geographic features to collapse onto themselves, creating an ugly, illegible mess. (b) The same map after much line generalization, demonstrating how (paradoxically) reducing the number of data points makes the map more detailed and useful.**

Lossless generalization and unlimited undoes

Our implementation of DPR stores all previous states so that users can always partially or fully retrace their steps and recover previously filtered spatial details on demand. This is important because it

- encourages the user to experiment more, and
- lets us create a new class of cartographic generalization tools (the spatial brush and the generalization eraser).

Spatial brush and generalization eraser

DPR simplifies all portions of the map equally. However, when mapmakers perform generalization by hand, they often keep spatial details in places that they deem important while filtering out more detail in other regions (see Figure 3). They will also generalize some features such as coastlines more heavily than others, such as roads. This inconsistency is not due to laziness, but helps focus the map and is driven by the cartographer's innate understanding of the map's purpose, audience, final size, and so on.

To bring this kind of flexibility into MapShaper, we have built a spatial brush that filters detail from whatever portion of the map to which it is applied. This tool functions much like the brushes found in Adobe's Photoshop, with adjustable brush size (search radius) and pressure (amount of filtering). The generalization eraser does the exact opposite: it allows recovery of localized details from portions of the map already generalized. Together, these tools let cartographers fine-tune the degree of simplification and smoothing in individual parts of a map, mimicking what they have long done with manual generalization, but with greater speed and ease.

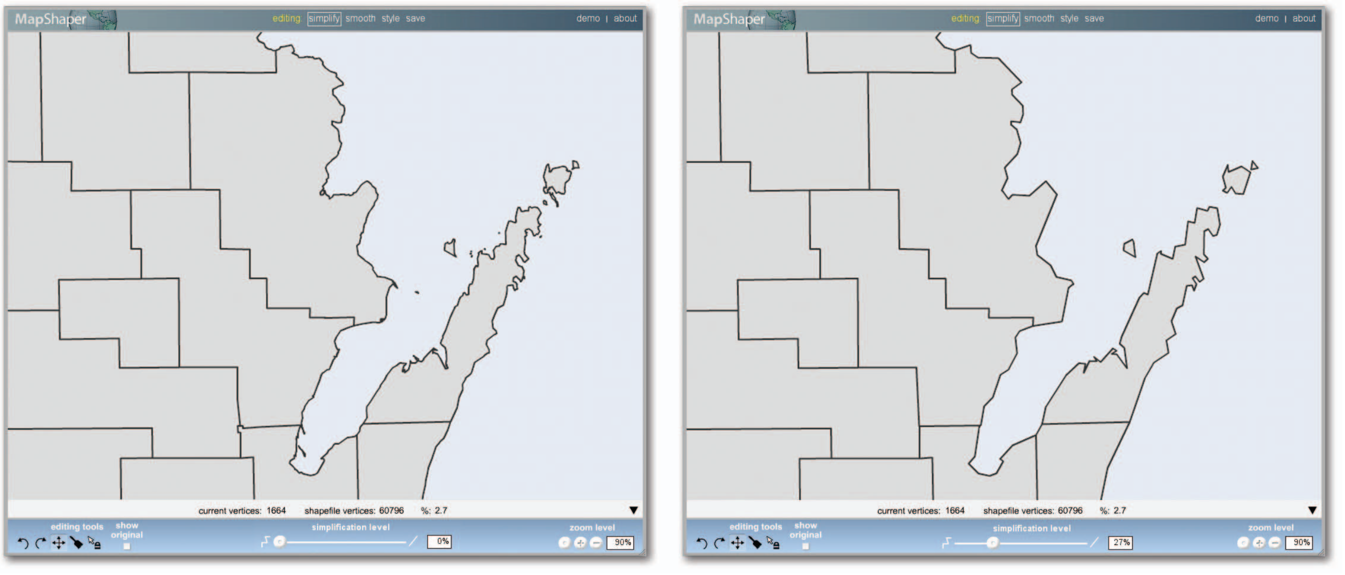
Automatically constructed topology

As Saalfeld[6] notes, the best generalization approaches are holistic and transform all map components (such as polygons) simultaneously, or en masse, searching for the best global solution. The worst approaches transform one element at a time without regard for the map's bigger picture or how changes to one polygon's shape, for example, affect its neighbors (what he terms *in vacuo generalization*). To achieve en masse generalization, however, we need topologically savvy files. Unfortunately, one of the most limiting features of many vector file formats (including ESRI's .shp) is that they don't explicitly encode topological relationships. In other words, the neighbors of any polygon (such as the surrounding counties) are not recorded, and if simplification routines were run on these files, polygons would pull apart, splitting shared borders and creating new (junk) polygons. To avoid this problem, MapShaper automatically builds topology into .shp files so that shared borders and critical shared points are preserved regardless of how much generalization is applied.

On many maps, in particular boundary maps, it's important to preserve right angles at the polygons' corners even as they become increasingly simplified, so that a place's overall spatial character remains recognizable. To support this, MapShaper automatically preserves right angles in a polygon so a place like Colorado, for example, in the Four Corners region of the western US does not end up with rounded corners or separate from its neighbors. Figure 4 shows this process in action.

Polylines converted to splines

As mentioned earlier, there are two common ways to draw curves in vector software: numerous small straight-line segments to approximate a curve (sometimes called polylines) or continuous curves described by mathematical functions (such as splines). MapShaper understands both kinds of curves and can apply smoothing and simplification to each (see Figure 5). However, we also built MapShaper to convert straight-line segment curves (found in .shp files) into more controllable and sophisticated cubic Bezier curves—which are supported in graphics software like Adobe Illustrator. Because MapShaper supports both curve types, it also needs to support two kinds of smoothing: converting polylines to Bezier curves, or moving and displacing vertices in polyline curves. In speaking with many cartographers, we found that this curve conversion feature


MapShaper
editing simplify smooth style save
demo | about
current vertices: 1664 shapefile vertices: 60796 %: 2.7
editing tools
show original
simplification level
zoom level
0%
27%
90%

4 **As the boundaries on this map become increasingly simplified, MapShaper preserves the shape of polygon corners so that places retain their overall spatial character and, thus, remain recognizable.**

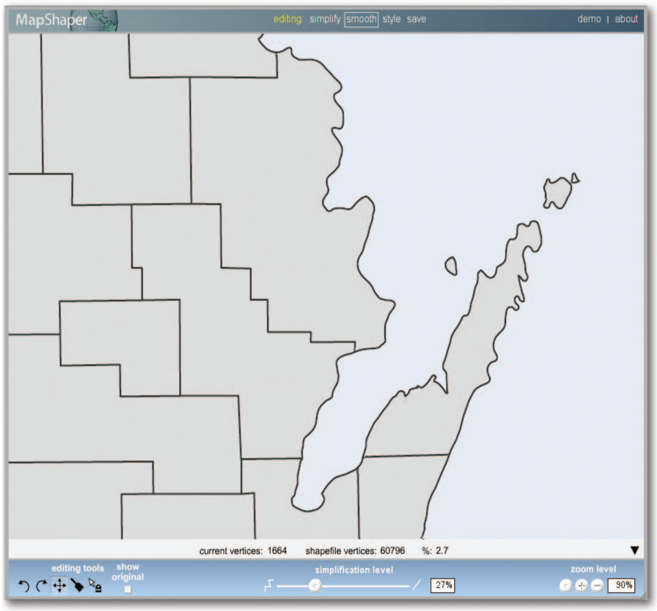
itself (independent of our generalization tools) will be helpful to them, as they routinely import polyline GIS data layers into vector graphics software yet have few options for converting those lines to curves.

MapShaper architecture

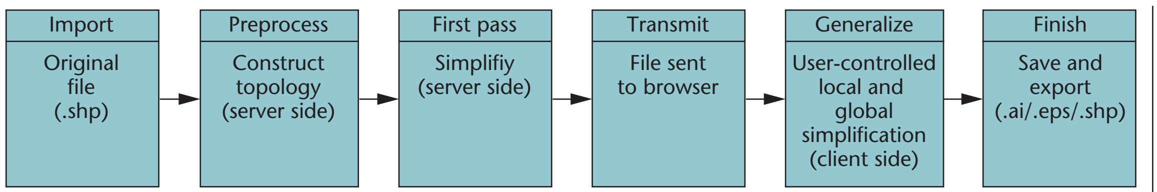
One of our key goals was to create a system that was fast enough to allow real-time visual editing of large files. Figure 6 outlines the system architecture that achieves this through a mix of server-side and client-side processing.

Preprocessing

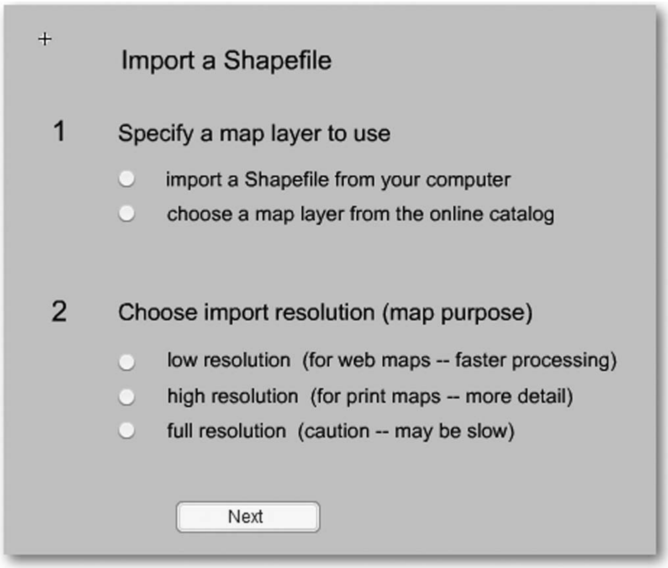
Because Flash has trouble rendering complex files, such as a county-level map of the US with a few hundred thousand vertices and line segments (which is a modest file by geographic data standards), we needed to perform some preprocessing of the .shp files on our server. After we upload the file, topology is automatically constructed, if needed, and then we run a first pass of the line simplification algorithm to strip away any unnecessary detail. The amount of detail filtered at this stage is driven by the users' end needs and how they plan to use the map. Figure 7 (next page) shows the MapShaper import dialogue, which prompts the user to pick a display environment for the final map. As cartographers have long noted, the display environment is an important factor in generalization


MapShaper
editing simplify smooth style save
demo | about
current vertices: 1664 shapefile vertices: 60796 %: 2.7
editing tools
show original
simplification level
zoom level
27%
90%

5 **The simplified map in Figure 4 further transformed and enhanced by converting jagged polyline segments to smooth, adjustable Bezier curves. The amount of smoothing is user-controlled with the slider bar.**


Import
Original file (.shp)
Preprocess
Construct topology (server side)
First pass
Simplifiy (server side)
Transmit
File sent to browser
Generalize
User-controlled local and global simplification (client side)
Finish
Save and export (.ai/.eps/.shp)

6 **MapShaper does much of the computational heavy lifting (such as constructing topology) on the server to speed up both transmission and client-side performance in Flash. The amount of preprocessing is driven by users' needs: will they use their map as a simple Web graphic or a high-resolution printed map?**



+

Import a Shapefile

1 Specify a map layer to use

- import a Shapefile from your computer
- choose a map layer from the online catalog

2 Choose import resolution (map purpose)

- low resolution (for web maps -- faster processing)
- high resolution (for print maps -- more detail)
- full resolution (caution -- may be slow)

7 Import screen for MapShaper encourages users to think about how much detail they really need in their map as a starting point for the editing session.

decisions as maps that look fine when printed might appear degraded when viewed onscreen.[11]

MapShaper has many of the favorable characteristics of a thin-client approach:

- improved and standardized performance by relying on our servers and less on the hardware at the users' end;
- improved data transmission since only smaller, optimized files are shuttled between the client and server;
- file storage for later processing; and
- always live and updated service.

Like any good Web service, MapShaper is also platform independent, something that cannot be said for GIS and illustration software.

Building topology

One potential system bottleneck is the need to build topology for large files, a computationally intensive task. Although the outcome might be similar, we can construct topology in a number of ways, some of which are unacceptably slow. Instead, we chose a hash table approach implemented in PHP to efficiently locate shared borders, allowing us to construct, for example, topology on a map of Wisconsin counties (60,668 vertices, 72 polygons, and 128 islands) in less than a second on our server (and just over 3 seconds on a 1.3-GHz Pentium M chip laptop). Moreover, we only need to run this step once—the first time the file is uploaded.

Designing the interface

Our goals with the MapShaper interface were to

- maximize the amount of map display space and minimize the tool footprint; and
- create a minimal, uncluttered work environment in which only the tools or controls actually needed at that stage were visible—to prevent overwhelming the user with irrelevant (that is, grayed-out) lists and endless tool palettes.

Because MapShaper walks users through the process (import, edit, save, and export) we hope to further reduce the learning curve and avoid the what-do-I-do-now feeling many of us experience with new software or services. Since it's been our experience that few people like to use lengthy, text-only help systems, we have instead built two additional help features into MapShaper:

- hover-over pop-ups that provide tips and hints attached directly to buttons (a common interface metaphor), and
- an animated and narrated demonstration movie that explains in more detail how to use the different components of MapShaper (a streaming file that loads on-demand).

One core interface feature is the ability to scale perfectly to any browser window size so users with a larger screen can open up their window and see more of their map (while the tools bars and metadata remain a fixed size). When space is tight, the MapShaper interface elements can scale and rearrange themselves so that MapShaper works well across a range of screen resolutions.

Two-tier service model

MapShaper offers users both sandbox and registered-user modes. The sandbox allows first-time users to immediately start using the system and learn its capabilities by working with a few of the demo data sets (a world map and a map of the US) that we provide free of charge, but with no export options. If a user registers and creates a login ID (their email address) and a password, they can upload their own data, save it to our server (for later editing), and export their files.

Conclusion and future directions

MapShaper.org is a work in progress and we will roll out new features as we develop them. The initial MapShaper release took one part-time programmer, a graphic designer, and a faculty member less than six months to develop. Working in such a small team was efficient, but also limited the scope of what we could accomplish. Some of our planned improvements include

- improved processing efficiency (for example, we hope to convert the PHP topology script to C),
- a larger list of supported file formats for importing and exporting, and
- a richer set of editing features and tools including generalization brushes with fuzzy edges.

We will also support emailing files back to users during export. Longer-term goals include investigating the possible use of more powerful and flexible simplification algorithms such as quadric-based shape simplification[12] because of DPR's documented limitations. The feedback we receive from users will, hopefully, help us refine Map-

Shaper.org into a tool that can easily and affordably bridge the current gap between GIS data and production cartography. ■

References

1. R.B. McMaster and S. Shea, *Generalization in Digital Cartography*, Assoc. of Am. Geographers, 1992.
2. A.H. Robinson et al., *Elements of Cartography*, 6th ed., John Wiley and Sons, 1995, p. 450.
3. D. Burghardt, M. Neun, and R. Weibel, "Generalization Services on the Web—A Classification and an Initial Prototype Implementation," *Cartography and Geographic Information Science*, vol. 32, no. 4, 2005, pp. 257-268.
4. P. Hardy, "Mobile Mapping On-Demand—Active Representation and Automated Generalisation of Spatial Databases for the Wireless Handheld Information Appliance," *Proc. Joint Symp. Soc. of Cartographers and British Cartographic Society*, 2000; http://www.hardy.34sp.com/papers/2000_bcs_soc_oxford_pgh.pdf
5. D. Douglas and T. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," *The Canadian Cartographer*, vol. 10, no. 2, 1973, pp. 112-122.
6. A. Saalfeld, "Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm," *Cartography and Geographic Information Science*, vol. 26, no. 1, 1999, pp. 7-18.
7. M. Bader and M. Barrault, "Improving Snakes for Linear Feature Displacement in Cartographic Generalization," *Proc. 5th Int'l Conf. GeoComputation*, CD-ROM, 2000; http://www.geocomputation.org/2000/GC034/Gc034.htm.
8. J.M. Ware, C.B. Jones, and N. Thomas, "Automated Map Generalization with Multiple Operators: A Simulated Annealing Approach," *Int'l J. Geographical Information Science*, vol. 17, no. 8, 2003, pp. 743-769.
9. V. Smith, "Demonstratable Interoperability of Agent-Based Generalization with Open, Geospatial Clients," *Proc. 8th ICA Workshop Generalization and Multiple Representation*, Int'l Cartographic Assoc. 2005; http://aci.ign.fr/Acoruna/Papers/Smith.pdf.
10. G. Dutton, "Scale, Sinuosity, and Point Selection in Digital Line Generalization," *Cartography and Geographic Information Science*, vol. 26, no. 1, 1999, pp. 33-53.
11. R.B. McMaster, "Automated Line Generalization," *Cartographica*, vol. 24, no. 2, pp. 74-111.
12. M. Garland and Y. Zhou, "Quadric-Based Simplification in Any Dimension," *ACM Trans. Graphics*, vol. 24, no. 2, 2005, pp. 209-239.

Mark Harrower *is an assistant professor of geography and associate director of the Cartography Laboratory at the University of Wisconsin-Madison. His research interests include interactive and animated mapping systems, perceptual and cognitive issues in map reading, interface design, and developing new tools for map production. Harrower has an MS and PhD from Pennsylvania State University, both in geography. Contact him at maharrower@wisc.edu.*

Matt Bloch *is a graduate student in GIS and cartography the University of Wisconsin-Madison. His research interests include interactive Web mapping, cartographic generalization, and interface design. Contact him at bloch@wisc.edu.*

For further information on this or any other computing topic, please visit our Digital Library at http://www.computer.org/publications/dlib.