View-Based Animation

Susanne Jul
Electrical Engineering and Computer Science
University of Michigan
1101 Beal Av.
Ann Arbor MI 48109-2110 USA
sjul@umich.edu

ABSTRACT
We present a technique for animating objects that is based on the concept of viewpoint rather than on time. Object properties are computed as a function of the viewpoint. The viewpoint, in turn, is controlled manually or is programmed to change as a function of time. This allows objects in a fly-through or zooming interface to be animated selectively. The speed of the animation is controlled by the viewer, and the animation can be run forwards or backwards and can be stopped, started and stepped by manipulating the view controls. This technique is not intended to replace time-based animation, but is rather to be used in special cases or in conjunction with time-based animation.

KEYWORDS
Computer Animation, View-Based Interaction, Pad++, Fly-Through Interfaces, Zooming Interfaces.

INTRODUCTION
Fly-through and zooming interfaces bring a cinematographic concept of “viewpoint” to user interfaces. Pad++ [1], a zooming interface, extends the utility of this concept by enabling objects to render differently depending on the scale (or magnification) of the current viewpoint. In the present work, we expand this capability by enabling properties of objects to be computed as functions of the viewpoint. For example, a text object might rotate about its center as a function of its distance from the current viewpoint. This allows us to create *view-based animations* wherein moving the viewpoint causes objects to be animated: The text object spins as the view moves closer. This technique is not a replacement for time-based animation, but is useful when only a small number of objects need special behaviors, and in situations where the timing of the animation needs to be flexible, such as in live presentations.

ANIMATION BEHAVIORS
Computer animations are generally achieved through changing the view onto a scene of objects (moving the camera) or through changing visual properties of individual objects (moving or changing the objects). Traditionally, both types of changes progress with time. In view-based animation, we link changes in object properties to changes in the view. Changes to the view, in turn, are controlled by the user directly or are controlled programmatically as functions of time. View-based animations, like time-based animations, “[create] the illusion of motion,”[1] but differ in that the viewer must move for the illusion to become apparent.

Zooming provides a natural animation behavior: As the scale of the view is increased or decreased, objects grow larger or smaller, respectively, in a linear relationship. We achieve more complex animation effects by providing behaviors that can be attached to objects. These behaviors alter the properties of an object—e.g., size, position, or color—as a function of the viewpoint. The rendering routines for objects are unaffected, so the behaviors can be attached to arbitrary objects. Behaviors are modular, so they can be combined or reused easily.

In authoring, view-based animation is similar to key-frame animation. However, the intervening “frames” are determined at run-time, and depend on the path and speed the viewer takes through the system. Two viewers may take different paths, possibly simultaneously, and, so, be exposed to different animations.

EXAMPLE 1: INFORMATION PRESENTATION
We have used view-based animations extensively in presentations developed in Pad++ for use as conference talks and/or interactive demonstrations. In either case, the animations help the viewer find and follow transitions between different parts of the content.

Figure 1 shows a presentation that employs view-based animation. The first (left-most) frame shows two topic areas that may be explored. In frame 2, the viewer has zoomed in on the word “Photographs.” This text object has an inverse-zoom behavior attached, so grows *smaller* as the scale of the view increases. The actual photographs magnify normally, creating a crossover effect as the images grow and the label shrinks. In frame 3, both photographs and the names of the photographers, magnifying normally, have become visible. The “Photographs” label has disappeared. In frame 4, an exaggerated-zoom behavior attached to photographs and names has been activated. Any name that is in the *center* of the view and its associated photographs magnify at a *greater* than normal rate. The name has also been assigned a color-change behavior. The resulting focus + context [3] interaction resembles a magic lens [2]

[1] From the American Heritage Dictionary definition of “animate.”

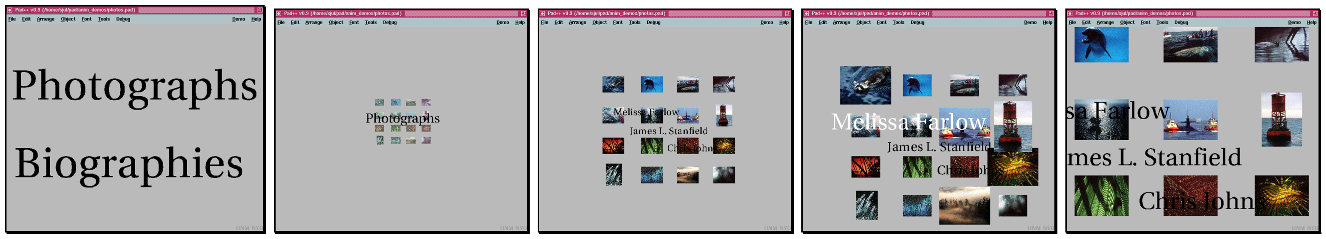

Figure 1 Screenshots of a constant zoom-in sequence. Read left to right.

approach, but allows different objects to exhibit different behaviors in the same view. It also allows one object to exhibit different behaviors in only slightly differing views. In frame 5, no names are centered, so all objects again magnify normally.

In authoring presentations, we find that considering how individual elements can be animated to create supportive visual transitions helps us pay greater attention to how the different parts of the presentation are related to each other logically. In giving or exploring presentations, the viewer controls the view-changes, so the animations automatically follow the pace and direction of the viewer.

EXAMPLE 2: INFORMATION VISUALIZATION

Our first example used geometric view coordinates—x, y position or magnification—as the animation parameter. Our second example uses an abstract view coordinate to control the animation. This coordinate is maintained by the display system, but, unlike geometric view coordinates, is not used directly in rendering.

Figures 2 and 3 show two interactive visualizations of a data set with three dimensions, such as might be plotted on a line graph. Two of the dimensions are linked to screen-position (Figure 2) or size/color (Figure 3) of object. The third dimension is linked to the abstract coordinate, which is controlled manually by the user. In this case, it is controlled with mouse buttons, but it could easily be linked to a slider or other widget. These examples could readily be created by direct encoding. However, by using view-based animation and an abstract view coordinate, the implementation for the two examples differ only by the line of code that specifies the behaviors attached to the objects.

The two examples show the same data values. There are two data points for the x coordinate. Values for the y coordinate are computed as a function of the z coordinate (mapped onto the abstract view coordinate), interpolating between given data points. The three frames show the resulting plots for increasing values of the z coordinate. Manipulating the abstract view coordinate (the only part of the viewpoint under user control) animates the graphs.

SUMMARY

We have introduced the concept of view-based animation wherein computable object properties are mapped onto one or more continuous view dimensions. This allows select objects to be animated as the viewer moves around them. We achieve these animation effects through modular behaviors that are attached to objects. The animation is under full control of the viewer, so can be paced and directed interactively. This technique is intended to be used in special situations, such as zooming interfaces, or in conjunction with time-based animation.

ACKNOWLEDGEMENTS

My deepest thanks to Professor George W. Furnas for unflagging intellectual and emotional support as well as a good deal of plain good advice. This work was supported, in part, by Irma M. Wyman through the Center for the Education of Women at the University of Michigan and by ARPA Grant N66001-94-6039.

REFERENCES

1. Bedersen, B. B., Hollan, J. D. (1994). Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of ACM UIST'94*, New York, NY: ACM Press, 17-26.
2. Bier, E. A., Stone, M. C., Pier, K., Buxton, W., DeRose, T. D. (1993). Toolglass and Magic Lenses: The See-Through Interface. *Proceedings of the 20th Annual Conference on Computer Graphics*, 73-80
3. Furnas, G. W. (1986). Generalized Fisheye Views. Human Factors in Computing Systems CHI '86 *Conference Proceedings*, 16-23.

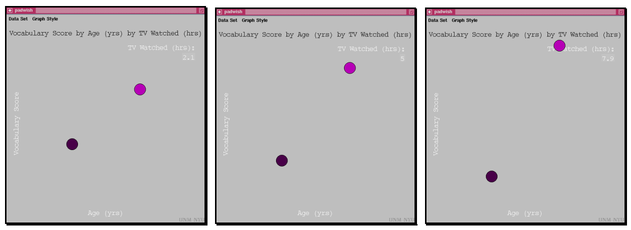

Figure 2 Mapping x and y to screen-position.

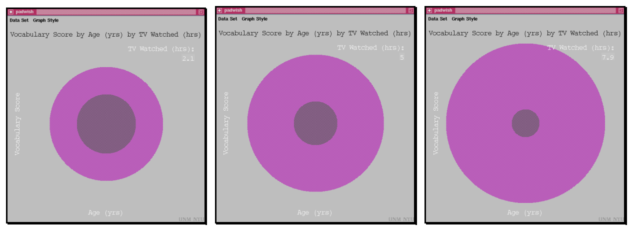

Figure 3 Mapping x to color and y to size.