Animation:
From Cartoons to the User Interface

Bay-Wei Chang

Computer Systems Laboratory
Stanford University
Stanford, CA 94305

bay@self.stanford.edu

David Ungar

Sun Microsystems Laboratories, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

david.ungar@sun.com

You must learn to respect that golden atom, that single frame of action, that 1/24th of a second, because the difference between lightning and the lightning bug may hinge on that single frame.

— Chuck Jones [10]

ABSTRACT

User interfaces are often based on static presentations, a model ill suited for conveying change. Consequently, events on the screen frequently startle and confuse users. Cartoon animation, in contrast, is exceedingly successful at engaging its audience; even the most bizarre events are easily comprehended. The Self user interface has served as a testbed for the application of cartoon animation techniques as a means of making the interface easier to understand and more pleasant to use. Attention to timing and transient detail allows Self objects to move solidly. Use of cartoon-style motion blur allows Self objects to move quickly and still maintain their comprehensibility. Self objects arrive and depart smoothly, without sudden materializations and disappearances, and they rise to the front of overlapping objects smoothly through the use of dissolve. Anticipating motion with a small contrary motion and pacing the middle of transitions faster than the endpoints results in smoother and clearer movements. Despite the differences between user interfaces and cartoons—cartoons are frivolous, passive entertainment and user interfaces are serious, interactive tools—cartoon animation has much to lend to user interfaces to realize both affective and cognitive benefits.

KEYWORDS: animation, user interfaces, cartoons, motion blur, Self

1 INTRODUCTION

User interfaces are often based on static presentations—a series of displays each showing a new state of the system. Typically, there is much design that goes into the details of these tableaux, but less thought is given to the transitions between them. Visual changes in the user interface are sudden and often unexpected, surprising users and forcing them to mentally step away from their task in order to grapple with understanding what is happening in the interface itself.

When the user cannot visually track the changes occurring in the interface, the causal connection between the old state of the screen and the new state of the screen is not immediately clear. How are the objects now on the screen related to the ones which were there a moment ago? Are they the same objects, or have they been replaced by different objects? What changes are directly related to the user's actions, and which are incidental? To be able to efficiently and reliably interpret what has happened when the screen changes state, the user must be prepared with an expectation of what the screen will look like after the action. In the case of most interactions in unanimated interfaces, this expectation can only come by experience; little in the interface or the action gives the user a clue about what will happen, what is happening, or what just happened.

For example, the Microsoft Windows interface [15] expands an icon to a window by eliminating the icon and drawing the window in the next instant. In this case the first static presentation is the screen with the icon; the next is the screen with an expanded window. Much of the screen changes suddenly and without indication of the relationship between the old state and the new state. Current pop-up menus suffer from the same problem—one instant there is nothing there; the next instant a menu obscures part of the display.

Moving objects from one location to another is yet another example. Most current systems let the user move an outline of the object, and then, when the user is finished the move, the screen suddenly changes in two places: the object in the old location vanishes and the object appears in the new location. Sudden change, flash of the screen, no hint how the two states are related: the user must compare the current state and the preceding state and deduce the connection.

Users overcome obstacles like these by experience. The first few encounters are the worst; eventually users learn the behavior of the interface and come to interact with it efficiently. Yet while some of the cognitive load of

interpreting changes in the interface may be reduced through repeated exposure, it is likely that some burden remains. For example, a user may know that clicking an icon will open a window, but the sudden change to the screen when the window opens may still require a moment of assessment.

Cartoons, in contrast, excel at providing enough information for the audience to follow the action without ever being startled and confused by puzzling behavior. Cartoon characters do not simply disappear from one place and appear in another; they appear to move solidly and continuously. Cartoon objects do not flash suddenly into different sizes and shapes; they appear to grow and deform smoothly. Animation provides the visual cues necessary to understand what is happening before, during, and after the action. Unlike user interfaces which burden the user with the responsibility of relying on experience and deductive ability to interpret changes, cartoon animation leverages off of human experience of how objects change and move smoothly in the real world.

The experimental user interface for the programming language Self uses techniques drawn from cartoon animation to replace sudden changes with smooth transitions, offloading some of the cognitive burden of interpreting the change to the perceptual system. For example, a small object grows continuously to a large, expanded object. A menu transforms smoothly from the menu button that is clicked on to the full menu. Objects move solidly from one location to another, maintaining the illusion of solid movement even if they must cross great distances in a single frame. These and other, more subtle, applications of animation to the interface work together to make changes on the screen smooth and clear.

Bringing this kind of animation to the user interface has both cognitive and affective benefits. By offloading interpretation of changes to the perceptual system, animation allows the user to continue thinking about the task domain, with no need to shift contexts to the interface domain. By eliminating sudden visual changes, animation lessens the chance that the user is surprised, thus reducing his uneasiness. So employing animation not only aids the user in understanding the events in the user interface, but also makes the user's experience of the interface more pleasant and comfortable.

Animation has been used in other work to illuminate change, for example, in data visualization [7, 14, 17], algorithm animation and program visualization [3, 5, 8, 21] and animated demonstration and help [2, 16, 22, 23]. These uses of animation clarify data, or programs, or information on how to use the interface, but not the operating of the interface itself. Cartoon-style animation as used in the Self interface is animation of *transition* and *feedback*, to borrow from the informal taxonomy of Baecker and Small [1]. The Alternate Reality Kit (ARK) [19, 20], an environment for building interactive simulations, widely uses animation to show the behavior of objects in the interface. ARK blurs the distinction between data and interface by unifying both simulation objects and interface objects as concrete objects. The Information Visualizer framework [6, 14, 17] also

makes data more concrete, using three-dimensional interactive graphics to further heighten the concreteness. The cone tree and perspective wall techniques implemented in the Information Visualizer use fast 3D animation to aid the user in tracking objects as they change position. This animation, however, is performed straightforwardly, without using cartoon-inspired techniques that could make the system even more alive and engaging.

2 APPLYING PRINCIPLES OF CARTOON ANIMATION TO THE USER INTERFACE

Over the years animators have developed a canon of principles which capture much of the essence of the craft of animation. This section looks at some of those principles of animation in their original context of cartoons, and also in the context of user interfaces—why the principles are relevant and how they have been applied in an experimental user interface for the programming language Self [26].

We discuss the principles of animation in three groups: solidity, exaggeration, and reinforcement. The principles of animation mentioned in this section are drawn from the excellent treatise on cartoon animation, *Disney Animation: The Illusion of Life*, by Disney animators Frank Thomas and Ollie Johnston [24].

2.1 Solidity

Characters and objects in cartoons are *solid*. They move about as if they are three-dimensional, real things; they react to external forces as if they have mass and are susceptible to inertia. From beginning to end, they are tangible things; you could reach out and grab them if only you could get into the cartoon. This solidity is achieved not merely by drawing objects as filled-in areas, but by infusing their motion with a sense of weight and balance. Animation principles such as slow in and slow out, follow through, and arcs, all discussed later, contribute in keeping the motion of objects realistic and reinforcing the sense that they are solid.

Making an object in the user interface solid gives it an individual identity as a separate, interactable thing. Solidity elevates an element from a mere picture on the screen to a tangible entity that is real enough to possess its own behavior. In the Self user interface, objects are drawn as three-dimensional boxes, and they move solidly and continuously as the user drags them with the mouse. In contrast, many current interfaces only allow the user to drag the outline of the object. This optimization destroys the illusion of solidity; it forces on the user the fact that he deals only with pictures of the object: a rough outline that can be dragged and a more detailed rendering that appears first in one place and then suddenly, when the mouse button is released, in another. By maintaining their solidity as they are dragged, Self objects subtly reinforce the illusion that they are not mere pictures on the screen, but are individual, manipulable things.

Moving an object solidly is not enough, however. In order to maintain the illusion, the movement must be at a high enough frame rate to keep up with the user's cursor. A drop in performance can result in the object lagging behind the cursor and jumping from place to place, a distracting and
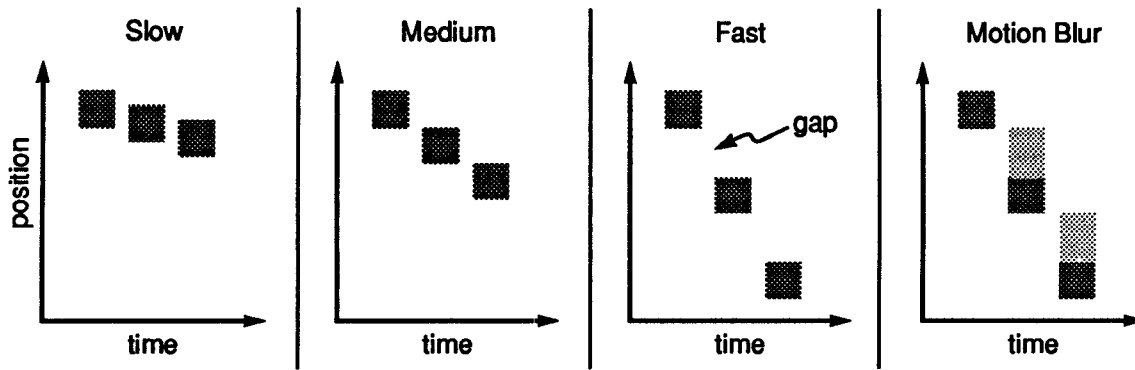
Figure 1. At slow and medium speeds, objects overlap with their previous images. At fast speeds, however, there is a gap between the object and its previous image, making the movement look disconnected and confusing. Motion blur fills in that gap, aiding the eye in tracking the motion of the object. (Higher frame rates reduce, but do not eliminate, the need for motion blur—a fast enough moving object will still leave a gap.)

perhaps disturbing sight. The Self user interface has been tuned to maintain a frame rate of 20 to 30 frames per second during interactive object movement. Furthermore, the inner loop of the movement code has been carefully crafted to prevent flicker, which would undermine the illusion of solidity.

2.1.1 Solidity: Motion blur

But frequent, flicker-free frames are still not enough: a sudden jerk of the mouse can still cause the object to leap from one side of the screen to the other in a single frame, without passing through the intervening landscape. Although the object is indeed keeping up with the cursor, the large movement across the screen no longer seems like the movement of one object. Because of persistence of

vision, the eye is presented with the short-lived image of two objects, as if one had flashed into being and then the other had winked out of existence. The solidity, and even the very identity of the object breaks down. (This effect is also known as temporal aliasing.)

Cartoon animators confronted this problem long ago, and developed a rule of thumb: if an object moves more than half its size (alternatively, more than its full size) between any two frames, *motion blur* must be added. Motion blur fills in the gaps between the old and new position of the object, appeasing the eye into seeing continuous motion from one point to another (see Figure 1). Although related to realistic temporal anti-aliasing, cartoon motion blur attempts to remedy the same flaws more cheaply—for example, a smear of color. Cartoons suggest that even loose
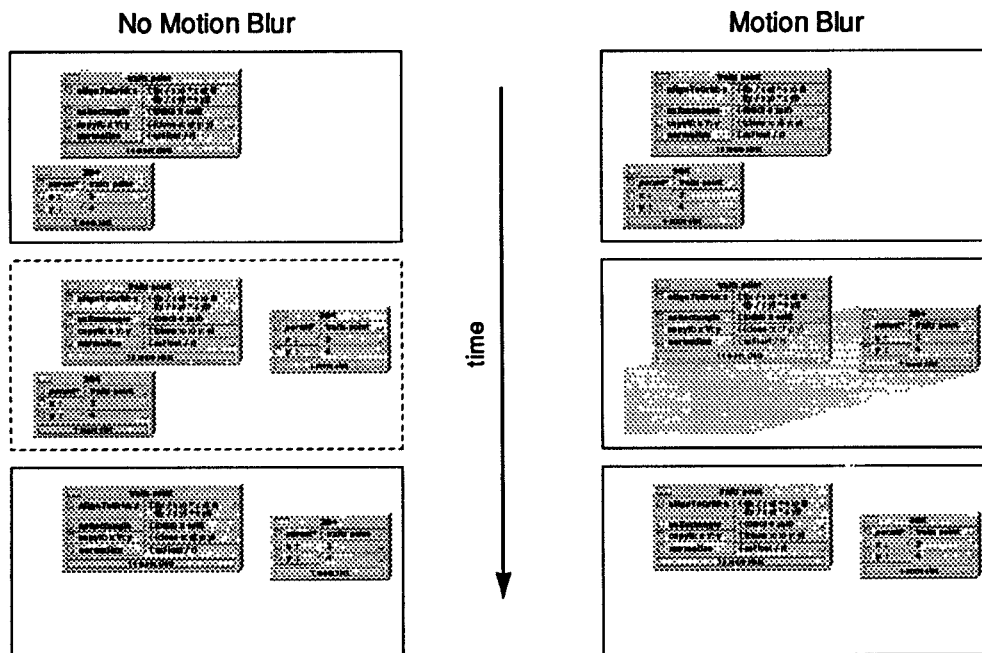

Figure 2. When objects are moved suddenly from one position to another, it can seem as if there are two instances of the object on the screen at the same time. The eye sees something like the middle frame of the "no motion blur" figure, even though such a frame doesn't actually ever appear on the screen. Motion blur reduces this effect, and gives a visual indication of the travel of the object, so that it is easy to see which object moved where.

approximations to true temporal anti-aliasing can contribute great benefits. The Road Runner zips across the width of the screen in a handful of frames, but the Road Runner is not drawn at its appropriate location in each of those frames. Instead, a streak of colors (the motion blur) follows the Road Runner, and sometimes entirely supplants the Road Runner. This same effect is used for running legs, falling objects, and anything in which the action moves the object more than part of its width, that is, any movement that would look disconnected and wrong.

Careful study of cartoons reveals that animators use at least two different kinds of motion blur. The most realistic form is a translucent streak, typically painted on with a drybrush technique. The partial transparency of the resulting streak mimics the physical effect in reality and produces a very realistic effect. The other technique used for motion blur involves painting more than one pose in a single frame. For example, a single frame of an arm swinging rapidly back and forth might be rendered with several arms in different positions. This technique increases the effective frame rate and is most convincing with oscillatory motion, where the endpoints are more strongly perceived. (The animation of icons expanding to windows in the Macintosh Finder [27] uses this stutter motion blur.)

Just as in cartoons, the Self interface uses motion blur to connect the gaps that occur between fast moving objects. The motion blur is implemented as a trail the color and width of the object, connecting the object's old location with its new location (see Figure 2). The trail is stippled to create the illusion of translucency. The blur makes motion seem faster, smoother, and helps the user keep track of the moving object.

The effectiveness of motion blur demonstrates that showing motion is not just showing a series of still pictures. A frame
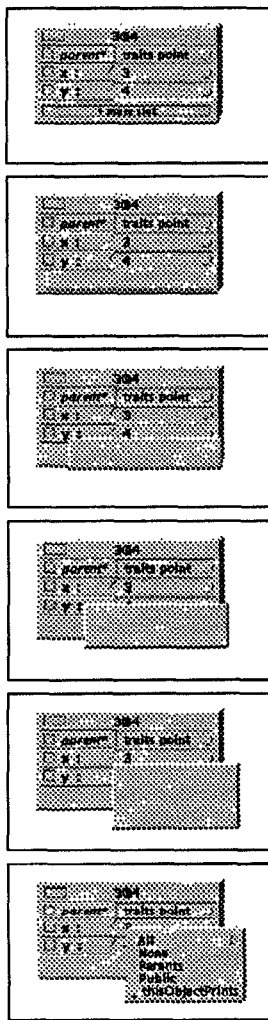
Figure 3. Upon a click, a menu button transforms itself from a button into the full menu. After a selection has been made, it shrinks back down to a button. (In this and other figures, only a few frames of the actual animation is shown.)
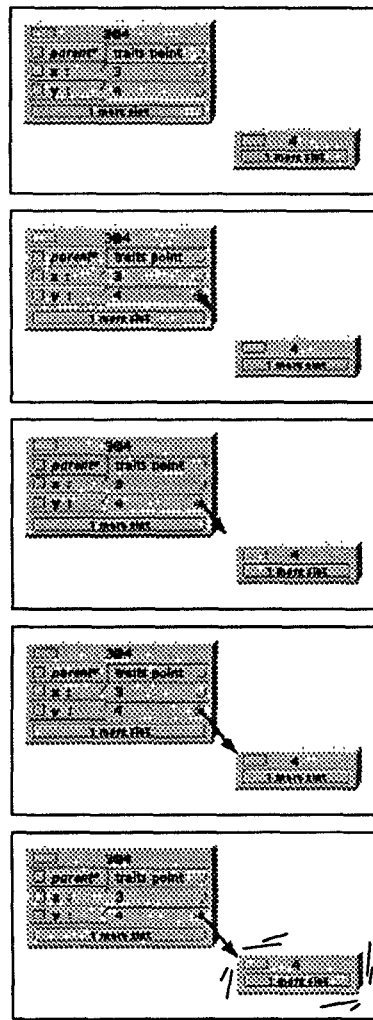
Figure 4. Arrows grow from their tail to hit their target. The target reacts to the contact with a small wiggling jolt (here suggested by a few lines), as discussed in section 2.3.3. Arrows will also shrink back down into their tail.
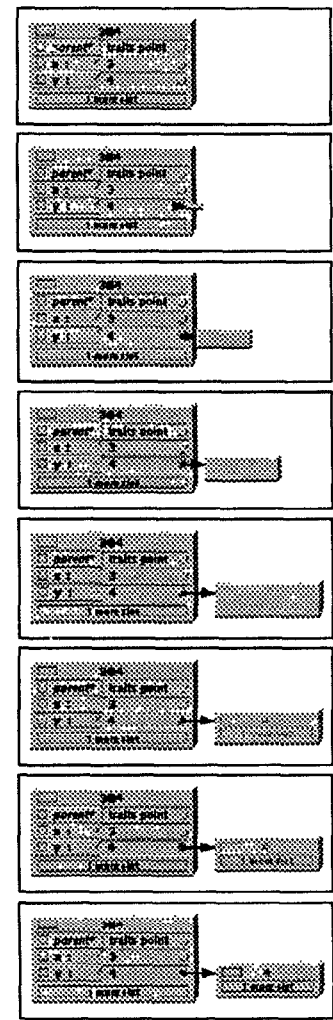
Figure 5. Objects grow from a point to the full size object; any connecting arrow will grow smoothly along with the object. Currently, text does not grow along with the object; instead, the text fades in smoothly on the fully grown object.

from the middle of an animated motion sequence would not look like a snapshot of that instant in reality: it would not show the object frozen in time, sitting pristinely in between its starting point and its destination. Instead, it would show the object with a blur trail, which has no physical existence. Its claim in the real world comes from the workings of the human visual system. If one models the visual system as having a simple finite rise and fall time, one would expect a short leading trail of blur, growing smoothly less transparent, and a longer following trail, tailing off into transparency. Cartoon animators do include both leading and trailing blurs, but even rendering a crude, trailing-only, fixed-transparency streak is enough to provide the visual system with the important subliminal cues that are essential to maintaining the illusion of fast motion.

2.1.2 Solidity: Arrivals and departures

In addition to moving continuously, real objects arrive and depart continuously without materializing from or disappearing into thin air. Rather than simply appearing and disappearing, which are sudden and potentially confusing changes, Self objects maintain the illusion of reality by smoothly easing on and off the screen. Three variations have appeared in the interface for arriving objects: objects fly in from offscreen, objects grow from a point to their full size, and objects dissolve onto the screen. Objects exit in the opposite way that they entered, for example, flying off the edge of the screen or fading into nothing. In each of these cases, the action, though quick, is smooth and continuous, giving the user the visual cues necessary to understand what is happening.

Animation to maintain solidity is ubiquitous in the Self world. Arrows that point from one object to another grow smoothly from their tail to their destination, and retract smoothly as well. Pop-up menus are really grow-up menus—the menu button transforms into a menu in its full open state, and shrink back down to a button after a selection is made (see Figures 3, 4, and 5).

A final example of solidity in the Self world is how objects raise and lower themselves to the top or bottom of the layers of overlapping objects. In typical systems with overlapping elements, objects change layers suddenly and without any indication of the action other than the final, changed state. In the Self interface, an object can *dissolve* through other objects to show that it is rising to the top of the pile or sinking to the bottom (see Figure 6). Paradoxically, dissolving through other objects rather than suddenly changing layers seems to increase the sense that the object is solid, even though solid objects in the real world do no such thing.

2.2 Exaggeration

Cartoon animation does not merely mimic reality. Like all dramatic mediums, it takes liberties with what is strictly realistic in order to more effectively convey its message. Adhering to what is possible in the physical world is not only limiting, but also less effective at achieving "realism." Paradoxically, only by exaggeration do cartoons achieve more realism. The dwarves in Disney's *Snow White*, for example, are highly stylized—their faces and bodies are
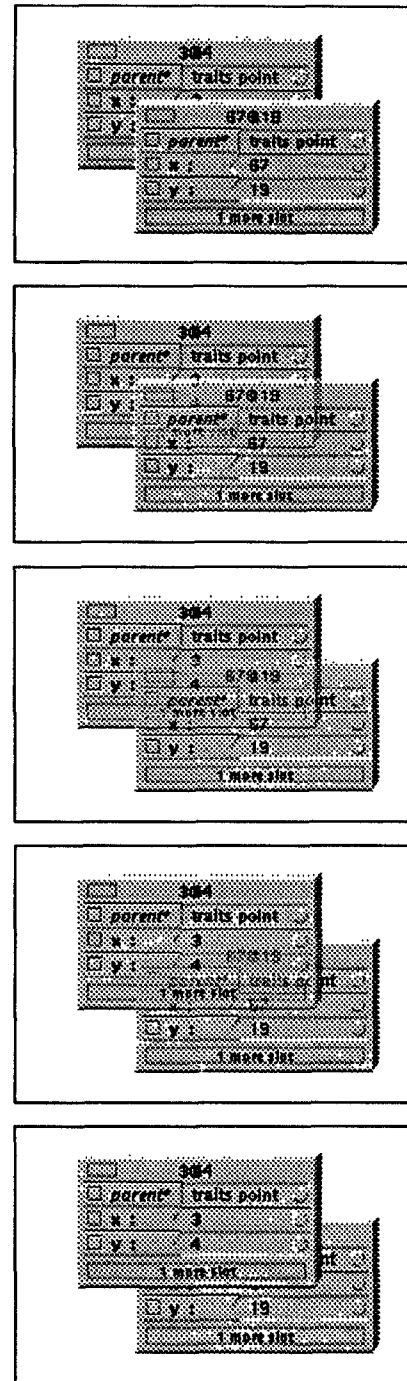

Figure 6. Objects dissolve through one another as they change their layering.

drawn with oversize, weighty features, their movements are large and exaggerated. In contrast, Snow White is drawn with realistic proportions (her nose, to mention just one feature, many times smaller than any of the dwarf's), and her movements just those a real person would have, much smaller and more measured than that of the dwarves'. Yet Snow White, as realistically as she is rendered, seems bland and wooden in comparison to the expressive and engaging dwarves. The dwarves simply seem more realistic.

Exaggeration increases the prominence of features deemed significant by the animator, whether these features are physical characteristics, qualities of action, or extremities of situation. By increasing the salience of certain aspects of the world, the animator gives the audience footholds from which to better interpret the nature of the character, action, or situation. For example, each of the seven dwarves in *Snow White* has a defining characteristic, often echoed by his name (Dopey, Grumpy, Sneezy). These characteristics would come out as strongly even without the name serving as an additional clue: each dwarf's physical features, movements, and actions all are broadly exaggerated to emphasize his designated trait.

In addition, exaggeration makes salient particulars that would likely otherwise be overlooked by the audience. Unlike in the real world, in which we can only point our fingers to something, in a cartoon the animator can cause that something to call more attention to itself, simply by exaggerating its appearance or its movement. Important events that might pass unnoticed because they exist so fleetingly can be exaggerated to make them more noticeable.

Like cartoons, the user interface must both give the user footholds from which to better interpret the nature of the objects and the situation in the user interface world, and also make salient particulars that might otherwise be overlooked by the user. Employing exaggeration to the behavior of user interface objects makes that behavior more understandable, more "realistic," and thus makes the user interface more engaging.

2.2.1 Exaggeration: Anticipation

Anticipation is a way of exaggerating a preliminary action in order to give the audience a cue about the main action to follow. When the action does occur, the audience is prepared and is ready to follow that action, without having to deal with being surprised by the occurrence itself. When an object in the Self interface is about to move (due to any action other than the user grabbing the object to move it directly), it anticipates that move with small, quick contrary movement (see Figure 7). Like the Coyote in the Road Runner cartoons, springing back onto its rear leg before dashing off after the Road Runner, this anticipatory move draws the user's eye to it, preparing the user to perceptually, as well as cognitively, follow the ensuing action.

Exaggeration techniques like anticipation (and some uses of *follow through*, discussed later) do not obey the laws of physics as we know them in the real world. As unrealistic as these purely theatrical techniques are, however, they are instrumental in keeping the user engaged with the interface. They are continual providers of hints to the user, cues to interpreting the succession of states in the user interface. Without such cues, the notion of the interface as a world that encompasses the user breaks down, and its existence noses itself into the user's awareness. Every element in the interface must be tuned to reinforce the illusion of an artificial reality.
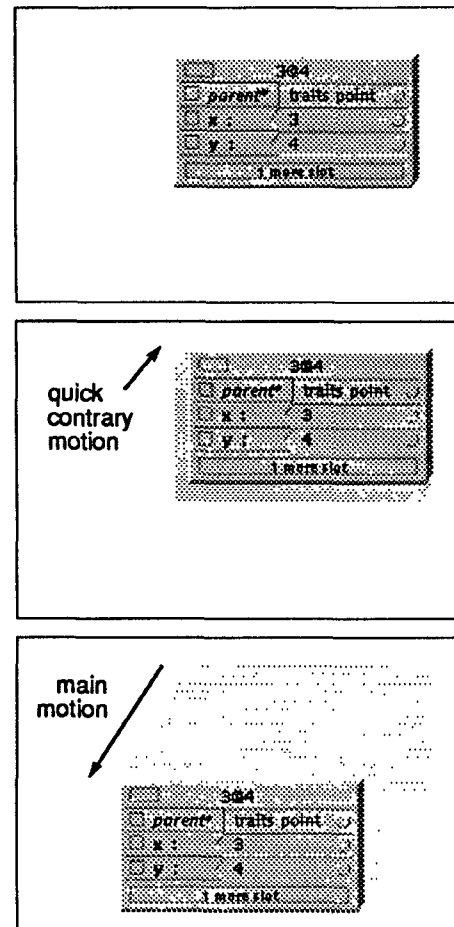

Figure 7. Objects anticipate major actions with a quick contrary motion that draws the user eye to the object in preparation for the main motion to come.

2.3 Reinforcement

All of the animation techniques contribute to the reinforcement of the illusion of reality. While some are quite drastic effects—attentive watching easily reveals the blur of colors that represent the Road Runner streaking across the screen—in effect they are subliminal. The audience is not consciously aware of these effects. Rather, the audience is aware of the feeling of realism in the action. They are not jarred out of the world of the cartoon by confusion about what's going on, because all the details that conspire to make the illusion have been attended to. The techniques described in this section, by reinforcing the illusion of reality, work to keep the audience engaged with the cartoon, or the user engaged with the interface.

2.3.1 Reinforcement: Slow in and slow out

Cartoon characters, as already mentioned, move solidly. However, this motion is not simply composed of drawings equally spaced in space and time. To capture the feeling of movement in the real world, animators draw action with *slow in and slow out*: characters move out of a pose slowly, then quickly during the bulk of the entire movement, and then slowly into the ending pose. This helps give a feeling
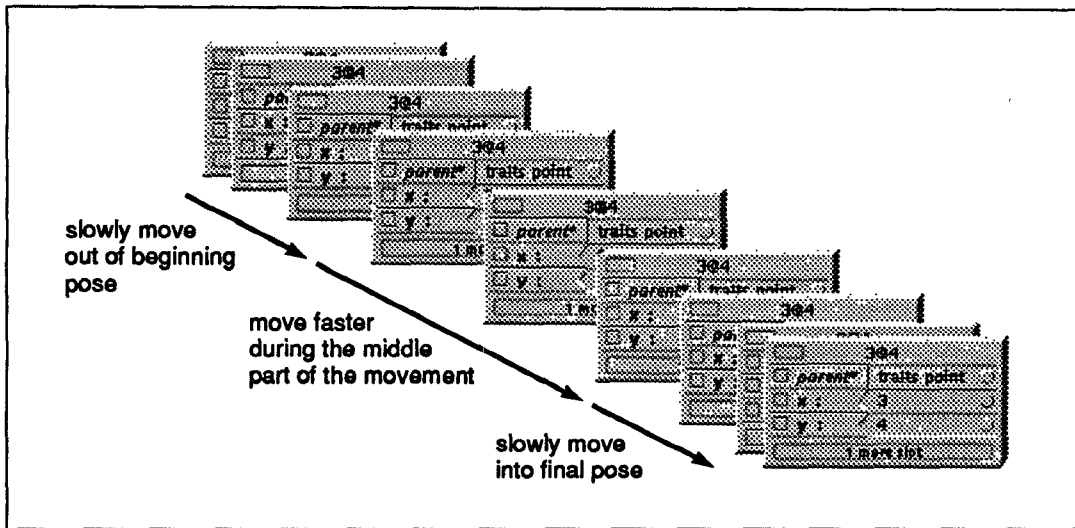
Figure 8. Objects ease out of their beginning poses and ease into their final poses. Although these motions are slower than that during the main portion of the movement, they are still quite fast.

of weight to the character and physicality to the movement. Slow in and slow out also contribute to minimizing surprise, slow in preparing the audience for faster movement to come and slow out preparing for an end to the movement. Furthermore, slow in and slow out satisfy these purposes while being a subtle and quickly passing effect. Yet without it, animated movement feels artificial and dead. Small details like slow in and slow out can have large effects on the feel of the animation.

Slow in and slow out is ubiquitous in the animation in the Self world. All movement uses it: boxes and menus growing or shrinking; arrows growing and shrinking; boxes entering from offscreen and exiting offscreen (see Figure 8). Even the dissolves used to dissolve objects or fade-in text uses slow in and slow out.

2.3.2 Reinforcement: Arcs

Another principle used for its subliminal effect is the principle of *arcs*. Rather than move in straight lines, objects move along gentle curves when they are moving non-
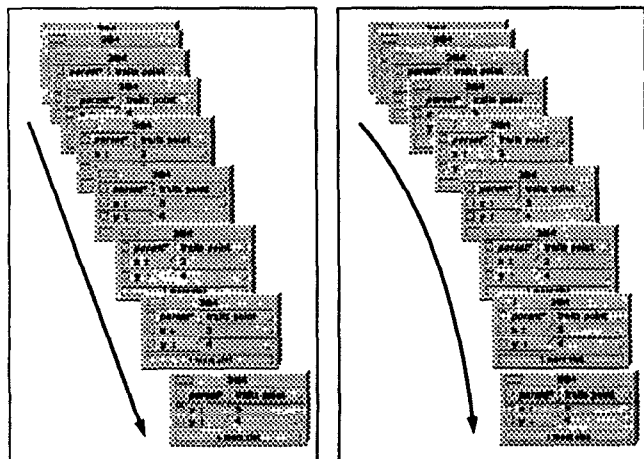

Figure 9. When objects travel under their own power (non-interactively), they move in arcs rather than straight lines.

interactively (see Figure 9). Like objects in cartoons, Self objects are meant to be existing in some physical world, and like objects in the real world, ought to move along lifelike arcs. The arc flavors the movement, giving it a livelier and more appealing character than a simple linear path would.

2.3.3 Reinforcement: Follow through

A final example of reinforcing principle is *follow through*. Objects in the real world do not come to sudden stops, all of the object coming to a standstill at once. Animators made this observation and made sure that objects coming to a stop in the cartoon would also have motion that continues after the main motion is completed. This could include parts of the body, clothing, or the entire ending pose slowly becoming slightly more extreme (known as "the moving hold" [24]). Follow through is also a technique of exaggeration—emphasizing some motions at the end of a movement to make the overall action clearer. The Self interface employs the principle of follow through in its use of wiggling objects. Objects coming to a stop will wiggle at the end of their motion, as if reacting to a small spring at the end of their travel (see Figure 10). The wiggle flavors the motion, giving the object a more distinct character and enhancing the illusion that it is solid. The wiggling is also used when an arrow grows from one object to hit another object. The target object reacts to the arrow's collision by vibrating back and forth, giving the sense that the arrow has weight, and that the object is situated in the world, not merely a picture drawn on the background (see Figure 4).

Building an illusion is a fragile affair; details of the animation have disproportionately large effect on the overall feel of the world. While such subtle effects as slow in and slow out, arcs, and follow through could be left out of the animation without destroying the sense of a somewhat concrete world, their addition yields a world much livelier and more realistic. The overall effect is a more convincing reality, one more likely to capture and retain the engagement of the user.

Summary: Animation principles in cartoons and Self

	Principle	Examples from cartoons	Examples from the Self interface
Solidity	solid drawing	• parts of Snow White's dwarves may squash and stretch, but always maintain their connectedness and weight	• objects move solidly • objects enter screen by travelling from offscreen or growing from a point • menus transform smoothly from a button to an open menu • arrows grow and shrink smoothly • transfer of momentum as objects respond to being hit by an arrow
	motion blur	• Road Runner is a blue and red streak	• stippled region connects old and new locations of a moving object
	dissolves	n/a	• objects dissolve through one another when changing layering
Exaggeration	anticipation	• Coyote rears back onto back leg before dashing after Road Runner	• objects preface forward movement with small, quick contrary movement
	follow through	• Road Runner vibrates for an instant after a quick stop	• objects come to a stop and vibrate into place • objects wiggle when hit by an arrow
Reinforcement	slow in and slow out	• Coyote springs up from ground, with fastest movement at center of the arc	• objects move with slow in and slow out • objects and arrows grow and shrink with slow in and slow out • objects dissolve through other objects with slow in and slow out • text fades in onto an object with slow in and slow out
	arcs	• dwarves' limbs move in an arc • Coyote springs up along an arc	• objects travel along gentle curves when they are moving non-interactively
	follow through	• Road Runner vibrates for an instant after a quick stop	• objects do not come to a sudden standstill, but vibrate at end of motion

3 WHY CARTOON-STYLE ANIMATION?

But why employ cartoon-style animation in the user interface? Why not simply use straight, strictly realistic, animation? Three characteristics of cartoon animation explain why it can so effectively inform user interface design: its theatrical basis, the engagement of its illusion, and the nature of its artistic medium.
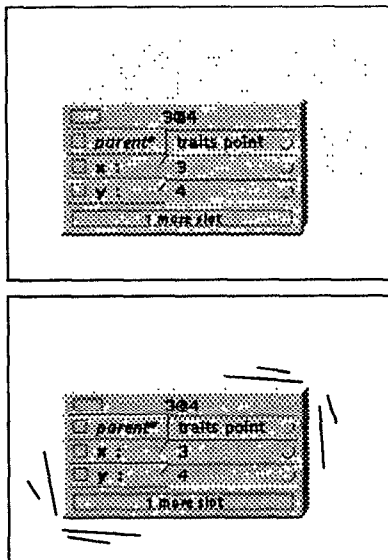

Figure 10. When objects come to a stop after moving on their own, they exhibit follow through in the form of wiggling back and forth quickly. This is just suggested by the "wiggle lines" in the figure—in actuality, the object moves back and forth, with motion blur.

First, cartoons are theatrical. They have license, and even mandate, to go beyond literal portrayal of the real world in order to convey their message. So, actions and reactions are exaggerated, situations are staged, all of the energy of movement and depiction is calculated to best telegraph the animator's point. User interfaces have the same need to communicate clearly and concisely, and theatricality can contribute to this goal. (Laurel [12, 13] has explored related issues of theater and the user interface; her emphasis is on the treating of the user interface as a two-way theatrical experience between the user and the computer. Tognazzini [25] notes correspondences between the theatrical techniques used in stage magic and that of user interface design.)

Second, cartoons are engaging. Cartoons create an illusory world, but effectively absorb the audience into that world. The mechanics of the cartoon (brightly colored, outlined shapes in front of largely static backgrounds) are quickly forgotten, and do not reappear in the audience's awareness. The animation techniques employed in the cartoon make the illusion so complete that even the wackiest events are easy to grasp by the audience. The audience is never jolted out of the cartoon world by the need to figure out what is going on. User interfaces can benefit by becoming more engaging, fully absorbing the user into its world so that her attention may be devoted entirely to the task. Users do not feel as if they are manipulating an interface, but rather as if they are in *direct engagement* with the task-domain objects [9].

Finally, the medium of cartoon animation is, in important ways, remarkably like that of graphics on the computer. The craft of animation is turning a succession of still drawings into a dynamic, lively image; animators were the first to

have that total control over each pixel in space at each instant in time. Computer animation has benefited from the techniques developed by traditional animators; Lasseter [11] discusses practical aspects of applying traditional animation principles to 3D computer animation. User interface designers as well can draw upon the wealth of experience in the field of cartoon animation to achieve similar successes of communication, vibrancy, and illusion.

4 USER INTERFACES ARE NOT CARTOONS

Making user interface objects behave more like cartoon objects can be beneficial, but user interfaces are not cartoons. The most important difference between them is that the cartoon is a passive medium, while the user interface is an interactive one. Because the user is in control, the final product must be responsive to the user's desires. (For example, motion techniques like slow in and slow out, anticipation, and follow through in the Self interface are not imposed upon objects when they are being explicitly moved by the user.) While the animation should be designed so that the user will not feel as if he is waiting for it to play out (it should be both fast enough and engaging enough so that the user isn't consciously aware of it), the user may want to move to the next interaction before an animation finishes. In those cases, the user should be given immediate control, but the integrity of the interface should still be maintained by the principled use of animation techniques. For example, when user interaction requires a moving object to skip the rest of its planned movement and go immediately to its destination, motion blur can still be used to visually connect the large jump.

Another difference between cartoons and user interfaces are their purposes. Cartoons are purely for enjoyment and diversion; user interfaces are usually for getting work done. Of course, user interfaces should be enjoyable as well, but its more serious nature usually precludes the employment of some of the entertainment values of cartoons. Cartoons are often wacky and silly, and although the techniques that make the wackiness comprehensible to the audience are the same that are used to make actions in the user interface comprehensible to the user, the wackiness itself does not need to be transferred. For example, cartoons can use anticipation to set up "surprise gags" [24], by making the anticipation suggest to the audience one action, while having a different action actually happen. Such a surprise can be desirable in cartoons, but would be quite disturbing in the user interface. The user interface uses the effect of solidity and physicality from cartoons, but not the affect of silliness and manic action.

Because a user interface is a tool for doing some task, animation in it should be as fast as it can be while still preserving legibility. For example, in moving an object from one side of the screen to the other, an appreciation of motion blur makes it possible to accomplish this in only two frames. (Like the Road Runner exiting stage left.) When used in this fashion, animation may actually speed up an interface, by reducing the time it takes a user to perform *and comprehend* an action.

5 IMPLEMENTATION NOTES

One goal of the Self user interface effort is to explore use of these animation techniques in a practical setting—which means using stock graphics hardware while at the same time ensuring the high performance necessary for interactive use. To achieve the interactive speeds of 20 to 30 frames per second without special hardware support, we use colormap animation—colormap cycling for dissolves, colormap overlays for moving objects, and colormap double-buffering for moving arrows [4, 18]. We chose to accept the limited number of colors that results from the use of colormap animation techniques as a trade-off for high performance.

The 8-bit colormap is divided into four parts: a 3-bit "stationary" layer, a 3-bit "moving" layer, and two single-bit "arrow" layers. Moving animations are carried out by elevating the animated object to the moving layer. For example, when a user grabs an object and moves it around on the screen, the moving object is drawn only on the moving layer; as it is moved about, its vacated locations are filled in with a "transparent" color that allows the stationary layer to show through. Dissolves are handled by cycling through colormaps that change the transparency of the moving layer.

Arrows are a special case. Arrows connected to a moving object are drawn on their own layer, again allowing quick erasing with a transparent color. Since there may be many arrows attached to a moving object, it is important to be able to efficiently erase and redraw many arrows at once. To prevent flickering, the arrows are double-buffered via the colormap; thus the two arrow layers.

As mentioned above, a drawback to using colormap manipulations for animation is that it can severely limit the number of colors available. For example, although an 8-bit framebuffer supports 256 colors, the above architecture limits the Self user interface to seven colors for general use, one color for background, and one color for arrows. A separate limitation of this architecture is that, although more than one thing can be animated at once, spatially overlapping animations are not directly supported.

Some user testing of the animation in the Self user interface has been carried out. Most of the users fall into one of two categories: those who immediately notice the use of animation, and are generally delighted; and those who seem to take the animation for granted from the start. The latter tend to be people who are less involved in the computer field, which suggests that such animations conform reasonably well to natural experience and expectations. Observations of users also show that inconsistencies in the animation or performance glitches can suddenly wreck the illusion of the interface and draw attention to the animation itself. Perhaps because animated objects seem more real, defects in their behavior can be more disturbing and disruptive than a defect in an unanimated object. Further study is needed to more fully determine what are the useful boundaries of animation in the user interface, and the effect of animation as the user becomes more experienced with the system.

6 CONCLUSION

The benefits of applying techniques from cartoon animation to the user interface are both cognitive and affective. By making it easier for the user to track objects and understand what is changing on the screen, animation offloads some of the cognitive burden associated with deciphering what is going on in the interface from higher cognitive centers to the periphery of the nervous system. By eliminating flashes as the contents of the screen change suddenly, animation makes the interface less startling and thus makes the user's experience more pleasant. By presenting a physical world in which motion is realistic and convincing, animation engages the user, keeping her in the world, concerned with the task and not the mechanics of the interface.

We showed how the principles of animation could be applied to one user interface, interacting with objects in the Self world. These techniques are widely applicable, and may have additional benefits in certain kinds of interfaces. In particular, collaborative applications stand to benefit greatly from the use of this kind of animation: in addition to the user's own actions, remote users often will initiate actions, so the user may not even have the advantage of expecting something to happen. Anticipation, slow in and slow out, motion blur, and follow through can be used to more fully and more gently inform the user about remotely initiated actions.

Cartoon animation goes far beyond static presentations and even beyond straight animation in elucidating action in the interface. The animation doesn't have to be slow, or distracting, or silly; on the contrary, with careful tuning, cartoon animation can turn the user interface into an understandable, engaging, and pleasurable experience.

ACKNOWLEDGMENTS

REFERENCES

1. Baecker, R. and Small, I. Animation at the interface. In B. Laurel, *The Art of Human-Computer Interface Design*, Addison-Wesley, New York, 1990.

2. Baecker, R., Small, I., and Mander, R. Bringing icons to life. *CHI '91 Conference Proceedings*, 1991, 1-6.

3. Böcker, H.-D. and Herczeg, J. Track—A trace construction kit. *CHI '90 Conference Proceedings*, 1990, 415-422.

4. Booth, K.S. and MacKay, S.A. Techniques for frame buffer animation. *Graphics Interface '82 Conference Proceedings*, 1982, 213-223.

5. Brown, M.H. Perspectives on algorithm animation. *CHI '88 Conference Proceedings*, 1988, 33-38.

6. Card, S.K., Robertson, G.G. and Mackinlay, J.D. The Information Visualizer, an information workspace. *CHI '91 Conference Proceedings*, 1991, 181-188.

7. Donoho, A.W., Donoho, D.L. and Gasko, M. MacSpin: Dynamic graphics on a desktop computer. *IEEE Computer Graphics and Applications*, 8(4), 1988, 51-58.

8. Duisberg, R.A. Animation using temporal constraints: An overview of the Animus system. *Human-Computer Interaction*, 3(3), 1987-1988, 275-307.

9. Hutchins, E.L., Hollan, J.D. and Norman, D.A. Direct manipulation interfaces. In D. Norman & S. Draper, *User Centered System Design*. Lawrence Erlbaum, Hillsdale NJ, 1986, 87-124.

10. Jones, C. *Chuck Amuck: The Life and Times of an Animated Cartoonist*. Farrar Straus Giroux, New York, 1989.

11. Lasseter, J. Principles of traditional animation applied to 3D computer animation. *SIGGRAPH '87 Conference Proceedings*, 1987, 35-44.

12. Laurel, B. Interface as mimesis. In D. Norman & S. Draper, *User Centered System Design*. Lawrence Erlbaum, Hillsdale NJ, 1986, 67-85.

13. Laurel, B. *Computers as Theatre*. Addison-Wesley, New York, 1991.

14. Mackinlay, J.D., Robertson, G.G. and Card, S.K. The Perspective Wall: Detail and context smoothly integrated. *CHI '91 Conference Proceedings*, 1991, 173-179.

15. Microsoft Corporation. *Microsoft Windows for Workgroups Version 3.1 User's Guide*. Microsoft Corporation, 1992.

16. Palmiter, S. and Elkerton, J. An evaluation of animated demonstrations for learning computer-based tasks. *CHI '91 Conference Proceedings*, 1991, 257-263.

17. Robertson, G.G., Mackinlay, J.D. and Card, S.K. Cone Trees: Animated 3D visualizations of hierarchical information. *CHI '91 Conference Proceedings*, 1991, 189-194.

18. Shoup, R. Color table animation. *SIGGRAPH '79 Conference Proceedings*, 1979, 8-13.

19. Smith, R.B. The Alternate Reality Kit: An animated environment for creating interactive simulations. *Proceedings of the 1986 IEEE Computer Society Workshop on Visual Languages*, 1986, 99-106.

20. Smith, R.B. Experiences with the Alternate Reality Kit: An example of the tension between literalism and magic. *CHI+GI '87 Conference Proceedings*, 1987, 61-67.

21. Stasko, J.T. Using direct manipulation to build algorithm animations by demonstration. *CHI '91 Conference Proceedings*, 1991, 307-314.

22. Sukaviriya, P. Dynamic construction of animated help from application context. *UIST '88 Conference Proceedings*, 1988, 190-202.

23. Sukaviriya, P. and Foley, J.D. Coupling a UI framework with automatic generation of context-sensitive animated help. *UIST '90 Conference Proceedings*, 1990, 152-166.

24. Thomas, F. and Johnston, O. *Disney Animation: The Illusion of Life*. Abbeville Press, New York, 1981.

25. Tognazzini, B. Principles, techniques, and ethics of stage magic and their application to human interface design. *INTERCHI '93 Conference Proceedings*, 1993, 355-362.

26. Ungar, D. and Smith, R. Self: The power of simplicity. *OOPSLA '87 Conference Proceedings*, 1987, 227-241.

27. Williams, G. The Apple Macintosh Computer. *Byte*, 9(2), 1984, 30-54.