CHAPTER NINETEEN

Animation for Visualization: Opportunities and Drawbacks

Danyel Fisher

DOES ANIMATION HELP build richer, more vivid, and more understandable visualizations, or simply confuse things?

The use of Java, Flash, Silverlight, and JavaScript on the Web has made it easier to distribute animated, interactive visualizations. Many visualizers are beginning to think about how to make their visualizations more compelling with animation. There are many good guides on how to make static visualizations more effective, and many applications support interactivity well. But animated visualization is still a new area; there is little consensus on what makes for a good animation.

The intuition behind animation seems clear enough: if a two-dimensional image is good, then a moving image should be better. Movement is familiar: we are accustomed to both moving through the real world and seeing things in it move smoothly. All around us, items move, grow, and change color in ways that we understand deeply and richly.

In a visualization, animation might help a viewer work through the logic behind an idea by showing the intermediate steps and transitions, or show how data collected over time changes. A moving image might offer a fresh perspective, or invite users to look deeper into the data presented. An animation might also smooth the change between two views, even if there is no temporal component to the data.

As an example, let's take a look at Jonathan Harris and Sep Kamvar's We Feel Fine animated visualization (*http://wefeelfine.org*). In this visualization, blog entries mentioning feelings are represented as bubbles. As users move between views, the bubbles

are reorganized into histograms and other patterns. For example, one screen shows the relative distribution of blog entries from men and women, while another shows the relative distribution of moods in the blog entries. While the bubbles fly around the screen freely, there are always a constant number on the screen. This constancy helps reinforce the idea of a sample population being organized in different ways. Animation is also used to evoke emotion: the bubbles quiver with energy, with those that represent "happy" moving differently than bubbles that represent "sad."

Not all animations are successful, though. Far too many applications simply borrow the worst of PowerPoint, flying data points across the screen with no clear purpose; elements sweep and grow and rotate through meaningless spaces, and generally only cause confusion.

I have had several occasions to build animated visualizations. In 2000, I worked with fellow grad students building GnuTellaVision, which visualized the growing Gnutella peer-to-peer network. Since then, I have been involved in a variety of projects that have shed light on animated visualization: for example, I worked on a project that explored animated scatterplots, and I was a close bystander on the DynaVis project, which looked at transitions between different visualizations. In this chapter, I will talk through some of these experiences and to try to develop some principles for animating visualizations.

Animation can be a powerful technique when used appropriately, but it can be very bad when used poorly. Some animations can enhance the visual appeal of the visualization being presented, but may make exploration of the dataset more difficult; other animated visualizations facilitate exploration. This chapter attempts to work out a framework for designing effective animated visualizations. We'll begin by looking at some background material, and then move on to a discussion of one of the most well-known animated visualizations, Hans Rosling's GapMinder. One of the projects I worked on explored animated scatterplots like GapMinder; this makes a fine launching point to discuss both successes and failures with animation. As we'll see, successful animations can display a variety of types of transformations. The DynaVis project helps illustrate how some of these transitions and transformations can work out. The chapter concludes by laying out a number of design principles for visualizations.

Principles of Animation

At its core, any animation entails showing a viewer a series of images in rapid succession. The viewer assembles these images, trying to build a coherent idea of what occurred between them. The perceptual system notes the changes between frames, so an animation can be understood as a series of visual changes between frames. When there are a small number of changes, it is quite simple to understand what has happened, and the viewer can trace the changes easily. When there are a large number of changes, it gets more complex.

The Gestalt perceptual principle of *common fate* states that viewers will group large numbers of objects together, labeling them all as a group, if they are traveling in the same direction and at the same speed. Individual objects that take their own trajectories will be seen as isolates, and will visually stand out. If all the items move in different directions, however, observers have far more difficulty following them. Perception researchers have shown that viewers have difficulty tracking more than four or five objects independently—the eye gives up, tracking only a few objects and labeling other movement as noise (Cavanagh and Alvarez 2005).

Animation in Scientific Visualization

Attendees at the annual IEEE VisWeek conference—the research summit for visualization—are divided into two groups: information visualizers and scientific visualizers. The two groups give different talks, sit in different rooms, and sometimes sit at different tables at meals. Watching the talks, one quickly notices that roughly half of the papers in the scientific visualization room feature animation, while almost no papers in the information visualization room do. You could say that the difference between the groups is that scientific visualizers are people who understand what the *x*-, *y*-, and *z*-axes actually mean: they are very good at picturing the dimensions of an image and understand the meaning of depths and distances. The dynamic processes they often represent—wind blowing over an airplane wing, hurricanes sweeping across maps, blood flowing through veins—also involve an additional dimension: that of time. As it would be difficult to squeeze its representation into any of the other three dimensions, animating is an attractive method for displaying such processes.

In contrast, data visualization is less straightforward. Information visualizers usually work with abstract data spaces, where the axes do not correspond to the real world (if they mean anything at all). Viewers need to get acclimated to the dimensions they can see, and learn how to interpret them. Consequently, there are comparatively few examples of animation published in the information visualization community. (We will discuss some of these later.)

Learning from Cartooning

Animation, of course, appears popularly in places outside of visualizations. Movies and cartoons depend on some of the same physical principles as computer animation, so several people have asked whether cartooning techniques might bring useful insights to the creation of animated visualizations. As early as 1946, the Belgian psychologist Albert Michotte noted the "perception of causality" (Michotte 1963). It is easy to believe that the movement in an animation shows intent: that this point is *chasing* another across the screen (rather than moving in an equivalent trajectory one second behind it), that this ball *hit* another (rather than "this dot stopped at point A, and this other dot moved from A to B"), and so on. Thus, we can ascribe agency and causality where none really exists.

In cartoons, of course, we wish to communicate causality. Traditional cartoonists have described how they endow drawn shapes with the "illusion of life" (Johnston and Thomas 1987) in order to convey emotion, and several rounds of research papers (Lasseter 1987; Chang and Ungar 1993) have tried to see how to distill those ideas for computer animation and visualization.

Traditional cartoonists use a barrage of techniques that are not completely true to life. *Squash and stretch*, for instance, distorts objects during movement to draw the eye toward the direction of motion: objects might stretch when they fly at their fastest, and squashing them conveys a notion of stopping, gathering energy, or changing direction. Moving items along *arcs* implies a more natural motion; motion along a straight line seems to have intent. Before objects begin moving, they *anticipate* their upcoming motion; they conclude with a *follow-through*. *Ease-in, ease-out* is a technique of timing animations: animations start slowly to emphasize direction, accelerate through the middle, and slow down again at the end. Complex acts are *staged* to draw attention to individual parts one at a time.

Visualization researchers have adapted these techniques with differing degrees of enthusiasm and success—for example, the Information Visualizer framework (Card, Robertson, and Mackinlay 1991), an early 3D animated framework, integrated several of these principles, including anticipation, arcs, and follow-through. On the other hand, some elements of this list seem distinctly inappropriate. For instance, squashing or stretching a data point distorts it, changing the nature of the visualization; thus, we can no longer describe the visualization as maintaining the consistent rule "height maps to *this*, width maps to *that*" at each frame of the animation. In their research on slideshows, Zongker and Salesin (2003) warn that many animation techniques can be distracting or deceptive, suggesting causality where none might exist. Also, they are often meant to give an illusion of emotion, which may be quite inappropriate for data visualization. (An exception would be We Feel Fine, in which the motion is supposed to convey emotion and uses these techniques effectively to do so.)

The Downsides of Animation

Animation has been less successful for data visualization than for scientific visualization. Two metastudies have looked at different types of animations—process animations and algorithm visualizations—and found that both classes have spotty track records when it comes to helping students learn more about complex processes.

The psychologist Barbara Tversky found, somewhat to her dismay, that animation did not seem to be helpful for process visualization (i.e., visualizations that show how to use a tool or how a technique works). Her article, "Animation: Can It Facilitate?" (Tversky, Morrison, and Bétrancourt 2002), reviews nearly 100 studies of animation and visualization. In no study was animation found to outperform rich static diagrams. It did beat out textual representations, though, and simple representations that simply showed start and end state without transitions.

Algorithm animation is in many ways similar to process visualization: an algorithm can be illustrated by showing the steps that it takes. Some sort algorithms, for example, are very amenable to animation: an array of values can be drawn as a sequence of bars, so the sort operations move bars around. These animations can easily show the differences between, say, a bubble sort and an insertion sort. Christopher Hundhausen, Sarah Douglas, and John Stasko (2002) tried to understand the effectiveness of algorithm visualization in the classroom, but half of the controlled studies they examined found that animation did not help students understand algorithms. Interestingly, the strongest factor predicting success was the *theory* behind the animation. Visualization was most helpful when accompanied by constructivist theories—that is, when students manipulated code or algorithms and watched a visualization that illustrated their own work, or when students were asked questions and tried to use the visualization to answer them. In contrast, animations were ineffective at transferring knowledge; passively watching an animation was not more effective than other forms of teaching.

GapMinder and Animated Scatterplots

One recent example of successful animated visualization comes from Hans Rosling's GapMinder (*http://www.gapminder.org*). Rosling is a professor of Global Health from Sweden, and his talk at the February 2006 Technology, Entertainment, Design (TED) conference* riveted first a live audience, then many more online. He collected public health statistics from international sources and, in his brief talk, plotted them on a scatterplot. In the visualization, individual points represent countries, with *x* and *y* values representing statistics such as life expectancy and average number of children and each point's area being proportionate to the population of the country it represents. Rosling first shows single frames—the statistics of the countries in a single year—before starting to trace their progress through time, animating between the images with yearly steps in between.

Figure 19-1 shows three frames of a GapMinder-like animation. On the *x*-axis is the life expectancy at birth; on the *y*-axis is the infant mortality rate. The size of bubbles is proportionate to the population. Color-coding is per continent; the largest two dots are China and India.

Rosling's animations are compelling: he narrates the dots' movement, describing their relative progress. China puts public health programs in place and its dot floats upward, followed by other countries trying the same strategy. Another country's economy booms, and its dot starts to move rapidly rightward. Rosling uses this animation to make powerful points about both our preconceptions about public health problems and the differences between the first and third world, and the animation helps viewers follow the points he is making.

* Available online at *http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html*. Rosling presented similar discussions at TED 2007 and TED 2009.

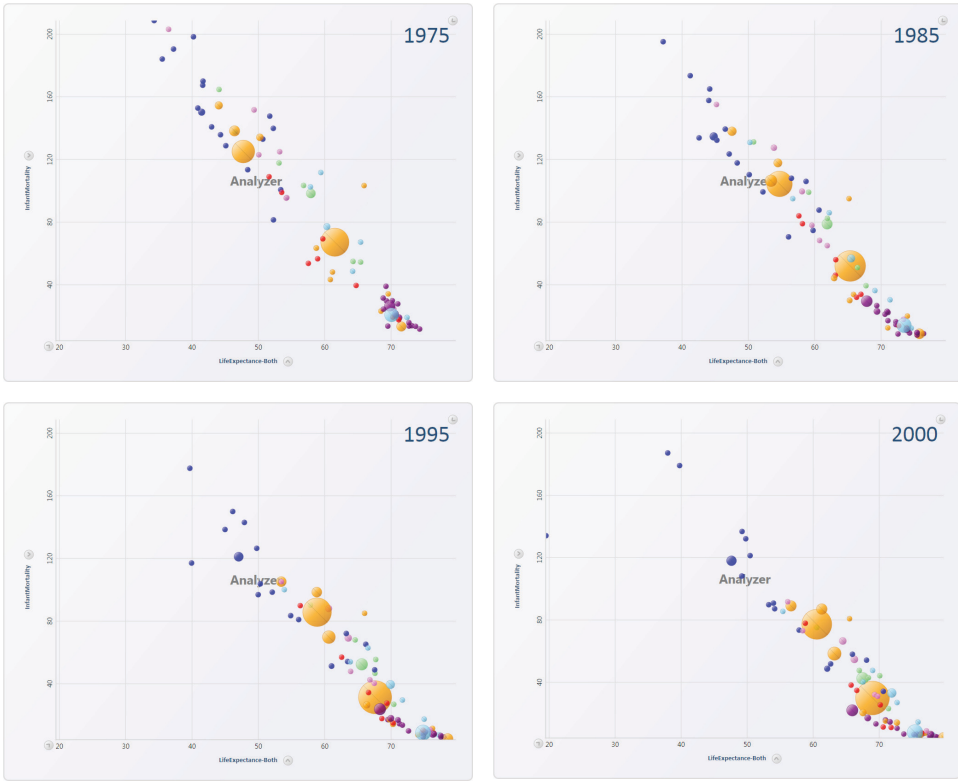

Figure 19-1. *A GapMinder-like visualization showing information about a set of 75 countries in 1975, 1985, 1995, and 2000; this chart plots life expectancy (x axis) against infant mortality (y axis)—countries at the top-left have a high infant mortality and a short life expectancy*

Too many dots?

The perceptual psychology research mentioned earlier showed that people have trouble tracking more than four moving points at a time. In his presentation, Rosling is able to guide the audience, showing them where to look, and his narration helps them see which points to focus on. He describes the progress that a nation is making with the assistance of a long pointer stick; it is quite clear where to look. This reduces confusion.

It also helps that many of the two-dimensional scatterplots he uses have unambiguously "good" and "bad" directions: it is good for a country to move toward a higher GDP and a longer life expectancy (i.e., to go up and to the right), and bad to move in the opposite direction (down and to the left).

With Rosling's sure hand guiding the watcher's gaze, the visualization is very effective. But if a temporal scatterplot were incorporated into a standard spreadsheet, would it be useful for people who were trying to learn about the data?

Testing Animated Scatterplots

At Microsoft Research, we became curious about whether these techniques could work for people who were not familiar with the data. We reimplemented a GapMinder-like animation as a base case, plotting points at appropriate (*x*, *y*) locations and interpolating them smoothly by year. We then considered three alternative static visualizations that contained the same amount of information as the animation. First, of course, we could simply take individual frames (as in Figure 19-1). Even in our earliest sketches, however, we realized this was a bad idea: it was too difficult to trace the movement of points between frames. The ability to follow the general trajectories of the various countries and to compare them is a critical part of GapMinder; we wanted users to have a notion of continuity, of points moving from one place to another, and the individual frames simply were not helpful.

We therefore implemented two additional views, using the same set of countries and the same axes as Figure 19-1, for the years 1975–2000. The first is a *tracks* view, which shows all the paths overlaid on one another (Figure 19-2). The second is a *small multiples* view, which draws each path independently on separate axes (Figure 19-3). In the tracks view, we cue time with translucency; in the small multiples view, we instead show time by changing the sizes of the dots.

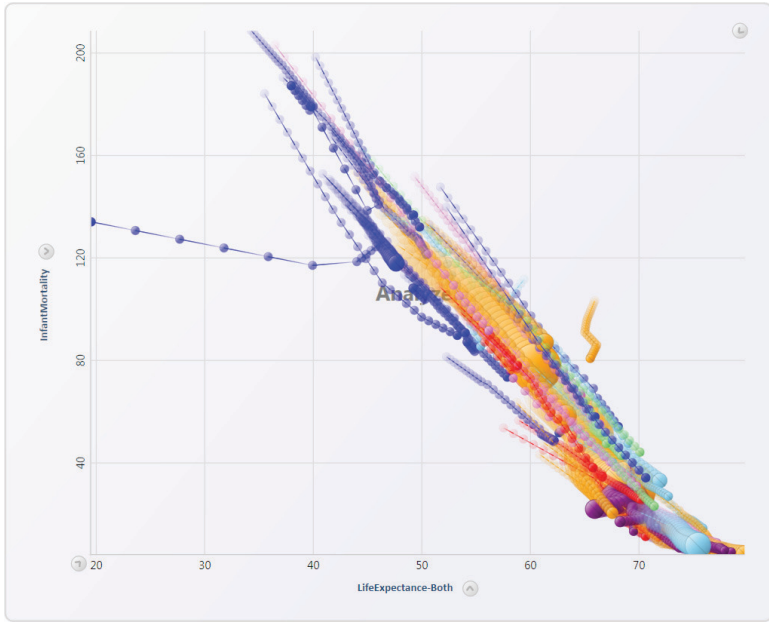

Figure 19-2. *Tracks view in which each country is represented as a series of dots that become more opaque over time; years are connected with faded streaks*

Figure 19-3. *Small multiples view in which each country is in its own tiny coordinate system: dots grow larger to indicate the progression of time*

We wanted to understand how well users performed with the animation, as compared with these static representations. Users can set up their own scatterplots at the GapMinder website, but would they be able to learn anything new from their data?

We chose 30 different combinations of (*x*, *y*) values based on public health and demographic data from the United Nations, and presented users with fairly simple questions such as "In this scatterplot, which country rises the most in GDP?" and "In this scatterplot, which continent has the most countries with diminishing marriage rates?" We recruited users who were familiar with scatterplots, and who dealt with data in their daily work. Some subjects got to "explore" the data, and sat in front of a computer answering questions on their own. Others got a "presentation," in which a narrator showed them the visualization or played the animation. We measured both time and accuracy as they then answered the questions.

The study's numerical results are detailed in Robertson et al. (2008). The major conclusions, however, can be stated quite simply: animation is both slower and less accurate at conveying the information than the other modalities.

Exploration with animation is slower

We found that when users explored the data on their own, they would often play through the animation dozens of times, checking to see which country would be the correct answer to the question. In contrast, those who viewed a presentation and

could not control the animation on their own answered far more rapidly: they were forced to choose an answer and go with it. Thus, animation in exploration was the slowest of the conditions, while animation in presentation was the fastest.

Interestingly, this might shed light on why the process animations by Tversky et al. found so little success. In our tests, users clearly wanted to be able to move both forward and backward through time; perhaps this is true of process animations, too. More effort may be required to get the same information from an animation as opposed to a series of static images, because you have to replay the entire thing rather than just jumping directly to the parts you want to see.

Animation is less accurate

Despite the extra time the users spent with the animation, the users who were shown the static visualizations were always more accurate at answering the questions. That is, the animation appeared to detract from the users' ability to correctly answer questions. Their accuracy was not correlated with speed: the extra time they spent in exploration did not seem to drive better outcomes.

This seems like bad news for animation: it was slower and less accurate at communicating the information. On the other hand, we found the animation to be more engaging and emotionally powerful: one pilot subject saw life expectancy in a war-torn country plummet by 30 years and gasped audibly. Generally, users preferred to work with the animation, finding it more enjoyable and exciting than the other modes. They also found it more frustrating, though: "Where did that dot go?" asked one angrily, as a data point that had been steadily rising suddenly dropped.

These findings suggest that Rosling's talk is doing something different from what our users experienced. Critically, Rosling knows what the answer is: he has worked through the data, knows the rhetorical point he wishes to make, and is bringing the viewers along. He runs much of his presentation on the same set of axes, so the viewers don't get disoriented. His data is reasonably simple: few of the countries he highlights make major reversals in their trends, and when he animates many countries at once, they stay in a fairly close pack, traveling in the same direction. He chooses his axes so the countries move in consistent directions, allowing users to track origins and goals easily. He takes advantage of the Gestalt principle of common fate to group them, and he narrates their transitions for maximum clarity.

In contrast, our users had to contend with short sessions, had to track countries that suffered abrupt reversals, and did not have a narrator to explain what they were about to see; rather than learning the answer from the narrator, they had to discover it themselves. This suggests to us that what we were asking our users to do was very different from what Rosling is doing—so different, in fact, that it deserves its own section.

Presentation Is Not Exploration

An analyst sitting before a spreadsheet does not know what the data will show, and needs to play with it from a couple of different angles, looking for correlations, connections, and ideas that might be concealed in the data. The process is one of foraging—it rewards rapidly reviewing a given chart or view to see whether there is something interesting to investigate, followed by moving on with a new filter or a different image.

In contrast, presenters are experts in their own data. They have already cleaned errors from the dataset, perhaps removing a couple of outliers or highlighting data points that support the core ideas they want to communicate. They have picked axes and a time range that illustrate their point well, and they can guide the viewers' perception of the data. Most importantly, they are less likely to need to scrub back and forth, as we saw users doing with our animation, in order to check whether they have overlooked a previous point. In these conditions, animation makes a lot of sense: it allows the presenter to explain a point vividly and dramatically.

The experience of exploration is different from the experience of presentation. It is easy to forget this, because many of our tools mix the two together. That is, many packages offer ways to make a chart look glossy and ready for presentation, and those tools are not clearly separated from the tools for making the chart legible and ready for analysis. In Microsoft Excel, for example, the same menu that controls whether my axis has a log scale also helps me decide whether to finish my bar chart with a glossy color. The former of these choices is critical to exploration; the latter is primarily useful for presentation. After I finish analyzing data in Excel, I can copy the chart directly into PowerPoint and show the result. As a result of this seamlessness, few people who use this popular software have seriously discussed the important distinctions between presentation and exploration.

Table 19-1 summarizes major differences between the needs of exploration and presentation.

Table 19-1. *Differentiating exploration from presentation*

	Exploration	**Presentation**
Characteristics	Data is surprising. Data may have outliers. Data is likely to move unpredictably. Viewer controls interaction.	Data is well known to the presenter. Data has been cleaned. Viewer is passive.

Table 19-1. *Differentiating exploration from presentation*

	Exploration	Presentation
Goals/procedures	Analyze multiple dimensions at once. Change mappings many times. Look for trends and holes.	Present fewer dimensions to make a point. Walk through dimensions clearly. Highlight critical points. Group points together to show trends and motion.

These two perspectives are not completely disjoint, of course. Many interactive web applications allow users to explore a few dimensions, while still not exposing raw data. The tension between presentation and exploration suggests that designers need to consider the purpose of their visualizations. There are design trade-offs, not only for animation, but more generally.

Types of Animation

Some forms of animation are most suited to presentation, while others work well for exploration. In this section, we'll discuss a hierarchy of different types of transformations, ranging from changing the view on a visualization to changing the axes on which the visualization is plotted to changing the data of the visualization. Let's begin with an example of a system that needs to manage two different types of changes.

Dynamic Data, Animated Recentering

In 2001, peer-to-peer file sharing was becoming an exciting topic. The Gnutella system was one of the first large-scale networks, and I was in a group of students who thought it would make a good subject of study. Gnutella was a little different from other peer-to-peer systems. The earlier Napster had kept a detailed index of everything on the network; BitTorrent would later skip indexing entirely. Gnutella passed search requests between peers, bouncing around the questions and waiting for replies. When I used a peer-to-peer search to track down a song, how many machines were really getting checked? How large a network could my own client see?

We instrumented a Gnutella client for visualization, and then started representing the network. We rapidly realized a couple of things: first, that new nodes were constantly appearing on the network; and second, that knowing where they were located was really interesting. The appearance of new nodes meant that we wanted to be able to change the visualization *stably*. There would always be new data pouring into the system, and it was important that users not be disoriented by changes taking place in the

visualization as new data came in. On the other hand, we did not want to pause, add data, and redraw: we wanted a system where new data would simply add itself to the diagram unobtrusively.

Because the Gnutella network used a peer-to-peer discovery protocol, it was often interesting to focus on a single node and its neighbors. Is this node connected to a central "supernode"? Is it conveying many requests? We wanted to be able to focus on any single node and its neighbors, and to be able to easily estimate the number of hops between nodes. This called for changing the *viewpoint* without changing the remainder of the layout.

Our tool was entitled GnuTellaVision, or GTV (Yee et al. 2001). We addressed these two needs with two different animation techniques. We based the visualization on a radial layout, both to reflect the way that data was changing—growing outward as we discovered more connections—and in order to facilitate estimation of the number of hops between the central node and others. A radial layout has the virtues of a well-defined center point and a series of layers that grow outward. As we discovered new nodes, we added them to rings corresponding to the number of hops from the starting node. When a new node arrived, we would simply move its neighbors over by a small amount (most nodes in the visualization do not move much). As the visualization ran, it updated with new data, animating constantly (Figure 19-4).

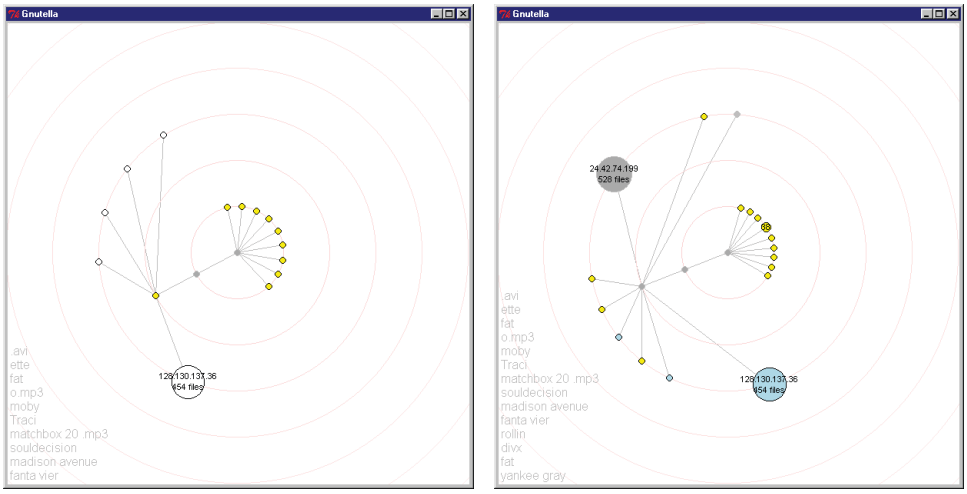

Figure 19-4. *GTV before (left) and after (right) several new nodes are discovered on the network—as nodes yield more information, their size and color can also change*

When a user wanted to examine a node, GTV recentered on the selection. In our first design, it did so in the most straightforward way possible: we computed a new radial layout and then moved nodes linearly from their previous locations to the new ones. This was very confusing, because nodes would cross trajectories getting from their old locations to the new ones. The first fix was to have nodes travel along polar coordinate

paths, and always clockwise. Thus, the nodes remained in the same space as the visualization was drawn, and moved smoothly to their new locations (Figure 19-5). Because GTV is oriented toward examining nodes that may be new to the user, and is constantly discovering novel information, it was important that this animation facilitate exploration by helping users track the node paths.

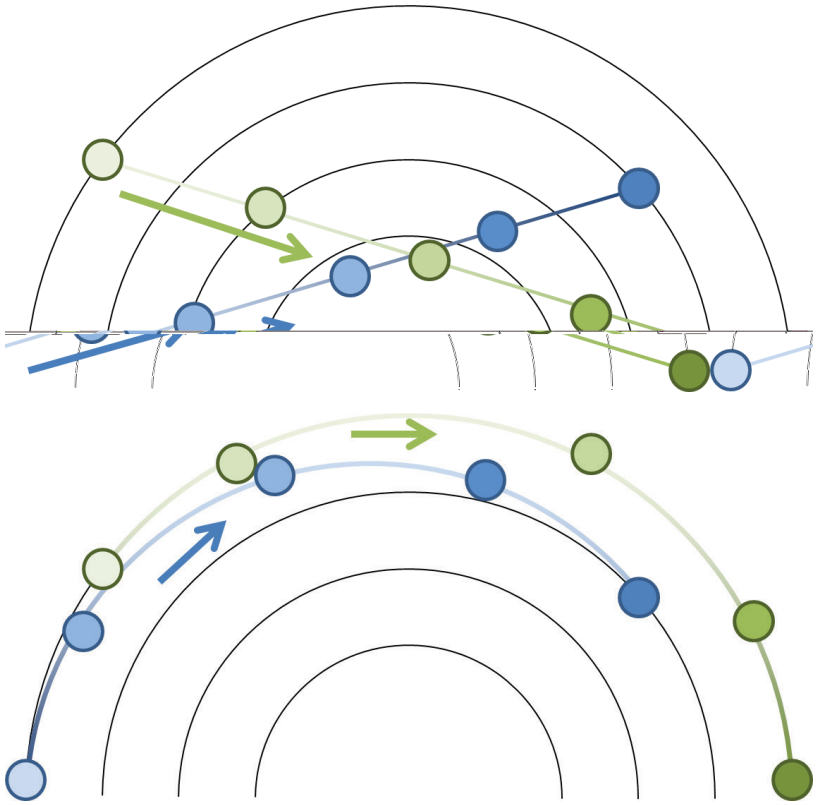

Figure 19-5. *Interpolation in rectangular coordinates (top) causes nodes to cross through each others' paths; interpolation in polar coordinates (bottom) makes for smooth motion*

A radial layout has several degrees of freedom: nodes can appear in any order around the radius, and any node can be at the top. When we did not constrain these degrees of freedom, nodes would sometimes travel from the bottom of the screen to the top. We wanted to ensure that nodes moved as little as possible, so we added a pair of constraints: nodes maintained, as much as they could, both the same relative *orientation* and *order*. Maintaining relative orientation means that the relative position of the edge from the old center to the new center is maintained. Maintaining relative order means that nodes' neighbors will remain in the same order around the rings. Both of these are illustrated in Figure 19-6.

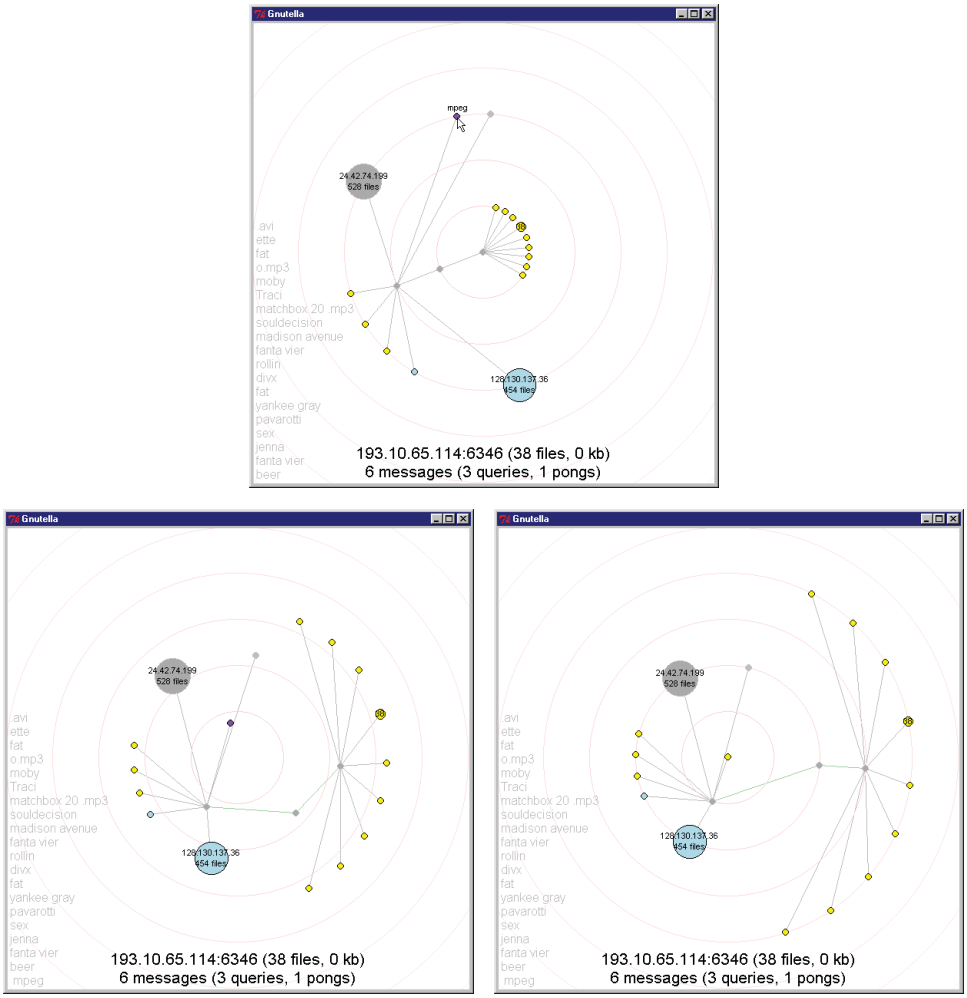

Figure 19-6. *Animated recentering: the purple highlighted node becomes the center, and other sets of nodes maintain their relative positions and orders (the large blue node stays below, and the set of small yellow nodes spreads along an outer ring)*

Last, we adapted the *ease-in, ease-out* motion from cartooning in order to help users see how the motion was about to happen.

This section demonstrated some useful principles that are worth articulating:

Compatibility

Choose a visualization that is compatible with animation. In GTV, the radial layout can be modified easily; new nodes can be located on the graph to minimize changes, and—like many tree representations—it is possible to recenter on different nodes.

Coordinate motion

Motion should occur in a meaningful coordinate space of the visualization. We want to help the users stay oriented within the visualization during the animation, so they can better predict and follow motion. In GTV, for instance, transforming through rectangular coordinates would be unpredictable and confusing; the radial coordinates, in contrast, mean that users can track the transition and the visualization retains its meaning.

Meaningful motion

Although animation is about moving items, unnecessary motion can be very confusing. In general, it is better to have fewer things move than more in a given transition. Constraining the degrees of freedom of the GTV animation allows the visualization to change as little as possible by keeping things in roughly the same position.

A Taxonomy of Animations

There are many sorts of change that might occur within a visualization. In the discussion of GapMinder, we talked about changes to data; in the example of GTV, we examined changes to both the data and the view. There are more types of transitions that one might wish to make in a visualization, though. The following is a list adapted from one assembled by Heer and Robertson (2007). Each type of transition is independent; it should be possible to change just the one element without changing any of the others. Many of these are applicable to both presentation and exploration of data:

Change the view

Pan over or zoom in on a fixed image, such as a map or a large data space.

Change the charting surface

On a plot, change the axes (e.g., change from linear to log scale). On a map, change from, for example, a Mercator projection to a globe.

Filter the data

Remove data points from the current view following a particular selection criterion.

Reorder the data

Change the order of points (e.g., alphabetize a series of columns).

Change the representation

Change from a bar chart to a pie chart; change the layout of a graph; change the colors of nodes.

Change the data

Move data forward through a time step, modify the data, or change the values portrayed (e.g., a bar chart might change from Profits to Losses). As discussed earlier, moving data through a time step is likely to be more useful for presentations.

These six types of transitions can describe most animations that might be made with data visualizations. Process visualizations would have a somewhat different taxonomy, as would scientific visualizations that convey flow (such as air over wings). Next, given this set of transitions, we will discuss some examples of how these animations might be managed.

Staging Animations with DynaVis

Two people exploring a dataset together on a single computer have a fundamental problem: only one of them gets the mouse. While it is perfectly intuitive for one of them to click "filter," the other user might not be able to track what has just happened. This sits at an interesting place between exploration and presentation: one of the major goals of the animation is to enable the second user to follow the leader by knowing what change the leader has just invoked; however, the leader may not know specifically what point he is about to make. Animation is plausibly a way to transition between multiple visualizations, allowing a second person—or an audience—to keep up. For the last several years, we have been experimenting with ways to show transitions of data and representations of well-known charts, such as scatterplots, bar charts, and even pie charts.

DynaVis, a framework for animated visualization, was our starting point. A summer internship visit by Jeff Heer, now a professor at Stanford, gave us a chance to work through a long list of possibilities. This discussion is outlined in more detail in his paper (Heer and Robertson 2007).

In DynaVis, each bar, dot, or line is represented as an object in 3D space, so we can move smoothly through all the transitions described in the preceding section. Many transformations are fairly clear: to filter a point from a scatterplot, for instance, the point just needs to fade away. There are several cases that are much more interesting to work through, though: those in which the type of representation needs to change, and those in which more than one change needs to happen at a time. When the representation is being changed, we try to follow several basic principles. Here are the first two:

Do one thing at a time

Ensure that the visualization does not entail making multiple simultaneous changes. This might mean *staging* the visualization, to ensure that each successive step is completed before the next one is started.

Preserve valid mappings

> At any given time during a step, ensure that the visualization is a meaningful one that represents a mapping from data to visualization. It would be invalid, for example, to rename the bars of a bar chart: the fundamental mapping is that each bar represents one *x*-axis value.

Figure 19-7 shows a first attempt at a transition from bar chart to pie chart. There are some positive aspects to the transition. For example, the bars do not move all at once, so the eye can follow movement fairly easily, and the bars maintain their identities and their values across the animation. While there are some issues with occlusion as the bars fly past each other, they move through a smooth trajectory so that it is reasonable to predict where they will end up. Finally, the animation is well staged: all the wedges are placed before they grow together into a full pie.

This visualization has a critical flaw, though. The length of the bar becomes the length of the pie wedge, so longer bars became longer wedges. However, longer bars will ultimately have to become fatter wedges in the final pie chart. That means that bars are becoming both fat and long, or both skinny and short. This, in turn, means that the visualization does not employ a constant rule (such as "number of pixels is proportionate to data value").

That leads us to the next principle:

Maintain the invariant

> While the previous rule referred to the relationship between data elements and the marks on the display, this rule refers to the relationship of the data values to the visualization. If the data values are not changing, the system should maintain those invariant values throughout the visualization. For example, if each bar's height is proportionate to the respective data point's value, the bars should remain the same height during the animation.

Figure 19-8 illustrates both of these principles in a more successful bar chart to pie chart animation. This chart shows a 1:1 correspondence between the drawn entity—the bar, the curved line, or the pie slice—and the underlying data. This assignment never changes: the bar furthest on the left ("A") becomes the leftmost pie slice (also "A"). The invariant is maintained by the lengths of the bars, which remain proportionate to the data values. While we do not illustrate it here, we follow similar principles in changing a bar chart into a line chart: the top-left corner of the bar represents the value, so as the bar shrinks into a line, that data point will remain rooted at the top-left corner of the bar.

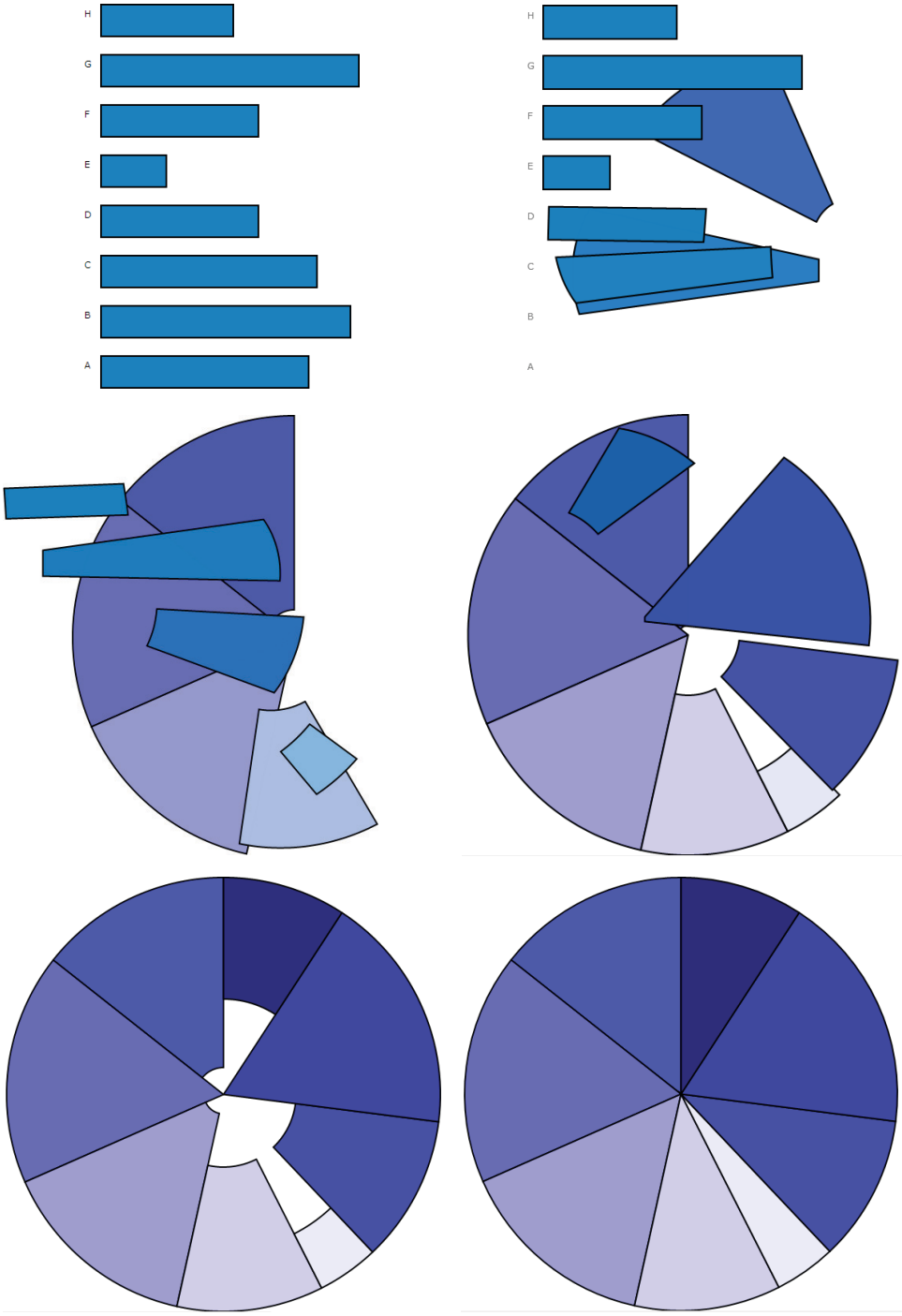

Figure 19-7. *Less successful bar chart to pie chart animation: long bars become long, fat wedges on the pie; short bars become short, skinny wedges; then all wedges grow to full length*

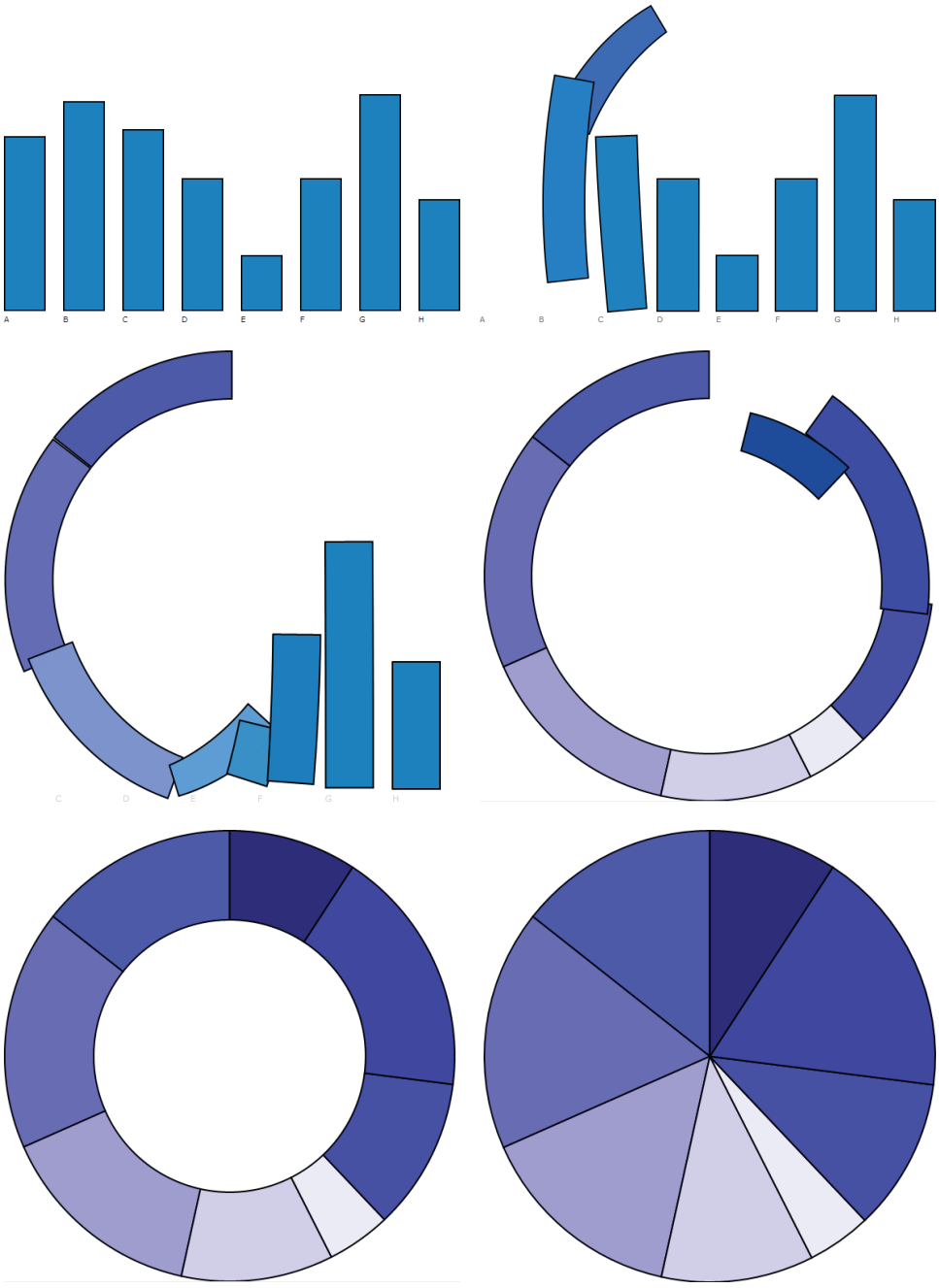

Figure 19-8. *Better bar chart to pie chart animation: the lengths of the bars are maintained as they are brought into the ring; the ring then fills to become a pie*

Another interesting case brings back the cartoon notion of *staging*. In GnuTellaVision we were able to recenter in a single motion, but in DynaVis it often makes more sense to break a transformation into two steps. For instance, in each of these examples, we ensure that we change only one thing at a time:

- To filter a dataset in a bar chart, we first remove bars we will not use, and then close ranks around them. To unfilter, we open space for the bars that will be added, and then grow the bars up.
- To grow or shrink a bar, such as when data changes, we may need to change the axes. Imagine growing a bar chart from the values (1,2,3,4,5) to (1,2,10,4,5)—the *y*-axis should certainly grow to accommodate the new value. If we grow the bar first, it will extend off the screen; therefore, we must change the axis before changing the bar.
- When sorting a selection of bars, sorting them at once could cause all bars to pass through the center at once. This is confusing: it is hard to figure out which bar is which. By staggering the bars slightly, so that they start moving a small amount of time apart, we found that the sort operation was much clearer.

Staging is not always appropriate, though. In Heer and Robertson's report on the project (2007), they found that some staged animations are more challenging to follow. In particular, when resizing segments of a donut or pie chart, it was difficult to monitor the changes as the pie turned to accommodate the new sizes. DynaVis attempted to stage this transition by extracting segments to either an external or an internal ring, adjusting their sizes, and then collapsing them back into place. While this made the changes much more visible, it also added a layer of potentially confusing action.

Heer and Robertson collected both qualitative results—how much users liked the animations—and quantitative ones—finding out which animations allowed users to answer questions most accurately. They found that users were able to answer questions about changes in values over time more easily with the animations than without; furthermore, the animations that were staged but required only one transition did substantially better than the animations that required many transitions.

Even with these caveats, though, it is clear that these sorts of dynamics could potentially help users understand transitions much more easily: compared to a presenter flipping through a series of charts, forcing the audience to reorient after each slide, a DynaVis-like framework might allow users to remain oriented thoughout the presentation.

Principles of Animation

There have been several attempts to formulate principles for animation. Tversky, Morrison, and Bétrancourt (2002) offer two general guidelines at the end of their article: that visualizations should maintain *congruence* and *apprehension*. The former

suggests that the marks on the screen must relate to the underlying data at all times. The latter suggests that the visualization should be easy to understand. The principles we have articulated fit into these categories. (Other, related guidelines have been suggested in Heer and Robertson's [2007] discussion of the DynaVis research, by Zongker and Salesin [2003] in their discussion of animation for slideshow presentations, and, with regard to graph drawing, by Freidrich and Eades [2002].)

The principles that we have discussed in this chapter are:

Staging
: It is disorienting to have too many things happen at once. If it is possible to change just one thing, do so. On the other hand, sometimes multiple changes need to happen at once; if so, they can be staged.

Compatibility
: A visualization that will be disrupted by animation will be difficult for users to track. For example, it is not disruptive to add another bar to a bar chart (the whole set can slide over), and it may not be disruptive to add another series to a bar chart. However, a squarified treemap is laid out greedily by size; growing a single rectangle will require every rectangle to move to a new location and will look confusing.

Necessary motion
: In particular, avoid unnecessary motion. This implies that we want to ensure that motion is significant—i.e., we should animate only what changes. In general, the image should always be understandable. As the DynaVis user tests showed, excess motion—even significant motion—can be confusing.

Meaningful motion
: The coordinate spaces and types of motion should remain meaningful. This also entails two points discussed earlier: *preserve valid mappings* and *maintain the invariant.*

Verifying that you've adhered to these principles can help you figure out whether an animation is headed in the right direction.

Conclusion: Animate or Not?

In this chapter, we have discussed the difference between presentation and exploration of data. We have also discussed the various layers of a visualization that might be altered, and some principles for making a visualization-safe animation.

So now you're staring at a visualization you're working on, and trying to decide whether to animate it or not. The question that this chapter has repeatedly asked is: what function does the animation serve? If it is meant to allow a user to smoothly

transition between views, then it is likely to be helpful. On the other hand, if the user is meant to compare the "before" to the "after," the animation is less likely to be of use.

Users want to understand why a change is happening, and what is changing. If everything on the screen is going to move around, perhaps it would be better to simply switch atomically to a new image; this might spare the user the difficulty of trying to track the differences. Finally, animations mean that it can be more difficult to print out visualizations. Individual frames should be meaningful, so that users can capture and share those images. Animation imposes a burden of complexity on the user, and that complexity should pay off.

Further Reading

Here are a few animated data visualization projects that have some relevance to this discussion, which you may want to explore further:

- Many researchers begin playing with zooming and panning as basic operations in a visualization with Pad++, a zoomable architecture for laying out data in large spaces (Bederson and Hollan 1994).
- Scatterdice (Elmqvist, Dragicevic, and Fekete 2008) explores a way to transition between scatterplots by rotating through the third dimension.
- Visualizations of tree data structures include ConeTrees (Card, Robertson, and Mackinlay 1991), CandidTree (Lee et al. 2007), and Polyarchy (Robertson et al. 2002). Researchers have explored animation with treemaps by zooming (distorting) the treemap (Blanch and Lecolinet 2007) and moving through 3D space (Bladh, Carr, and Kljun 2005).
- Graph layout is often animated as the layout progresses; in the last 10 years, the graph-drawing community has turned to considering ways to update graphs in response to underlying data. In addition to the work cited earlier (Friedrich and Eades 2002), we note GraphAEL (Erten et al. 2003).

Acknowledgments

I am grateful to Professor Jeffrey Heer of Stanford University, both for his valuable conversations on these topics when we shared an office and for his predigested versions of these concepts, produced in his 2007 Infovis paper (Heer and Robertson 2007) and his Stanford course notes. Jeff also contributed a chapter to *Beautiful Data*, the sister volume to this book, discussing his work with *sense.us*. My thanks also for feedback and ideas for this paper from my colleagues, Steven Drucker, Roland Fernandez, Petra Isenberg, and George Robertson.

References

Bederson, B.B., and J.D. Hollan. 1994. "Pad++: A zooming graphical interface for exploring alternate interface physics." In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*. New York: ACM Press.

Bladh, Thomas, David A. Carr, and Matjaz Kljun. 2005. "The effect of animated transitions on user navigation in 3D tree-maps." In *Proceedings of the Ninth International Conference on Information Visualization*. Washington, DC: IEEE Computer Society.

Blanch, Renaud, and Eric Lecolinet. 2007. "Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques." *IEEE Transactions on Visualization and Computer Graphics* 13, no. 6: 1248–1253.

Card, Stuart K., George G. Robertson, and Jock D. Mackinlay. 1991. "The information visualizer, an information workspace." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM Press.

Cavanagh, Patrick, and George Alvarez. 2005. "Tracking multiple targets with multifocal attention." *TICS* 9: 349–354.

Chang, Bay-Wei, and David Ungar. 1993. "Animation: From cartoons to the user interface." In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*. New York: ACM Press.

Elmqvist, N., P. Dragicevic, and J.-D. Fekete. 2008. "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation." *IEEE Transactions on Visualization and Computer Graphics* 14, no. 6: 1141–1148.

Erten, C., P.J. Harding, S.G. Kobourov, K. Wampler, and G. Yee. 2003. "GraphAEL: Graph animations with evolving layouts." In *Proceedings of the 11th International Symposium on Graph Drawing*. Springer-Verlag.

Fisher, Danyel A. 2007. "Hotmap: Looking at geographic attention." *IEEE Transactions on Visualization and Computer Graphics* 13, no. 6: 1184–1191.

Friedrich, C., and P. Eades. 2002. "Graph drawing in motion." *Journal of Graph Algorithms and Applications* 6, no. 3: 353–370.

Heer, Jeffrey, and George G. Robertson. 2007. "Animated transitions in statistical data graphics." *IEEE Transactions on Visualization and Computer Graphics* 13, no. 6: 1240–1247.

Hundhausen, Christopher D., Sarah A. Douglas, and John T. Stasko. 2002. "A meta-study of algorithm visualization effectiveness." *Journal of Visual Languages & Computing* 13, no. 3: 259–290.

Johnson, Ollie, and Frank Thomas. 1987. *The Illusion of Life.* New York: Disney Editions.

Lasseter, John. 1987. "Principles of traditional animation applied to 3D computer animation." In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. New York: ACM Press.

Lee, Bongshin, George G. Robertson, Mary Czerwinski, and Cynthia Sims Parr. 2007. "CandidTree: Visualizing structural uncertainty in similar hierarchies." *Information Visualization* 6: 233–246.

Michotte, A. 1963. *The Perception of Causality*. Oxford: Basic Books.

Robertson, George, Kim Cameron, Mary Czerwinski, and Daniel Robbins. 2002. "Polyarchy visualization: Visualizing multiple intersecting hierarchies." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM Press.

Robertson, George, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. 2008. "Effectiveness of animation in trend visualization." *IEEE Transactions on Visualization and Computer Graphics* 14, no. 6: 1325–1332.

Tversky, Barbara, Julie B. Morrison, and Mireille Bétrancourt. 2002. "Animation: Can it facilitate?" *International Journal of Human-Computer Studies* 57: 247–262.

Yee, Ka-Ping, Danyel Fisher, Rachna Dhamija, and Marti A. Hearst. 2001. "Animated exploration of dynamic graphs with radial layout." In *Proceedings of the IEEE Symposium on Information Visualization*. Washington, DC: IEEE Computer Society.

Zongker, Douglas E., and David H. Salesin. 2003. "On creating animated presentations." In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. New York: ACM Press.