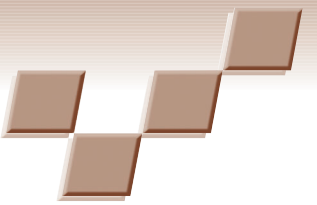


Attribute Blocks: Visualizing Multiple Continuously Defined Attributes

James R. Miller
University of Kansas

Attribute blocks are a technique for visualizing continuously varying attributes across a region, letting you quickly locate patterns in large data sets. Attribute blocks can be displayed in a flat 2D projection without needing constant 3D dynamic rotations.

Visualization of multiple attributes across a region is a complex problem for which several experimental tools have been developed. In some cases, the attributes correspond to events occurring at discrete locations; in others, the data are phenomena that vary continuously across a region. In the discrete case, the focus is often on using maps for geospatial referencing. For example, events might be bunched together in some spaces and sparse in others. Much of the research effort therefore seeks methods that adjust the physical presentation of the space to make it more amenable to visualizing all the events and their relative geographical locations. This might include, for example, distorting space to enlarge areas with dense concentrations of events while shrinking others with relatively few events. In the continuously varying case, space distortion isn't appropriate. Instead, the emphasis is on visualizing multiple attribute values at all points throughout the space.

This article describes a new technique based on *attribute blocks*, a dynamically configurable array of lenses through which users can visualize specific attribute values at given locations throughout a region. Users can exploit this ability to dynamically adjust the array's configuration as well as the attribute blocks' size and origin in this new exploratory tool for discovering patterns of interest in multivariate data sets.

Attribute blocks

I use the technique described in this article to visualize n attributes defined continuously across a region on a single map. Coloring each pixel using some multivariate function of the attribute values at that location is ineffective; as I note in the sidebar, approaches such as multivariate choropleth maps haven't proved effective for more than two or three variables.

I instead arrange all or an interactively selected subset of the n attributes into a dynamically configurable $k_r \times k_c$ array of visual representations called an *attribute block*. Each element of the attribute block array encodes a single attribute value. All attributes are defined everywhere; the attribute block array simply acts as a "screen door" of lenses, each allowing a single attribute value to be seen at the specific location. The dimensions of each lens—denoted $b_r \times b_c$—are also dynamically adjustable. I then tile the current $k_r \times k_c$ array of $b_r \times b_c$ lenses across the entire map. Figure 1 illustrates a basic

Previous Work

Phenomena of interest from a visualization viewpoint are either discrete objects or events whose locations are one of the attributes that must be visualized or continuous functions that are defined everywhere. Examples of the former include sightings of endangered species, voting results, and windmill locations. These might be rendered using glyphs or icons whose attributes (color, shape, topology, and so on) are determined by associated attributes. Examples of continuous attributes include average temperature or height above sea level. Continuous phenomena like these have values everywhere, even when they're only captured at discrete locations.

Keim et al.'s PixelMaps is a technique for displaying values for a single attribute whose geolocation must be visualized.[1] They address problems such as ensuring that distinct points are mapped to distinct pixels, and that absolute and relative positions of points in the input set are preserved as much as possible. Addressing these goals can cause map scales to

Continued on page 58

 0272-1716/07/$25.00 © 2007 IEEE

Continued from page 57

become distorted. I focus on application domains in which many attributes are defined continuously throughout the model space and in which nonlinear distortions of model space aren't allowed.

When focusing on the visualization of multiple attributes defined continuously throughout a geographic region, one obvious approach is to simply display one 3D surface $z = f_i(x, y)$ for each attribute i. The vertical scales are generally unrelated, however—for example, we might wish to visualize how temperature, precipitation, and air pressure vary throughout a region—hence, there would be no significance whatsoever to the fact that one surface was above another or that two surfaces intersected. Yet, for example, a surface intersection region might draw an observer's attention, and the observer might unconsciously try to assign some significance to this visual manifestation. In earlier work, my colleagues and I used multiple surfaces for attribute display, but only when the attributes were closely related and used the same units.[2] However, the display required constant dynamic rotations to enable interpretation, and even then it was effective for only fairly small geographic regions.

Another obvious issue with this approach is that functions such as $f_i(x, y)$ are rarely, if ever, explicitly known. Instead, the user often employs some sort of (regular or irregular) discrete data sampling followed by an interpolation process to create an approximation to functions such as $f_i(x, y)$. A common approach is based on interpolating the data onto a regular grid indexed by discrete geospatial (x, y) coordinates at some appropriate resolution. Such data gathering and interpolation schemes are beyond this article's scope. (Slocum et al. and references cited therein discuss a few methods.)[3] Once this interpolation process is complete, you essentially have a 2D array for each attribute, which represents a discrete approximation to $f_i(x, y)$ across some finite domain. Each element of the array corresponds to one grid cell within the domain, and you can use subsequent interpolation to further approximate how the attribute value might vary across the region corresponding to a given grid cell.

A complete survey of multidimensional visualization techniques is beyond this article's scope; however, I highlight a few relevant techniques and relate them to the attribute blocks approach.

Beddow used a technique somewhat similar to ours to visualize atmospheric data sets generated by the NASA Goddard Space Flight Center.[4] He recorded hourly averages for various attributes (such as magnetic strength, plasma temperature, and ion density) over many days. In his visualization, the horizontal axis represented the hour and the vertical axis encoded the day. Each resulting grid cell (indexed by [day, hour]) contained an array of patches in which an attribute's value determined the corresponding patch's color. The display captured the data set's resolution completely, and the position of the colored patches within the grid cell were irrelevant because there was no need to visualize how that value might change across the area represented by the display cell.

Although similar in appearance, my application differs in several respects. Most notably, I wanted to decouple attribute-value display resolution from the resolution at which the attribute data is stored, and to do so with minimal computational overhead. For example, I wanted to avoid having to regenerate and resample polygonal descriptions and attribute values at each interaction during an exploratory process. So, I defined the geometry and all attributes once at the resolution of the original data set, and then arranged for a low-level scan-conversion process to determine which attribute value to display on a pixel-by-pixel basis.

Slocum classifies techniques for displaying multivariate attributes in geospatial applications based on whether the user compares multiple maps visually or combines all attributes on a single map.[3] In the former case, the user typically encodes one attribute per map. This multiple map approach requires visually integrating the distinct maps to visualize all of the attributes at a given location. By contrast, when you use a single map for the attributes, you need a scheme to encode all the attributes at locations throughout that map. Researchers have explored several such techniques, most of which rely on color and/or texture mixtures. Alternatively, characteristics (for example, color, shape, size, and topology) of special symbols can encode a collection of attributes at the symbols' location.

MacEachren et al. used GeoVista Studio to develop techniques for visualizing geospatially referenced objects with multiple attributes.[5] Their multiform bivariate matrix correlates a statistic of interest (cancer mortality rates, in their example) to pairs of possible risk factors (environmental factors, health care access, and so on). Their matrix uses side-by-side displays. They use the map in position (i, j), for example, to visualize the extent to which risk factors i and j lead to increased mortality rates throughout the region of study (the US, in their case). Their displays present data that's both continuously defined (for example, presence of atmospheric emissions) and discretely defined (such as mortality statistics). Each map in the matrix displays only a single pair of attributes, however. The data is binned for each bivariate map, and they use one of four colors to visualize the resulting data.

Choropleth maps use attributes to determine the shading of a region on a map. Frequently the regions have geopolitical significance independent of the data—for example, they might represent counties, states, or countries.[3] Univariate choropleth maps are probably the most common; however, researchers have used bivariate and even trivariate choropleth maps. The bivariate technique maps two independent attributes into two distinct color scales, combines the resulting colors, and uses them in the corresponding map region.

Rather than mixing three distinct color scales,

approaches that use trivariate choropleth maps frequently employ texture in lieu of one or two of the colors.[3] You can theoretically extend this technique to *n* attributes, but even the trivariate case can be difficult to grasp, and you generally need relatively large geographic regions when using textures to encode attribute values.

Multivariate dot mapping is an alternative to color and texture mixing for single map visualization of multiple attributes. Attribute values are rarely constant throughout a region, hence drawing the entire region in a single color can be deceiving. Dot mapping originated as a technique to better visualize discrete phenomena when additional information could be exploited to illustrate how the phenomena varied throughout a region on a map.[3] For example, instead of visualizing a state's population by drawing the entire state using a color selected from a range according to the state's total population, you can display dots throughout the state, concentrating the dots locally to reflect the population's actual nonuniform distribution throughout the state.

For multiple discrete attributes, multivariate dot mapping uses differently colored dots for each attribute in an analogous fashion. User testing indicates that this approach is slightly more effective than an approach using multiple individual attribute maps when three attributes are being visualized.[6] However, it might not be at all effective for larger numbers of attributes. Our interest is in visualizing an arbitrary number of continuously varying attributes, so this approach isn't directly applicable. Nevertheless, our attribute block technique might be viewed as an analog of multivariate dot mapping in the continuous domain, particularly when our attribute block size is 1 pixel.

A completely different approach to displaying multivariate attribute data involves generating symbols located at points of interest, with their color, size, shape, and topology encoding quantities of interest. This technique was first developed for visualizing multiple attribute values at specific locations.[7] This scheme involved families of stick figure icons in which one reference leg was fixed, with other legs branching away from the fixed leg. A specific topology characterized each family. For a given visualization application, the user selected a specific family and placed instances of the family throughout the field of view. The geometric characteristics of each *n*-legged instance encoded $n - 1$ attribute values at the instance's position. An *n*th attribute value could be represented if the stick figure's overall orientation was allowed to vary. With a sufficiently dense placement of stick figures, the resulting texture in the display could reveal interesting aspects of the data.[7]

Researchers have also successfully used stick figure icons to compare different but related data sets. For example, Keller uses these icons to locate hot spots in pairs of magnetic resonance imaging images. Neither MRI image is itself displayed; instead, the display consists of an array of stick figure icons. The values at location (*i, j*) in each of the two MRI images determine the shape and size of the icon at location (*i, j*) in the display.[8] In Keller's example, a hot spot appears as a bright region in the display.

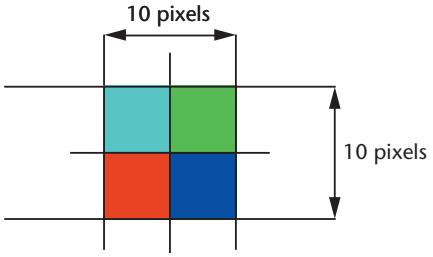
Chernoff faces is another variation on this general idea. The icons placed in the field of view are faces, and the size, shape, and color of various facial features (eyes, nose, mouth, and so on) are generated from corresponding attribute values.[3]

More recently, Healey and Enns extended the basic idea of icon-based representation of multiple attributes using *perceptual texture elements*.[9] A pexel is a polygon of fixed width whose height and color can be used to encode attribute values at a given location. Healey and Enns use pexel density and regularity to encode additional attributes. They identified three levels of regularity: regular, irregular, and random. Given sufficient numbers of pexels, you can perceive changes in pexel placement density, independent of regularity.[9]

Healey and Enns report on extensive user studies using pexels in two visualization applications: an oceanographic study involving plankton densities and a severe weather application studying typhoon activity in the Pacific. Their studies confirm that pexels effectively visualize four attributes at given locations, each mapped to one of height, color, density, and regularity of pexels in a region. They hypothesize that you can use motion and orientation to increase the number of attributes that can be presented in dynamic displays.[9]

References

1. D.A. Keim et al., "Pixel-Based Visual Data Mining of Geo-spatial Data," *Computers and Graphics*, vol. 28, no. 3, June 2004, pp. 327-344.
2. D.C. Cliburn et al., "Design and Evaluation of a Decision Support System in a Water Balance Application," *Computers and Graphics*, vol. 26, no. 6, Dec. 2002, pp. 931-949.
3. T.A. Slocum et al., *Thematic Cartography and Geographic Visualization*, 2nd ed., Pearson Prentice Hall, 2005.
4. J. Beddow, "Shape Coding of Multidimensional Data on a Microcomputer Display," *Proc. IEEE Conf. Visualization*, IEEE CS Press, 1990, pp. 238-246.
5. A.M. MacEachren et al., "Geovisualization for Knowledge Construction and Decision Support," *IEEE Computer Graphics and Applications*, vol. 24, no. 1, Jan./Feb. 2004, pp. 13-17.
6. J.E. Rogers and R.E. Groop, "Regional Portrayal with Multi-Pattern Color Dot Maps," *Cartographica*, vol. 18, no. 4, winter 1981, pp. 51-64.
7. R.M. Pickett and G.G. Grinstein, "Iconographic Displays for Visualizing Multidimensional Data," *Proc. 1988 IEEE Int'l Conf. Systems, Man, and Cybernetics*, 1988, pp. 514-519.
8. P.R. Keller and M.M. Keller, *Visual Cues: Practical Data Visualization*, IEEE CS Press, 1993.
9. C.G. Healey and J.T. Enns, "Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 2, Apr.-June 1999, pp. 145-167.



1 A basic diagram of an attribute block with $k_r \times k_c = 2 \times 2$ and $b_r \times b_c = 5 \times 5$.

diagram of the attribute blocks used to create Figure 2. Although the four cells of the attribute block in Figure 1 exhibit constant shading, the shading generally varies across a cell.

A secondary goal of this effort was to explore the extent to which it was possible to visualize multiple attributes using a single planar projection without requiring dynamic 3D rotations. Obviously, I don't advocate throwing dynamic rotations out of the toolbox, but some visualization approaches require constant or at least frequent rotations to be effective (see the sidebar). I'm simply interested in investigating tools that might be useful without needing such dynamic rotations.

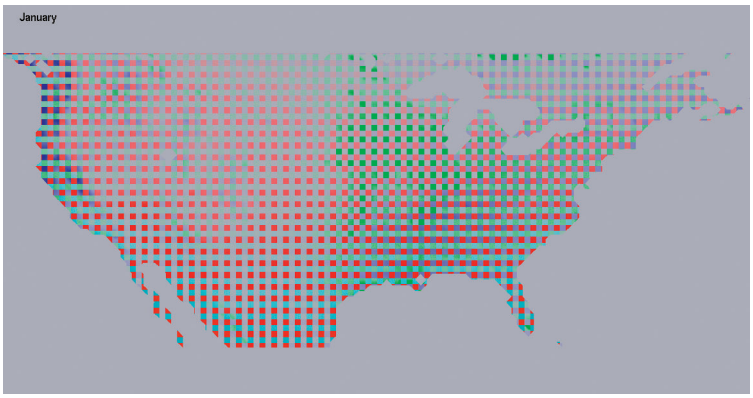
Climatology application

A region's climate affects issues ranging from the ability to grow certain crops to the effective use of land resources in general. Many public policy decisions are inextricably linked to current and potential future climatology measures because each can affect the other. Water balance gives a relatively simple yet effective measure of climatology. Conceptually, this is a simple budgeting issue: water enters the system and is consumed by the system. A water balance model is one way to simulate that process and determine a measure of a region's water balance (and hence climatology). Several such models exist. For purposes of the demonstration here, I use a relatively simple model whose primary inputs are precipitation, temperature, and soil moisture-holding capacity.[1] Depending on the specific region of the world, you can also compute the potential evapotranspiration (the amount of water that would be lost due to evaporation and plant transpiration, if available) and the actual evapotranspiration (the amount actually lost to those processes).

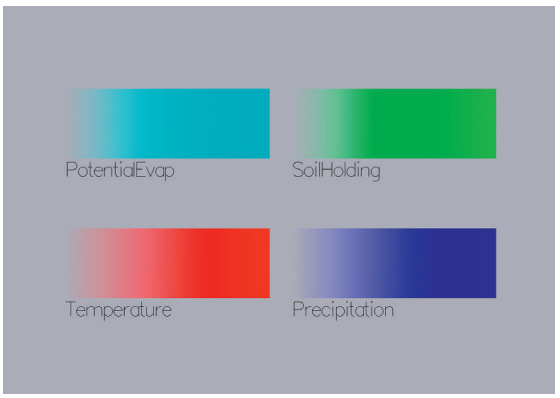
The data set I use includes 12 monthly averages and an annual average for temperature, precipitation, and soil moisture-holding capacity at a $1/2 \times 1/2$-degree (approximately 50 km at the equator) resolution. I obtained data sets describing the averages from 30 years' worth of data gathered at stations scattered around the world. This data represents the world's climatology today. In earlier work, my colleagues and I applied global circulation data to model global climate change's effects in terms of how these averages are likely to change 30 years from now.[1] I chose not to include visualizations of those climate change predictions in this article.

Using the raw data and some intermediate results, I use attribute blocks to display (subsets of) $n = 8$ attributes. Figure 2 shows a collection of $k_r \times k_c = 2 \times 2$ attribute blocks tiled across a map of the United States. The blocks encode four of the eight attributes: precipitation, temperature, soil moisture-holding capacity, and potential evapotranspiration. The attributes' color scales are dynamically configurable. For the example in Figure 2, they vary from the background color for low attribute values, ramping up to blue (for precipitation), red (for temperature), green (for soil moisture-holding capacity), and cyan (for potential evapotranspiration) for high attribute values. These color assignments are the ones my colleagues and I used in earlier work,[1] and aren't uncommon in climatology applications. Figure 3 shows the legend dynamically generated for each $k_r \times k_c$ arrangement, showing both the color scales and the current $k_r \times k_c$ structure.

Adjusting the color scales this way makes sense in this application because Figure 2's input attributes tend to drive the water balance away from 0 (that is, from perfect balance) as their values increase. For example, as temperature increases, more water is needed, and the balance decreases. As precipitation increases, more water is available, and the balance increases. By using the background color for areas where such an attribute has a low value and a more distinct color when the attribute's value is larger, the areas where a given



2 Attribute blocks in a 2×2 arrangement illustrating temperature, precipitation, soil moisture-holding capacity, and potential evapotranspiration.



3 Dynamically generated legend illustrating the color scale used as well as the current $k_r \times k_c$ structure.

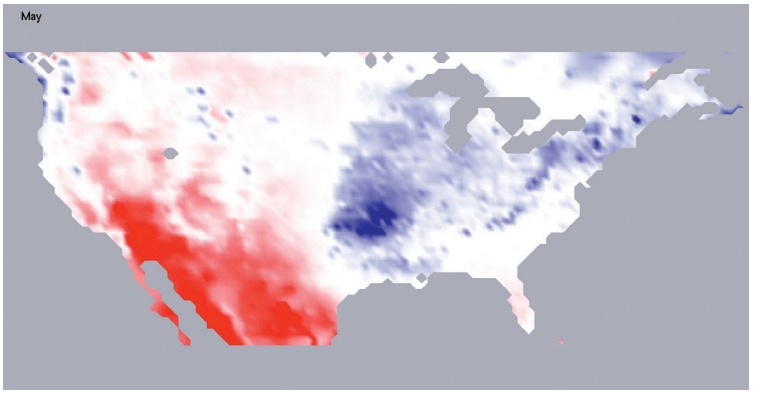
attribute most affects the balance becomes immediately obvious. By contrast, the attribute doesn't draw attention in other areas, where the background color dominates.

Looking at Figure 2, we can immediately see for the month of January that precipitation has a strong effect in the West and Northwest; soil moisture-holding capacity has an especially strong impact in the upper Midwest; and potential evapotranspiration plays a relatively small role only in the South, most notably in Florida, Mexico, and the Baja.

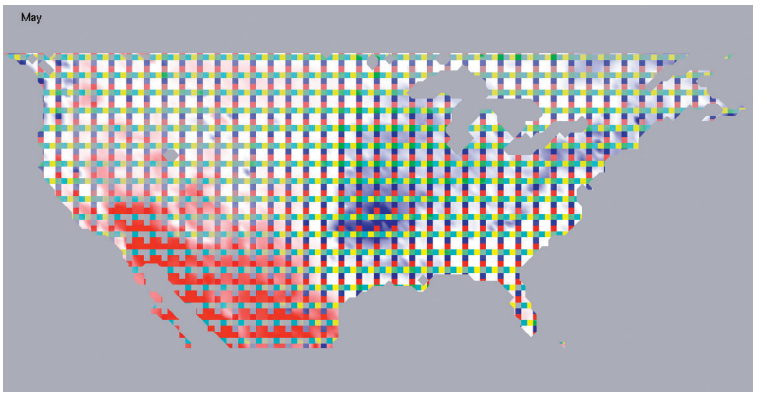
Setting $k_r \times k_c$ to be 1×1 lets us see a single attribute. For example, the result of the water balance model as generated using the parameters shown in Figure 2 (but for the month of May) is presented in the $k_r \times k_c = 1 \times 1$ attribute block display of Figure 4. The water balance is the measure computed by the underlying simulation and is important in the context of my example application because, more than any of the raw input or other intermediate variables, it represents a measure of a region's climatology. The balance might be negative, indicating a deficit—that is, not enough water is available to satisfy the need—or it might be positive, indicating more water was available than the environment needed. To emphasize this dichotomy, the water balance color scale uses a diverging scale in which negative values are mapped from red (highest deficit) to white (no deficit); positive values are then mapped from white (no surplus) to blue (highest surplus).

Not all $k_r \times k_c$ array positions need be unique. Figure 5, for example, shows a $k_r \times k_c = 3 \times 3$ array. The upper 2×2 portion of this array displays the water balance, while the other five array positions display five of the remaining seven attributes down the first column and across the bottom row. The other attributes are mapped to colors, as described earlier. The superposition of the attributes in Figure 5 helps us understand why the balances illustrated in Figure 4 occur. What we see in Figure 5 is the water balance of Figure 4 with the attributes laid on top of it. For the month of May, for example, we can conclude that on average, the large deficit in the Southwest extending into Mexico is chiefly due to high temperatures (and hence high potential evapotranspiration), coupled with low precipitation and low soil moisture-holding capacity. The Midwest has comparable temperatures, but much more precipitation and (especially in the upper Midwest) better soil moisture-holding capacity, and hence has a surplus.

If we compare Figures 4 and 5, we see a weakness in this method. Some isolated patches of surplus in the Rockies are apparent in Figure 4. Several of these areas are overlaid by other attribute blocks in Figure 5, so not only do we fail to see those surpluses in Figure 5, but we see visual representations of attributes that contributed to a surplus in the context of a balanced water budget. Users can employ the interactive data exploration tools—especially those relating to the periodic attribute block grid's characteristics—to ameliorate the effects of this sampling-related problem. A better and more satisfying approach would be to develop automated tools to help the user determine ideal attribute block sizes and locate such potential problem areas.



4 The water balance computed from the model for the month of May.



5 Attribute blocks in a 3×3 arrangement illustrating water balance (in the upper 2×2 portion of each 3×3 grid), precipitation, temperature, soil moisture-holding capacity, potential evapotranspiration, and actual evapotranspiration.

Interactive controls for dynamic data exploration

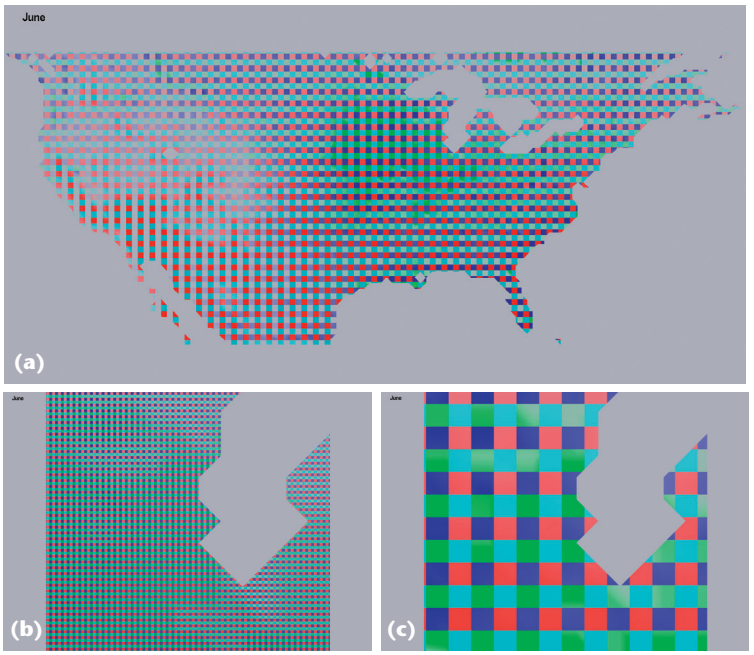
Visualization is a process of exploration, and you must be able to dynamically alter fields of view, attribute block sizes and positions, configurations, and so on. Responses must be immediate to facilitate the exploration process. As de Oliveira and Levkowitz argue,[2] many patterns and relationships only become obvious as you dynamically explore the data set. I accomplish this exploration primarily via several interactive operations that are only effective if display updates are at least as fast as user interactions. None of these facilities require geometric redefinition or attribute resampling, and all are reflected immediately in the display.

Adjusting attribute block configurations

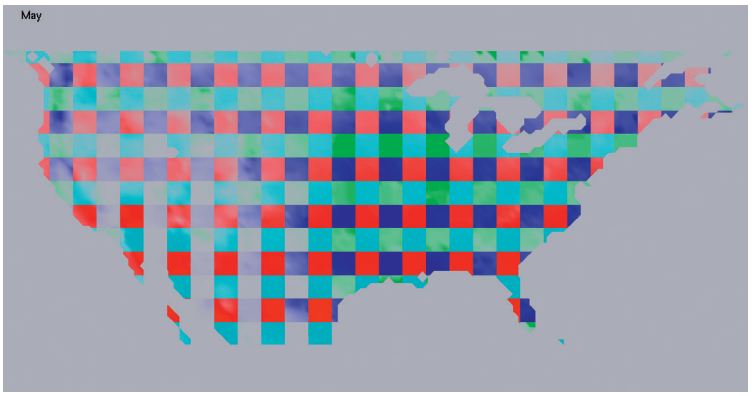
In the prototype used to generate the images in this article, supported attribute block configurations were predefined in a file read by the program at startup. (The user could interactively specify what single attribute was displayed for a 1×1 configuration or for unsupported $k_r \times k_c$ configurations.) The current version lets users define this mapping at runtime.

The periodic attribute block grid

The attribute block grid is simply a periodic function in two directions whose frequency is controlled by



6 Data zooming options: (a) original view, (b) zoomed, attribute blocks fixed size in pixels; and (c) zoomed, attribute blocks fixed size in model space.



7 Two results generated with our method using single-pixel-sized neighborhoods.

adjusting cell block sizes and whose phase shift is adjusted by fixing a starting point. Both the size of attribute block cells and the attribute block grid's origin are decoupled from the geometry of the underlying model. This might seem counterintuitive at first, but it's a powerful advantage of the method. Because all attributes are continuously defined everywhere, locking display resolution to that of the polygons used for display doesn't make sense. Instead, letting users dynamically adjust these two parameters of the periodic function gives them explicit control over where attribute grid cells appear in the display and how big their visual representation will be. An attribute block cell will, in general, overlap more than one model space polygon.

From the model space polygons' perspective, I define the geometry to the underlying graphics API in a conventional fashion using arbitrary polygons and specifying all n attribute values at each vertex. During low-level polygon scan conversion, the graphics system determines the attribute block cell containing each pixel and uses the corresponding color map for that attribute to set the pixel color. (See the "Implementation" section for more details.)

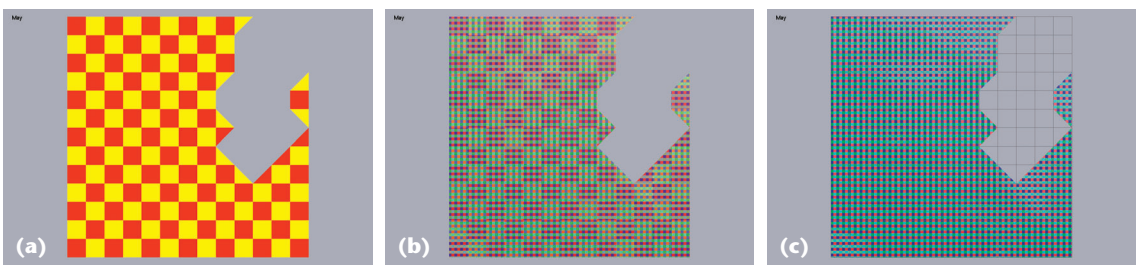
Adjusting attribute block cell sizes. The user can specify dimensions of cells within attribute blocks in model or pixel space. If the user requests, for example, that attribute blocks always have a fixed pixel size, zooming in on geometry has a side effect of visualizing attribute variation at a higher resolution (see Figures 6a and 6b). Alternatively, by specifying attribute block sizes in model space, users can achieve independent control over display resolution and attribute variation resolution (see Figure 6c).

Adjusting attribute block grid starting point. Dynamically adjusting the attribute block grid's origin effectively slides the "screen door of lenses" pixel-by-pixel across the surface, revealing features that might not be obvious with the current cell block sizes. For example, I could ameliorate the problem related to the inability to see the pockets of surplus in the Rockies in Figure 5 by sliding the attribute block array in this manner.

Adjusting miscellaneous display attributes

Attribute values are linearly interpolated across the polygon's interior, and hence this interpolation continues across the interior of the attribute blocks themselves. The obviously changing colors in the interiors of several of the attribute blocks demonstrate this. In Figure 7, for example, such variations are most obvious in the interiors of precipitation and soil attribute blocks.

At any point in time, attribute blocks can be larger or smaller than the actual data resolution. Moreover, if



8 Comparing resolution of raw data set with that of attribute block visualizations: (a) grid cell resolution; (b) grid cells superimposed on attribute blocks; and (c) grid cell lines superimposed on attribute blocks.

attribute block sizes are specified in pixel space, this relationship will change dynamically as the user zooms in or out. Users therefore need to be able to visualize how the resolution of the point-sampled data sets compares to that of the attribute blocks themselves. Figure 8a shows the data resolution for the display in Figure 6b; Figure 8b shows the two images superimposed; and Figure 8c shows just the polygon grid lines superimposed on the attribute block display. (The polygons in the running example are bounded by latitude and longitude lines, hence the orientations of the polygons and the attribute blocks are the same. This is coincidental and is neither assumed nor exploited anywhere in the application.)

Attribute rows and columns

A special case of note is when exactly one of k_r or k_c is 1. In that case, the attribute blocks become attribute rows or attribute columns, respectively (see Figure 9). The relative advantages and disadvantages of attribute rows and columns as compared to ordinary attribute blocks needs more study. For example, in some situations it might be easier to visualize how an important attribute varies across an entire field of view using rows or columns of attributes. Although attribute blocks provide good local information in terms of how relevant attributes affect important quantities in a given region, the block structure can make it difficult to compare trends for two or more attributes simultaneously in terms of their impact on composite quantities.

Pointillism

Adjusting the attribute blocks' pixel sizes can be useful when exploring a data set for unknown patterns in input data. The display in Figure 10 is the $k_r \times k_c = 2 \times 2$ case of Figure 2, but for the month of May with attribute block sizes set in pixel space with $b_r \times b_c = 1 \times 1$. The resulting display has the feel of pointillism (a type of graphical display that uses small colored dots to simulate an image composed of a much wider range of colors). It's immediately clear where various input parameters have the strongest effect: soil in the upper Midwest, temperature in the Southwest, and precipitation in the Northwest and much of the East.

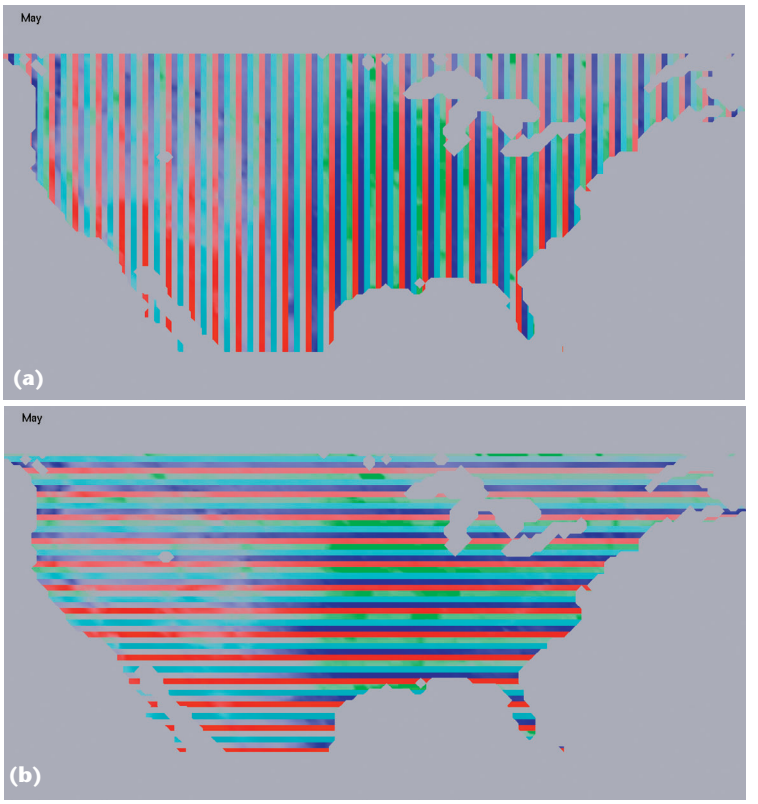
The example in Figure 11a (next page) is an attempt to replicate this feel for the $k_r \times k_c = 3 \times 3$ case of Figure 5 (and again with $b_r \times b_c = 1 \times 1$ in pixel space). The attribute blocks for the principal input parameters tend to be too small to permit an adequate visualization of their impact on the computed water balance.

The 10×10-pixel attribute blocks in Figure 5 seemed reasonable. But how small can the blocks be while remaining effective? The optimal (or minimal) size seems to depend on the application and the data. Empirically, I've found $b_r \times b_c = 3 \times 3$ to be a reasonable lower boundary in many instances. Figure 11b shows the $b_r \times b_c = 3 \times 3$ case with $k_r \times k_c = 3 \times 3$ for our sample data set in the month of May.

Using attribute blocks for uncertainty visualization

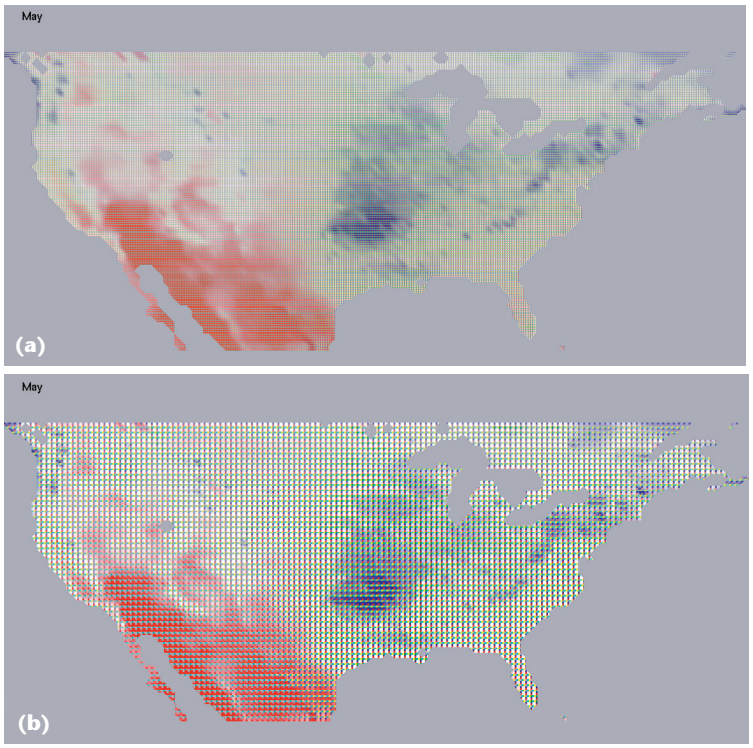
Scientific models and the physical data that drive them are subject to errors and uncertainty. Error in this context is generally understood to describe raw data incorrectly captured or recorded. By contrast, uncertainty refers to how confident you are in the results of some physical simulation, either a self-contained application or one driven by one or more raw data sets.[1]

Many techniques exist for modeling, measuring, and displaying uncertainty arising from scientific computations.[1,3] For the water balance model example, many interpolated global temperature, precipitation, and soil data sets are available. These interpolated data sets differ in what raw physical data the scientists used, how they interpolated the data, and whether they used various corrections to known problems (or relevant physical characteristics such as altitude). Comparing the



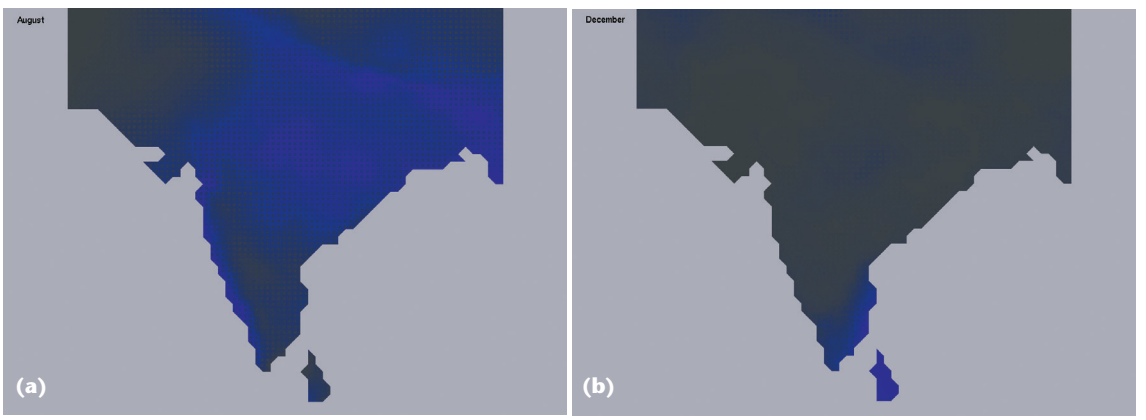
9 Two special cases occurring when either k_r or k_c is 1: (a) attribute columns and (b) attribute rows.

10 Setting attribute block size to 1×1 in pixel space achieves a pointillism effect.



11 Attribute blocks in a 3×3 arrangement as in Figure 5, except with (a) $b_r \times b_c = 1 \times 1$ and (b) $b_r \times b_c = 3 \times 3$.

12 Exploring temperature uncertainty in India.



13 Exploring precipitation uncertainty in India: (a) considerable differences exist for the month of August, but (b) few differences exist for the month of December.

results from using these different data sets provides one way to measure uncertainty in model results.[1]

We can use attribute blocks to look for patterns of agreement or disagreement between raw data sets. For example, we can assign different temperature data sources to an attribute block array's elements and explore a region. In this case, we use the same color ramp for all attribute block elements because they all correspond to temperature. We'd expect to see little or no block pattern in areas where the data sets are in agreement. On the other hand, areas where the attribute blocks are visually obvious suggest appreciable disagreement among the selected data sets.

Figure 12 shows temperature data sets for India and some surrounding areas. Although there is clearly little disagreement in the heart of the country, considerable differences exist in projected average temperatures for the month of January in the mountainous area of the Northeast, as well as some disagreement in the Northwest. Specifically, two of the data sets consistently predict cooler temperatures in these areas. Examining the corresponding data sources, we learn that one of them uses an interpolation scheme modified to account for altitude differences. The other uses fewer raw stations and limits data to a recent 30-year period.

Similar areas of uncertainty are evident in precipitation data sets. Figure 13a shows considerable differences in the predicted average precipitation levels for the month of August. However, little disagreement exists for the month of December, as Figure 13b indicates.

Some comparisons

I compare some visualizations using attribute blocks with select techniques from the "Previous Work" sidebar. As noted, application domains can be characterized by data with defined values only at discrete locations or continuously throughout the display region. Display goals and strategies are different for the two domains, so I don't include here those that are primarily applicable to the discrete domain.

We can generate Beddow's displays (see the sidebar) as a special case of attribute blocks. Specifically, suppose there are n attributes whose values have been sampled at a spatial resolution of $D_x \times D_y$. We set k_r and k_c so

there is a slot for each of the n attributes (that is, $n \leq k_r * k_c$). We specify that attribute block sizes are fixed in model space and set to $b_r = D_y/k_r$ and $b_c = D_x/k_c$. Linear interpolation across the attribute blocks can be enabled or disabled as desired.

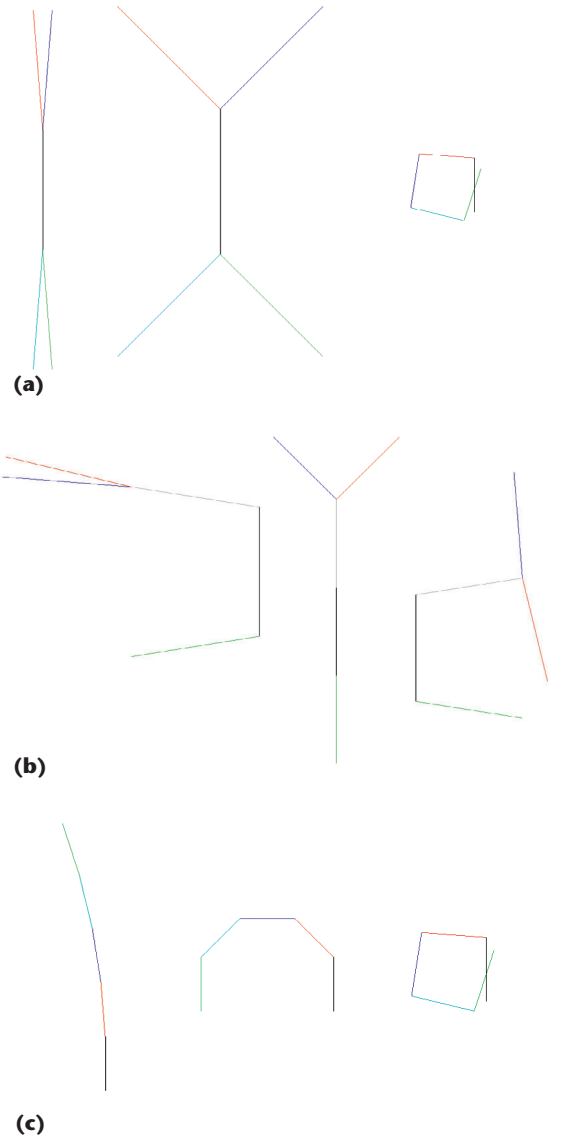
We could easily use stick figure icons (see the sidebar) for discrete or continuous data. In their original paper, Pickett and Grinstein identify 12 members of a stick figure family with four controllable limb angles for attributes. (We could use a fifth angle for the base limb's rotational orientation, but I ignore that here.) Family members had either one or two limbs extending from each parent limb. Certainly higher out-degrees are possible, but our initial experiments didn't lead to any configuration that appeared to perform better. Choosing a family member is a trial-and-error process. Members whose limbs had out-degrees of 1 or 2 were perhaps a bit easier to master. Those with out-degrees of 2 vary from being tall and skinny for small attribute values to being short and bushy for large values (Figure 14a). When all out-degrees are 1, the icon varies from being straight for minimum values of all attributes to being closed for maximum values (see Figure 14c). Family member 5 (Figure 14b) has mixed out-degrees.

Many important configuration considerations relate to the use of stick figures, including spaces, color, and angle range. Space limitations prevent me from discussing them in depth here, however.

Figure 15 (next page) compares attribute blocks with stick figure icons for the month of May for the entire US as well as for a small region containing Florida. I generated the stick figures using colored nonoverlapping stick figures from family member 12 (all out-degrees are 1) with attributes mapped into an angle range of 0 to 90 degrees. Choosing a background that maximized visualization of limb colors was an issue. I ultimately chose a white background.

As the top row of Figure 15 shows, the influence of soil moisture-holding capacity in the upper Midwest is more obvious in the attribute block display. Moreover, the influence of potential evapotranspiration in Florida—apparent in the attribute block display in the bottom row of Figure 15—is less obvious in the corresponding stick figure display. On close examination, it's apparent that the angle between the blue precipitation leg and the cyan potential evapotranspiration leg increases as you move south, but it isn't as immediately obvious as it is in the attribute block display. Notice, too, the mini hotspot of high soil moisture-holding capacity in South Florida apparent in the attribute block display. This hotspot is also evident in the stick figure display; in two corresponding stick figures, the soil leg makes a near 90-degree angle with the potential evapotranspiration leg in that area.

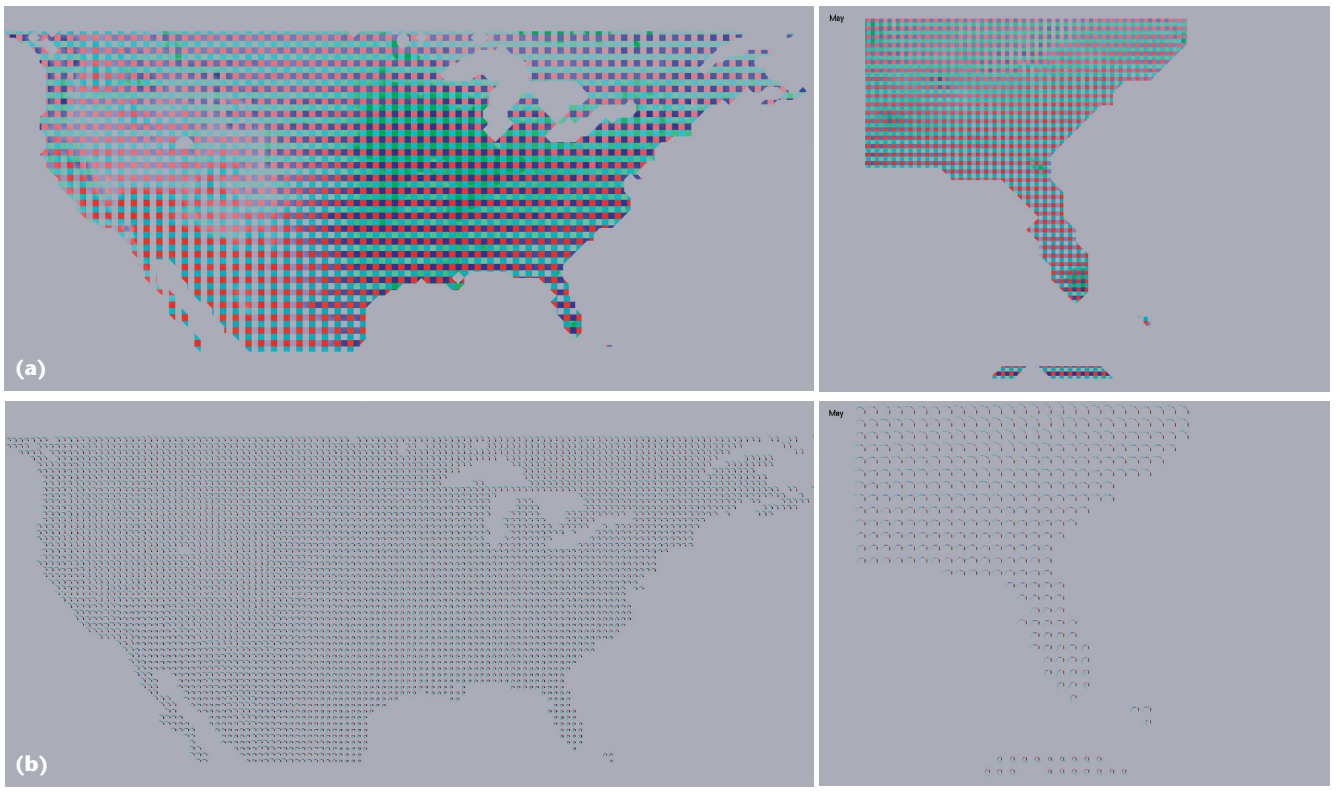
Figure 16 focuses on the same southeastern portion of the US. Figure 16a uses stick figure family member 5 (see also Figure 14) to show water balance (white leg), temperature (red), precipitation (blue), and soil (green). The white water balance leg uses a –90 to 90-degree range, the first half representing a deficit and the latter half representing a surplus. The soil leg coming off the icon's bottom also uses a –90 to 90-degree range,



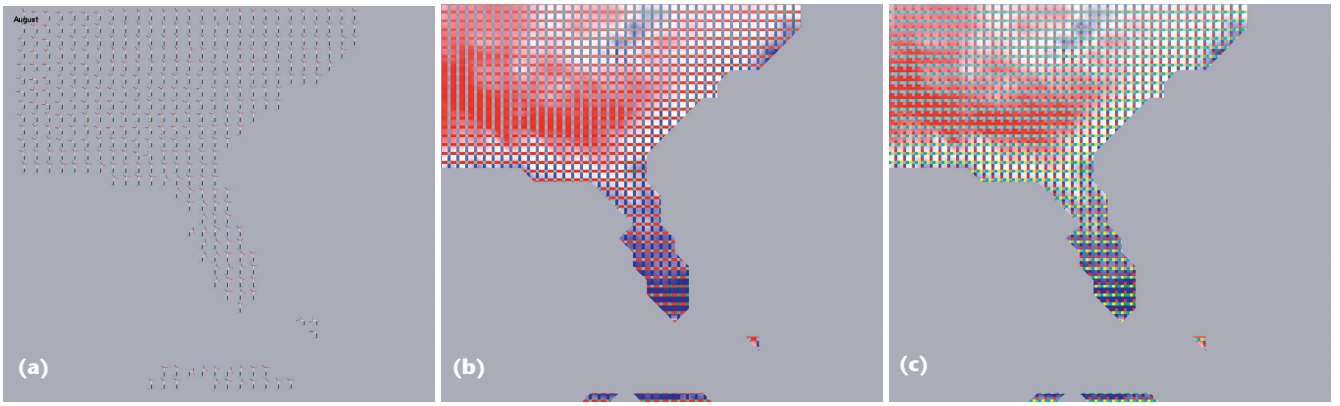
14 **Illustrating various attribute displays for family members 1, 5, and 12: (a) members whose limbs had out-degrees of 2, (b) members whose limbs had out-degrees of 1, and (c) members whose limbs had mixed out-degrees.**

indicating minimum to maximum soil moisture-holding capacity. Figure 16b uses attribute blocks to display the same data. The basic layout is $k_r \times k_c = 3 \times 3$, with the larger upper right 2×2 portion displaying the water balance, the first column displaying precipitation, the bottom row displaying temperature, and the lower left block showing the soil moisture-holding capacity. This arrangement—and in particular, the way the first column and first row are defined in a sort of Y-topology—was meant to mimic the stick figure configuration as much as possible.

The extreme deficit just to the north of Florida is obvious in the attribute block display in that all of the larger 2×2 blocks are very red. In this area of the stick figure display, all of the white legs bend to the left to indicate



15 Comparing (a) attribute blocks with (b) stick figure icons.

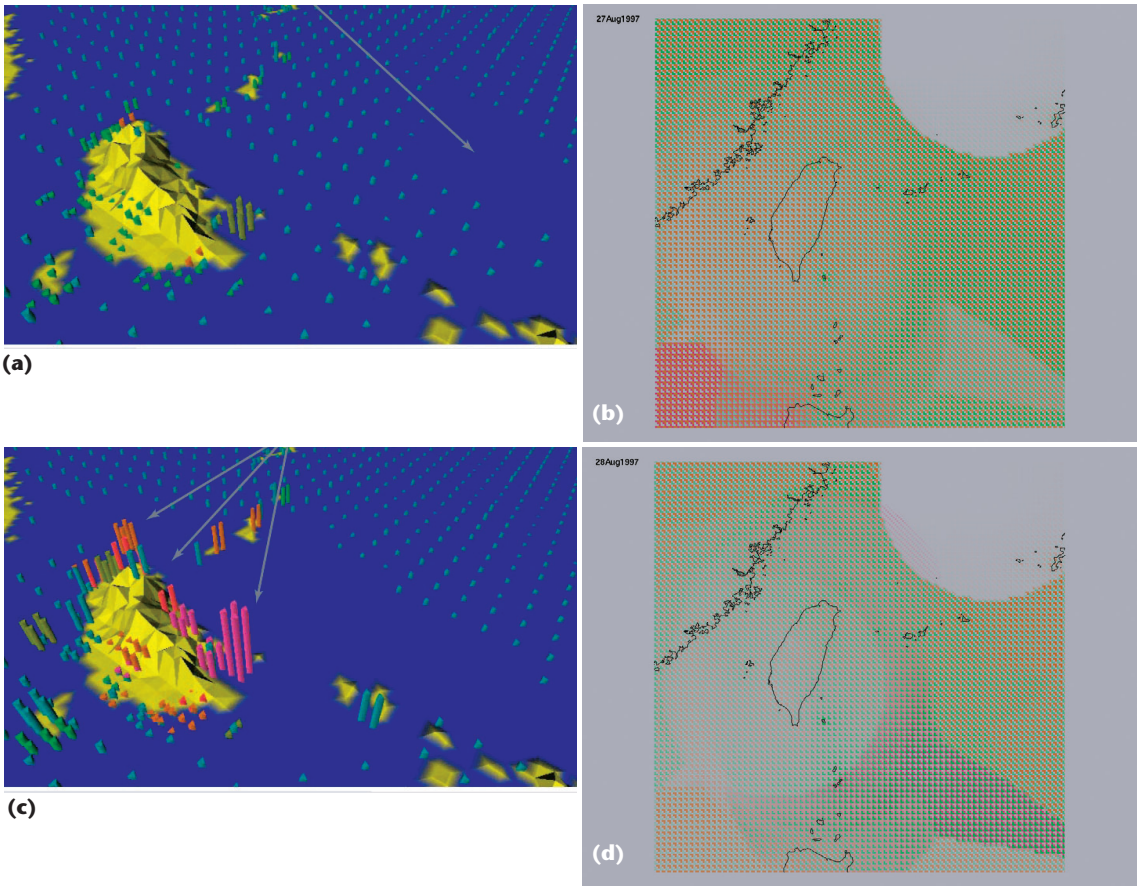


16 Comparing stick figure icons with attribute blocks in displaying water balance and associated model parameters: (a) stick figure representation of water balance data, (b) attribute block representation of the data, and (c) adding two additional attributes to the display in (b).

the deficit. Not surprisingly, the temperature's impact increases as you go south, as Figures 16a and 16b illustrate. In the attribute block display, the bottom row becomes a deeper red; in the stick figure display, the red leg approaches a 90-degree angle with the white. Similarly, there is little precipitation to the north and considerable precipitation to the south. The first blue column of the attribute blocks becomes a deeper blue, while the blue precipitation leg angle increases. Similar remarks are applicable to the display's soil portions.

Both visualizations convey the same information. The water balance result—generally the key quantity of interest—seems to be more immediately obvious in the attribute block display than in the stick figure display. Moreover, the attribute block mechanism lets us display other information of interest without a significant impact. For example, Figure 16c shows an attribute block display in which precipitation and temperature are combined in the first column, and the potential and actual evapotranspiration have occupied the vacated spots in the bottom row. The data described previously remains readily apparent in this view, and we can additionally observe that potential and actual evapotranspiration increase significantly as we go south.

Rather than compare perceptual texture elements (pexel) visualizations of our climatology data and mod-



17 Comparison of the Healey and Enns visualizations with attribute blocks: (a) Healey and Enns visualization of the 27 August typhoon; (b) the 27 August attribute block display; (c) Healey and Enns visualization of the 28 August typhoon; and (d) the 28 August attribute block display.

els, I used attribute blocks to generate visualizations of the Healey and Enns typhoon example. I obtained the typhoon season data from the original NOAA Web site (http://www.ncdc.noaa.gov). This data set consisted of typhoon season data recorded at various weather stations throughout the 1987 typhoon season. The Healey and Enns example focused on a typhoon that approached Taiwan in late August of that year.[4] I extracted the raw scattered data and applied a basic inverse-distance weighted interpolation algorithm to the data to interpolate it throughout the region of interest.

Healey and Enns' visualizations presented three attributes: wind speed, pressure, and precipitation during the typhoon event. I evaluated several attribute block configurations, and eventually settled on a 3×3-attribute block configuration in which precipitation was displayed along the diagonal, from upper left to lower right. Wind speed occupied the remaining lower-left cells, and I assigned pressure to the upper-right cells. For Figure 17, I used $b_r \times b_c = 4 \times 4$.

Figure 17a shows the Healey and Enns visualization,[4] and Figure 17b is the corresponding attribute blocks visualization. The orientation and field of view are somewhat different, but it's reasonably easy to see how Taiwan in the Healey and Enns figure relates to Taiwan's coastline shown in ours. Healey and Enns noted that the typhoon's center has virtually no wind, and appears as an absence of pexels in that view. The attribute block display shows wind speed as a mapping from the background color to a shade of green. The same no-wind area is apparent in the attribute block display because the attribute block cells assigned to wind are mapped to (nearly) the background color in that area.

Assigning attributes to cells in the attribute blocks as I've described led to a nice side effect. The right-angle shape taken on by the wind speed and pressure cells is readily identifiable in the displays and improves perception of the corresponding data. For example, an area of high pressure over Taiwan is apparent in Figure 17b, but has decreased noticeably in the corresponding display of Figure 17d. Similarly, the heavy rain apparent in the Healey and Enns visualization in Figure 17a is also obvious in the attribute block display of Figure 17d, this time because the purple in the diagonal precipitation cells is clearly apparent approaching from the southeast.

The exact locations of features differ somewhat between the Healey and Enns and attribute block figures. For example, the area of low wind in the center of the typhoon is in a slightly different place in Figures 17a and 17b. Similarly, the rain is on the coast in Figure 17c, whereas Figure 17d shows it off the coast. This is probably because of differences in how I extracted and used the raw NOAA data. Those issues aside, the ability to visualize critical features—such as high precipitation

and absence of wind—is clear. I don't claim that either approach is uniformly better, but that they represent different ways to see the data.

Implementation

Our implementation uses the OpenGL programmable vertex and fragment shader capability.[5] Typically, an application defines a scene's geometry to the OpenGL engine during a display callback using geometric forms and attributes predefined in OpenGL. For example, OpenGL knows about such rendering-related attributes as RGB colors and normal vectors. Current values of these attributes are associated with vertices of primitives and are sent down the graphics pipeline with them.

In a conventional OpenGL program, the pipeline is a black box. It contains a vertex shader, which performs per-vertex operations such as calculating a lighting model. OpenGL linearly interpolates the results of these per-vertex operations across the interior of the owning primitive during the primitive reassembly and scan-conversion process. It then invokes a fragment shader for each pixel identified during scan conversion. This shader is responsible for calculating and setting the pixel color. Using the OpenGL Shading Language, I can replace the standard vertex and fragment shader computations with arbitrary processing, provided that I follow certain basic conventions.[5]

The ability to decouple attribute block cells from the geometry definition was critical to our implementation's success.

I used this mechanism for my attribute block renderings. I associated the eight water-balance attributes with vertices of the polygon and communicated them to a specialized vertex shader. That shader mapped the attribute values into a 0-to-1 range and arranged for these mapped values to be linearly interpolated across the primitive.

Our fragment shader is somewhat more complex. It uses the current $k_r \times k_c$ and $b_r \times b_c$ settings to determine the attribute block cell in which the current pixel lies. This is actually a two-step process because it first determines the (i, j) cell index $(0 < i < k_r; 0 < j < k_c)$ from the k_r, k_c, b_r, and b_c values, and then determines the actual attribute to which that (i, j) position corresponds based on the current assignment of attributes to cell locations in the attribute-block array. Once the fragment shader identifies the appropriate attribute, it accesses the corresponding interpolated attribute value and color ramp and generates the resulting pixel color.

The ability to relegate these operations to the vertex and especially the fragment shader lets us decouple attribute block cells from the geometry definition and was critical to our implementation's success. Moreover, the entire C++ program is independent of the actual data. Our data file begins with a short header describing the data. The vertex and fragment shaders are then in separate files and use the variable names identified in the data file. To use attribute blocks in a different context, you need only design the data file header appropriately.

From the fragment shader's perspective, attribute block cell sizes are always specified in pixel units. If the user wants to fix these cell sizes in model space, the application computes the pixel dimensions based on the current field of view and viewport dimensions at the beginning of each display callback. In either event, the fragment shader simply gets (via uniform variables[5]) the current $b_r \times b_c$ dimensions in pixel units.

Ongoing and future work

Although my preliminary experiences with attribute block visualizations are positive, several limitations and areas of further study are apparent. Most notably, several questions related to sampling and its relationship to determining ideal attribute block cell sizes merit additional work. As I discussed in the context of Figures 4 and 5, in the current system, the user must detect and deal with undersampling problems by adjusting cell sizes and origins. Although this sometimes works, a much better approach would be to develop tools that could analyze the underlying data, suggest cell sizes, and perhaps even develop simple animations by automatically cycling through cell size and/or phase shift adjustments.

Attribute data is linearly interpolated across the interior of model space polygons, hence also across the interior of attribute block cells. A better understanding of this linear interpolation's limitations is needed. Does switching between linear interpolation functions in an attribute block cell's interior cause visible discontinuities? Should discontinuities be preserved so as to emphasize the resolution of the underlying data? Are there advantages to using higher-order interpolation functions?

My approach currently uses only color to display attribute values. I could also use texture variations (see the sidebar). As for the colors themselves, I need a more systematic study of color ensembles and how users perceive them. Earlier work used color mappings,[1] but because we never displayed mixed data types in that effort, we avoided potential problems related to color overloading. For example, a red ramp in Figure 5 indicates temperature; in that same figure, it's also part of the diverging color ramp used for the water balance.

The interplay between shapes taken on by attribute cell assignments in an attribute block and perception needs more study. As I mentioned when comparing attribute blocks with pexel displays, the specific shapes taken on by the wind speed, pressure, and precipitation cells in the attribute block display played an important role in bringing out the values of the corresponding attributes. We need to better understand how to exploit such shape-related cues. ■

Acknowledgments

I thank the reviewers for pointing me to several useful related references and for providing many extremely helpful suggestions to make the description much more complete and clear. I also thank Johan Feddema and Terry Slocum for the data sets used in the running example here as well as for providing helpful suggestions to improve interactions with the visualizations.

References

1. D.C. Cliburn et al., "Design and Evaluation of a Decision Support System in a Water Balance Application," *Computers and Graphics*, vol. 26, no. 6, Dec. 2002, pp. 931-949.
2. M.C.F. de Oliveira and H. Levkowitz, "From Visual Data Exploration to Visual Data Mining: A Survey," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 3, July-Sept. 2003, pp. 378-394.
3. A. MacEachren, "Visualizing Uncertain Information," *Cartographic Perspective*, vol. 13, 1992, pp. 10-19.
4. C.G. Healey and J.T. Enns, "Large Datasets at a Glance: Combining Textures and Colors in Scientific Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 2, Apr.-June 1999, pp. 145-167.
5. R.J. Rost, *OpenGL Shading Language*, Addison-Wesley, 2006.

James R. Miller *is an associate professor of computer science in the Department of Electrical Engineering and Computer Science and codirector of the eLearning DesignLab at the University of Kansas. His research interests include computer graphics, scientific visualization, geometric modeling, and e-learning. He has a PhD in computer science from Purdue University. Contact him at jrmiller@ku.edu.*

Article submitted: 9 Feb. 2006; revised: 14 Oct. 2006; accepted: 18 Nov. 2006.