The Kinetic Typography Engine: An Extensible System for Animating Expressive Text

Johnny C. Lee*, **Jodi Forlizzi***†, **Scott E. Hudson***
*Human Computer Interaction Institute and †School of Design
Carnegie Mellon University,
Pittsburgh, PA 15213 USA
{ johnny, forlizzi, scott.hudson }@cs.cmu.edu

ABSTRACT
Kinetic typography – text that uses movement or other temporal change – has recently emerged as a new form of communication. As we hope to illustrate in this paper, kinetic typography can be seen as bringing some of the expressive power of film – such as its ability to convey emotion, portray compelling characters, and visually direct attention – to the strong communicative properties of text. Although kinetic typography offers substantial promise for expressive communications, it has not been widely exploited outside a few limited application areas (most notably in TV advertising). One of the reasons for this has been the lack of tools directly supporting it, and the accompanying difficulty in creating dynamic text. This paper presents a first step in remedying this situation – an extensible and robust system for animating text in a wide variety of forms. By supporting an appropriate set of carefully factored abstractions, this *engine* provides a relatively small set of components that can be plugged together to create a wide range of different expressions. It provides new techniques for automating effects used in traditional cartoon animation, and provides specific support for typographic manipulations.

KEYWORDS: kinetic typography, dynamic text, time-based presentation, automating animation effects.

INTRODUCTION
The written word is one of humanity's most powerful and significant inventions. For over 4000 years, its basic communicative purpose has not changed. However, the method in which written communication is authored and presented has never stopped evolving. From cuneiform markings on clay tablets, to pen and parchment, to the Gutenberg press, to computers and the internet, technology has always provided text with new mediums to express itself. The explosion of available computing power has added a new possibility: *kinetic typography* – text that moves or otherwise changes over time.

UIST'02, October 27-30, 2002, Paris, FRANCE.

Kinetic typography can be seen as a vehicle for adding some of the properties of film to that of text. For example, kinetic typography can be effective in conveying a speaker's tone of voice, qualities of character, and affective (emotional) qualities of text [Ford97]. It may also allow for a different kind of engagement with the viewer than static text, and in some cases, may explicitly direct or manipulate the attention of the viewer.

In fact, the first known use of kinetic typography appeared in film – specifically, Saul Bass' opening credit sequence for Hitchcock's *North by Northwest* [Bass59] and later *Psycho* [Bass60]. This work stemmed in part from a desire to have the opening credits set the stage for the film by establishing a mood, rather than simply conveying the information of the credits. Use of kinetic typography is now commonplace for this purpose, and is also very heavily used in TV advertising where its ability to convey emotive content and direct the user's attention is generally a good match to the goals of advertising. We believe that if it can be made accessible via good tools, the power of kinetic typography can also be applied to benefit other areas of digital communications.

A second origin for time-based presentation of text comes independently from psychological studies of perception and reading. For example, [Mill87] studies perceptual effects of a number of text presentations, such as scrolling text. One of the most fruitful of these is a method known as *Rapid Serial Visual Presentation* (RSVP), where text is displayed one word at a time in a fixed position [Pott84]. Studies have shown that, because scanning eye movements are unnecessary when using RSVP, it can result in rapid reading without a need for special training. In addition, RSVP techniques provide advantages for designers because they allow words to be treated independently without regard to effects on adjacent text elements. Finally, RSVP can be seen as a means for trading time for space, potentially allowing large bodies of text to be shown at readable sizes on small displays.

Figures 1-3 illustrate some of the things that kinetic typography can do. (Please refer to the video proceedings for dynamic renditions of these figures.) Figure 1 shows two different renditions of the same words expressing a different emotional tone. As described by Ishizaki [Ishi97]:

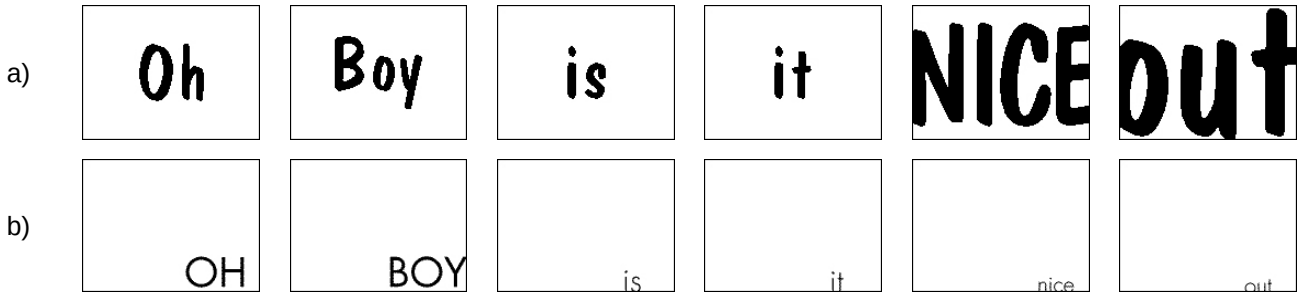

Figure 1. Expressing different emotional tones for the same text

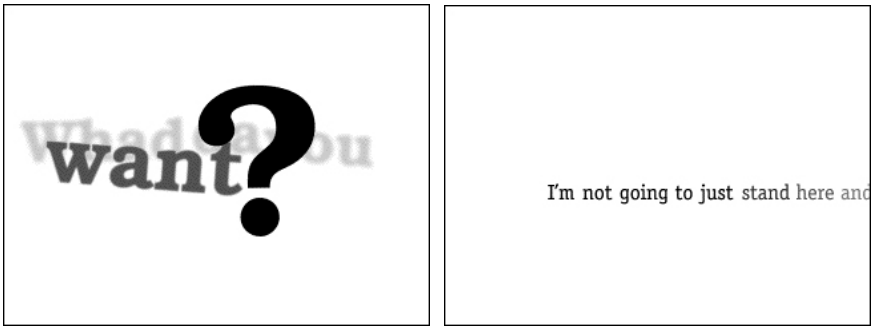

Figure 2. Two characters from the Monty Python "argument clinic" sketch

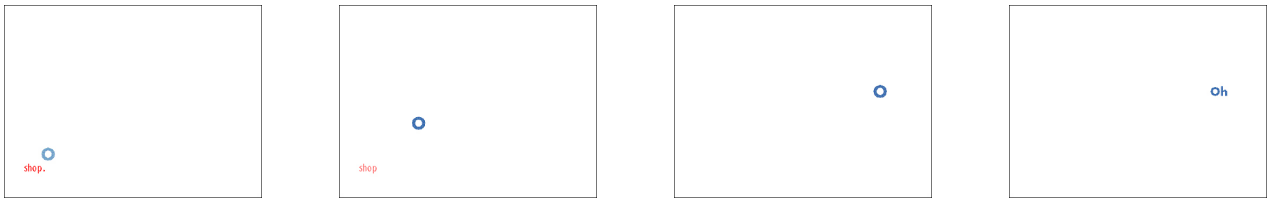

Figure 3. Direction of attention using movement

> *"Suppose it is early Saturday morning. The actor - perhaps a child or puppy - hurries to the door, eager to play outside. One speaker [top] finds sunshine; the other, rain."*

In Figure 1a, the choice of typeface, rapid rhythmic motions, changes of scale, and rotation, all combine to convey a sense of exuberance. Whereas, in Figure 1b, a very plain typeface has been chosen, and a combination of slow and decelerating pace, reduction of typeface weight, and a shrinking motion analogous to slumping of the shoulders, combines to convey a sense of disappointment.

Figure 2 illustrates the creation of clearly distinct characters in a dialog. This is done by having text originate from two distinct spatial regions and flow in two different directions, as well as giving each character a particular tone. On the left, shouting is expressed by large expanding, shaking, and vibrating text, which lingers and appears to reverberate. On the right a more diminutive tone is expressed by small size, slower pace, and clear adherence to the text baseline. We also see a clear change in tone of voice at a particular point in the piece, again accomplished by manipulating aspects such as size, pacing, maintenance of baseline, and fading.

Finally, Figure 3 also illustrates the creation and interplay of distinct characters, this time created using identifiably different aspects of position, typeface, size, and color. In addition, the segment highlighted by the static shots shown here is used to very clearly and inexorably transition the user's attention from the ongoing dialog of the character at the left, to the entry of the character at the right.

While the utility of kinetic typography has been recognized (and seen some practical use) for several decades, it still lacks the rich history, and accompanying academic study, of either static typography or film. There have been efforts to provide a taxonomy of typographic forms related to interface design [Bork83], as well as examinations of dynamic layouts of static typographic compositions [Hieb92, Chan98]. This work builds on a history of work that computationally explores graphical techniques, scaling and shading, as a means of focusing attention on a particular area of a display and increasing the amount of information the screen can display [Gren96, Kand98, Masu95, Neuw92]. However, the earliest systematic investigations of time-based presentation of text in its own right began in the late 1980s, when designers and technologists manipulated typographic characters by using simulated physical movements, such as springs [Smal89]. More recent research has attempted to provide a descriptive language of time-based forms [Wong95, Ishi96,

Ford97, Ishi98]. Finally, [Lewi99] discuss a very interesting interface for editing "live" kinetic typography.

ANIMATION TECHNIQUES FOR KINETIC TEXT

The craft of traditional (cartoon) animation was tremendously advanced by the Disney animators of the 20s and 30s, and from this work emerged a set of design principles and specific techniques which could be employed to turn drawing into compelling, engaging, and lifelike (*animate*) characters [Thom81]. Adaptation of these techniques to 3D computer graphics was described by Lassiter in [Lass87] (these techniques were also related to interactive systems in [Chan93] and toolkit level abstractions were described in [Huds93] and [Myer96]).

These techniques can easily be adapted to use in kinetic typography. Useful techniques from traditional animation include, for example, slow-in slow-out movement, squash and stretch, movement in arcs, anticipation, follow-through and secondary action. Later we will introduce new techniques for supporting several of these within the kinetic typography engine.

In addition to those from traditional animation, a number of techniques specific to kinetic typography have also been identified.

Previous work such as [Ford97, Ishi98] has identified several key areas in which kinetic typography has been particularly successful. These include:

- Expression of affective (emotional) content,
- Creation of characters, and
- Capture or direction of attention.

Working towards the goal of conveying emotive content, a set of techniques has been developed for expressing the equivalent of *tone of voice*. Tone of voice can be understood as variations in pronunciation when segments of speech including syllables, words and phrases are articulated. Tone of voice can be roughly divided into two sets of features: paralinguistic features, such as the husky quality of a voice, and prosodic or linguistic features, such as pitch, loudness, tempo or speed of delivery [Crys75, Ford97].

It has been difficult to find effective ways to portray paralinguistic features in recognizable form. However, aspects of pitch, loudness, and tempo can be effectively conveyed. For example, large upward or downward motions can convey rising or falling pitch. Loudness is used prosodically for a number of purposes. It can affect an entire utterance (establishing its volume, and producing significant effects such as whispers and shouting), a single word or phrase (to supply accent), or a specific syllable (for stress). Loudness can be mimicked by changing the size of text, as well as its weight, and occasionally contrast or color. For high volumes, motions mimicking vibration can be used. When combined with vibration, persistence of words beyond the point of their utterance can be used to mimic reverberation (see Figure 2). Since speed of delivery and tempo (as well as its regularization as rhythm) are temporal effects, these can generally be directly applied by manipulating timing, duration, and pacing. Finally, speed of delivery effects applied to individual words or phrases can be mimicked by modifications to text width (i.e., spatial stretching to indicate a temporally stretched word) as well as scaling effects.

The other major class of techniques for conveying emotive content is *analogous motion*. Analogous motion uses movements reminiscent of human actions that convey emotional content. For example, small vibrations that are analogous to trembling can be used to convey affective content with high levels of arousal, such as anticipation, excitement, or anger. Figure 1b has shown a "shrinking" motion analogous to slumping the shoulders that can convey affect with lower arousal such as disappointment. When combined with appropriate content, slow rhythmic motions reminiscent of calm breathing appear to induce feelings of empathy.

While expression of emotion has been a central use of kinetic typography, it is also important to recognize its limitations. In particular, experience has shown that while kinetic typography can set the emotional tone of ambiguous text, it cannot normally replace or override strong emotive content intrinsic to the meaning of the text. For example, it is not normally possible to use kinetic typography to make a sad story into a happy one. Instead, kinetic typography can reinforce or temper emotive content already present.

In addition to emotional content, kinetic typography has also been successful in portraying *characters* and *dialog*. Principles for portraying characters have been adapted from film [Smit95]. These principles include the need to establish identification and re-identification of a character across appearances (and conversely separation of distinct characters). We establish this *recognition* of a character by use of distinct, persistent, and identifiable visual, spatial, and other properties. Recognition techniques can be seen in Figures 2 and 3 where consistent use of spatial location, movement patterns, typeface, type weight, and tone of voice techniques are all used to distinguish characters.

A second important technique related to character creation is *attachment*. In film we typically follow (are *attached* to) one or two characters through time, with other characters appearing only as they encounter the main character(s). Techniques for attachment in kinetic typography are still being investigated. However, use of spatial locality is one way to establish attachment.

A final communicative goal found in many examples of kinetic typography is the capture and direction of attention. Here we can draw principles from the scientific knowledge of perceptual and cognitive psychology. For example, perceptual phenomena that have sudden onset tend to induce attentional capture effects [Pash98]. This fact can be used to guide the manipulation of timing and pacing to produce different demands for attention. Similarly, moving objects intrinsically draw ones attention, and

Figure 4. Geometric scaling (top) vs. typographic scaling (bottom).

movement along a path to an end point will tend to strongly draw the eye to that end point. Hence, large sweeping movements are a useful vehicle for explicitly transitioning the user's attention from one location to another (again, see Figure 3).

Note that an important potential pitfall of kinetic typography techniques can be that they demand too much attention from the user. As a result, it is important to consider techniques for manipulation of attention in the negative as well as positive, for example, avoiding sudden onsets when attention capture is not desirable.

KINETIC ENGINE ARCHITECTURE

To implement the breadth of effects needed to support kinetic typography, we have created a general purpose kinetic typography engine based on an extensible animation architecture. (The engine and its behavior library are implemented in just over 2700 lines of Java code under J2SE 1.3.1)

Under this architecture, individual textual elements (representing phrases, words, or even individual letters as needed for a presentation) are organized into hierarchical structures. In particular, objects representing *strings* (which include code for rendering text) are placed in *sequence* objects, which may in turn be contained in other sequences, etc. It is also possible to place non-textual display objects in this hierarchy. These sequence trees are reminiscent of the scene graphs typically used in 3D computer graphics as well the interactor trees employed in modern UI toolkits, and they serve many of the same purposes (such hierarchies were also used in the kinetic typography system described in [Lewi99]). For example, sequence trees provide for a hierarchical coordinate system whereby the position of a child object or subsequence is expressed relative to the position of its parent. However, for our architecture, this hierarchy is taken further by applying similar concepts to time. As a result, the expression of time within a subsequence is relative to the designated start time of its parent sequence. This temporal hierarchy offers the same kind of benefits as the typical spatial hierarchy – for example, allowing whole subsequences to be moved together in time.

Each object and sequence in the tree has a set of *properties* that affect the details of its behavior. For example every object has properties controlling its *x* and *y* position and string objects have properties controlling their typeface, type size, type style, kerning, color, transparency, rotation, shear, etc.

Note that while it may be tempting to consider the facilities of a typical general-purpose animation system sufficient, for kinetic typography, we have found it important to pay particular attention to typographic properties in ways that most general animation systems do not.

For example, Figure 4 presents two versions of the same text object, one that has been scaled geometrically and one scaled typographically using font size. Notice the poor letter spacing generated from the geometric transformation in comparison to the typographic transformation.

In general, there are many detailed typographic properties that may need to be manipulated [Brin01]. For example, both [Rose98] and [Cho99] refer to the concept of continuously parameterized fonts, or malleable fonts that may be manipulated so dramatically they appear to be completely different type faces. The architecture of the kinetic typography engine was designed to accommodate these kinds of manipulations (in addition to geometric manipulations) and provides the infrastructure necessary to animate any component of them (although our current text rendering objects do not currently make use these types of advanced fonts).

The kinetic typography engine works by varying the properties of objects within the system over time. This in turn determines how visible objects are drawn over time. Changes are accomplished by attaching *behaviors* to objects (in a manner similar to the attachment of animation constraints to slots in [Myer96]). A behavior is simply an object which computes a value based on another set of values (one of which is an indication of the current time). For example, to move a word across the screen at a uniform pace horizontally, a *UniformChange* behavior object could be attached to the *x* property of the corresponding string object as illustrated in Figure 5.

The input values used by a behavior are expressed as properties. Like all other properties, they can either simply be assigned static values, or they can have time varying behaviors attached to them. The connection of behaviors to behaviors forms a data-flow style computation that is conceptually related to the constraint-based animation control system described in [Duis86]. As we will see later, this ability to modify one behavior with another provides a powerful behavior composition capability that allows sophisticated and varied effects to be created from a very

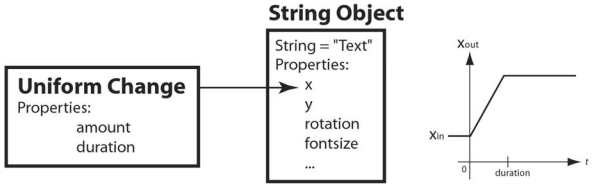

Figure 5. Behavior to move a word horizontally

simple base library of behaviors.

Because of the extensive and central use of time in an animation system, we have found it useful to uniformly "factor out" time from the other values involved in behavior computations, and provide special mechanisms for manipulating it. In particular, a hierarchical system of times is provided which operates much like a hierarchical coordinate system in the spatial domain. Each sequence can provide a time transformation function (a special form of behavior object called a *time filter*) that indicates how the local system of time seen by its children is related to the system of time it is embedded within. For example, local times can be offset from a more global time system using a *Delay* time filter, or local time can be sped up or slowed down using a *Speed* filter.

In addition, each behavior may optionally have a time filter associated with it that affects the apparent progression of time for that behavior. Finally, time filters can be composed by chaining, and most time filters are controlled by a set of parameter values. These parameter values are of course represented as properties to which other behaviors can be attached.

While not providing any fundamentally new capabilities, "factoring out" time has provided convenient mechanisms for implementing a number of common effects. For example, simple time filters are provided for repetition (*Loop*), reversal of action (*Reverse*), general pacing control (*PaceInOut* which provides capabilities similar to the general pacing function described in [Huds93]), and even programmatic or interactive stopping of time (*Pause*).

EXECUTION

Once a set of text objects with attached behaviors and time filters has been established, rendering of kinetic typography with the engine is very simple. We make repeated drawing traversals to render each frame. Each traversal step is passed the current time from its parent (expressed in its parent's time system) as a part of its drawing request. This is transformed by executing any local time filters to obtain a time expressed in *local time*. This local time is then passed on to each component being drawn (which may in turn transform it, etc.). Prior to drawing each object, we request the value of each of its properties. To obtain these values it may be necessary to execute a behavior attached to it. The behavior similarly transforms its notion of local time via any attached time filters, and then requests each of its property values (possibly causing additional time filters and behaviors to be invoked, etc.). This computation is equivalent to that of a typical one-way constraint system (see for example: [Huds91]).

To date we have concentrated only on real-time rendering. In this setting, we simply repeatedly make draw requests based on the current time – enforcing delays so as not to render faster than actual refresh rates, but otherwise letting the system run free. The same code could, however, also be used in an off-line fashion by rendering each frame separately at its intended time in the final product.

BEHAVIOR LIBRARY

The architecture and runtime system described above provide a robust and flexible base for implementing kinetic typography. To make a complete tool, however, this base needs a library of actual behaviors and time filters that provide the right actions, give very good coverage of the effects needed, and supports easy extensibility in the rare cases where new behaviors might be required.

In designing a set of behaviors for such a library, we have taken what can be seen as a *signal processing* approach where nearly every behavior is seen as either generating or transforming a *waveform* – a single dimensional quantity that varies over time. This approach has lead to a remarkably small set of rather simple (and easy to implement) behaviors that nonetheless have very wide coverage of effects. Further, by considering waveforms as functions over time we have uncovered simple analytical approaches to providing automated support for traditional animation techniques such as secondary action.

In order to increase uniformity, all current behaviors are treated as waveform transformations (in signal processing terms: *transfer functions*). For behaviors that are naturally waveform generators, this is achieved by simply adding the generated waveform to the incoming waveform (which can be set to a constant zero if desired).

One of the simplest but most useful behaviors in our library is *UniformChange*. In addition to time and an incoming waveform value, this behavior is parameterized by the properties of amount, and duration and it produces a simple uniformly increasing value. Specifically:

$$V_{out} = V_{in} \quad \text{when} \quad t \le 0$$

$$V_{out} = V_{in} + amount \times \left(\frac{t}{duration} \right) \quad \text{when} \quad 0 < t < duration$$

$$V_{out} = V_{in} + amount \quad \text{when} \quad t \ge duration$$

Two other basic behaviors include *Oscillate*, which produces an oscillating sine wave, and *Curve*, which produces a curved waveform controlled by a simple single segment Bézier curve. Like *UniformChange*, these behaviors are parameterized by amount and duration properties. *Oscillate* is also parameterized by frequency and phase properties. Finally, the *Jitter* behavior provides a waveform containing random values with certain spacing in time. It can be computed as:

$$V_{out} = V_{in} + amount \times 2 \times \left(Rand\left(\left\lceil \frac{t}{rate} \right\rceil + seed \right) - 0.5 \right) \quad \text{when} \quad 0 < t < duration$$

$$V_{out} = V_{in} \quad \text{otherwise}$$

where *Rand(x)* is a pseudo-random number generator restarted with seed value *x* and *seed* is an unique static integer for each instance of *Jitter*.

The library also includes a set of basic time filters. For example, the *Delay* filter simply computes:

$$T_{out} = T_{in} - amount \quad \text{when} \quad 0 < t < duration$$
$$T_{out} = T_{in} \quad \text{otherwise}$$

Since *Delay* is structured as a time filter it can be used to shift any waveform without regard to how it is constructed.

The *Speed* time filter computes:

$$T_{out} = T_{in} \times amount \quad \text{when} \quad 0 < t < duration$$
$$T_{out} = T_{in} \quad \text{otherwise}$$

and can be used to speed up or slow down an effect or sequence. Another standard time filter is *Loop*, which causes an interval of time to be delivered repeatedly. It is parameterized by period and duration, and computes:

$$T_{out} = T_{in} \bmod period \quad \text{when} \quad 0 < t < duration.$$
$$T_{out} = T_{in} \quad \text{otherwise}$$

Additional time filters include *Reverse*, which causes time to proceed backwards within a fixed interval, and *PaceInOut*, which provides non-uniform pacing during an interval based on a simple single segment Bézier pacing curve. This can be used, for example to implement slow-in/slow-out motions.

In order to provide live programmatic control over animations, the *Pause* time filter is provided. This time filter has pausedState and tPaused properties. Whenever the pausedState property is changed from a zero to a non-zero value, the current time (Tin) is automatically copied to tPaused. The time filter then computes the following value:

$$T_{out} = T_{paused} \quad \text{when} \quad pauseState \neq 0$$
$$T_{out} = T_{in} - T_{paused} \quad \text{when} \quad pauseState = 0$$

In addition to live programmatic control, several simple behaviors are also provided for live interactive control. These include the *EventTrigger* time filter. *EventTrigger* is implemented much like *Pause* except that it releases or pauses action when a user input event matching a specification occurs. This allows a range of simple interactive behaviors to be created, including, most notably, animations that are initiated by mouse clicks on objects. In addition, behaviors are provided which return one of the coordinates of the current mouse position as their value, and a behavior that returns the distance from a point to the current mouse position as its value.

BEHAVIOR COMPOSITION

We do not claim that the current behavior library is complete. In fact, the ability to very easily extend it is an important benefit of the system. However, even though the current library is quite small, we can demonstrate that it is already capable of covering a wide range of effects occurring in practice. Such a small library can provide very rich behavior because of the composition capabilities afforded by the underlying architecture and the specific organization we have chosen for behaviors.

All behaviors in the library support at least two different mechanisms for composition: *additive* composition, and at least one opportunity for *functional* composition.

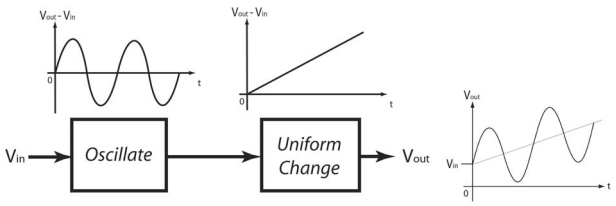

Figure 6. Waveform addition by chaining"

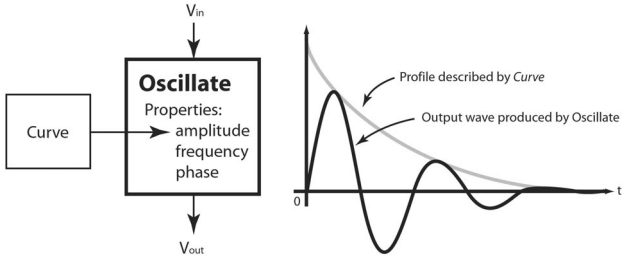

Figure 7. Waveform scaling by functional composition with amplitude

Since each behavior is structured as a waveform transformation that adds to another waveform, additive composition can be achieved by chaining behaviors as illustrated in Figure 6.

General functional composition can be achieved by attaching a behavior to a property of another behavior, or by attaching a time filter to a behavior. Figure 7 illustrates one use of this capability. It shows a multiplicative composition where one behavior imposes an envelope on the magnitude of another.

AUTOMATING EFFECTS

Based on principles of traditional (cartoon) animation developed and codified by Disney animators [Thom81], Lassiter enumerated 11 principles for applying traditional techniques to 3D computer animation [Lass87]. Several of these principles (notably, the need for "appeal" in characters, as well as most aspects of "staging" and "exaggeration") are too abstract for direct tool support. However, several of these principles lead directly to specific animation techniques that are extremely useful in creating appealing kinetic typography. These include:

Slow-in / slow-out movements: a pacing of action in which movements start slowly, move rapidly in the middle, then end slowly.

Movement in arcs: movement of objects along curved paths rather than straight lines.

Secondary action: the motion of an object resulting indirectly from another action, such as hair or clothing being blown back during a rapid movement, or arms trailing behind a torso at the beginning of a movement.

Squash and stretch: a volume conserving compression or extension of an object suggestive of the acceleration (stretch) or deceleration (squash) of a non-rigid body.

Anticipation: a motion before the action proper, intended

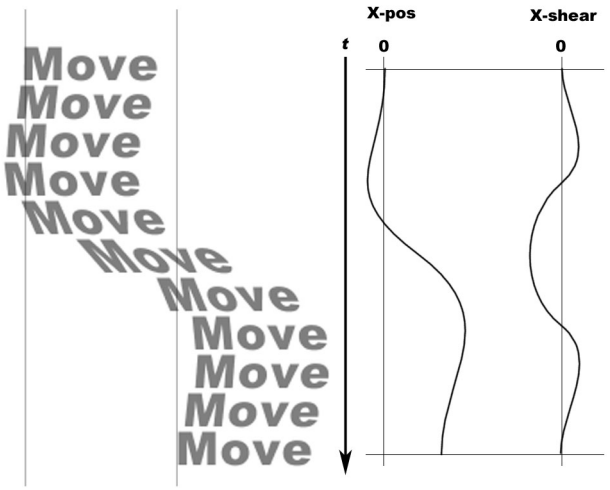

Figure 8. In many cases, taking the derivative of primary motion can yield the secondary motion curve. Here, the x-shear wave is simply the derivative of the x-position wave.

to set the stage and prepare the viewer for the action that is to about to take place – for example, an exaggerated leaning back before moving quickly forward.

Follow through: a motion indicating the termination of an action, typically carrying parts of an object somewhat beyond the termination point of the action such as arms swinging past a suddenly stopped torso.

Several of these effects are explicitly supported by our system. For example, slow-in / slow-out pacing of movements is directly supported by the PaceInOut time filter, and movement in arcs is directly supported by the Curve behavior for our library.

We also supply simple and direct support for secondary action and stretch. Initially these concepts seem quite dissimilar. However, they share a common fundamental: in typical usage for cartoons, these behaviors are correlated with the velocity of objects†. Therefore, taking the first derivative of motion can often yield the basic waveform needed to control secondary action. This derivation can be done explicitly by the programmer for simple behaviors or can be obtained computationally for arbitrary systems (via finite difference approximations). Note that we get this automatic support for secondary motion and stretch in a simple robust fashion precisely because we treat behaviors as waveforms and can apply simple analysis to them.

In Figure 8 we can see the waveforms for both the primary action (applied to *x-position* of the word) and its derivative

†Simple physics indicates that these effects should be correlated with acceleration rather than velocity. However, in a friction full world, many significant forces are in fact correlated with velocity and naïve observers may often confuse acceleration and velocity. It appears that in the caricatured world of "cartoon physics", effects such as wind resistance and other friction may have conceptual precedence over classical Newtonian inertia.

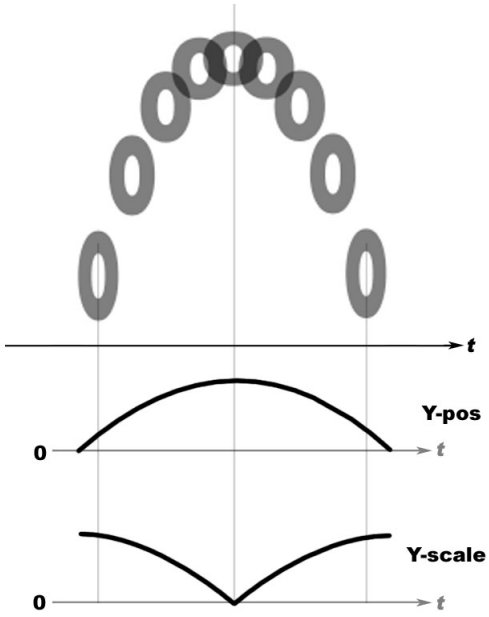

Figure 9. Stretching controlled by a function of velocity.

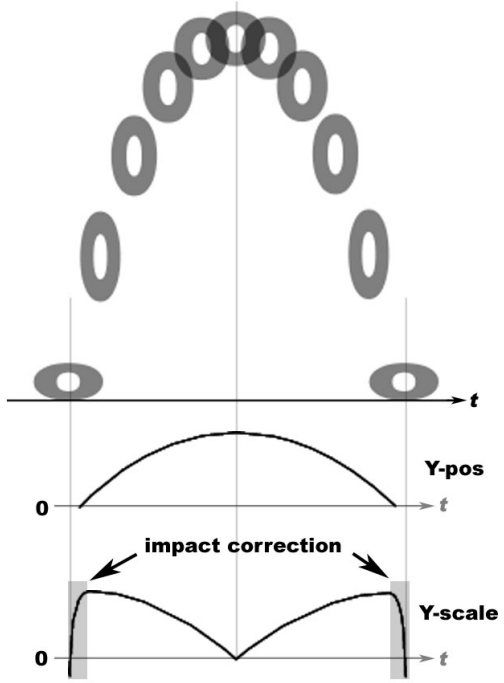

Figure 10. A combined squash and stretch waveform created from a stretch waveform with squash curves blended at the impact points.

(applied to the *x-shear* of the word). By applying these behaviors together we get a compelling overall effect which is much more appealing that a simple change in position.

In Figure 9, we see the same facility applied to automatically produce the control waveform for stretch. For stretch, we normally use the absolute value of the derivative (in this case of the primary motion in y position)

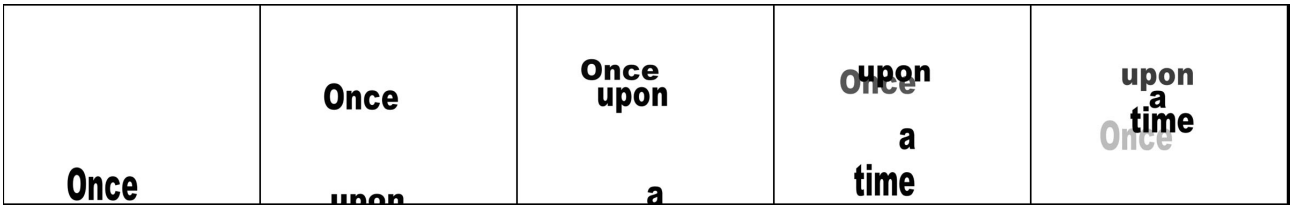

Figure 11. The HopIn composite effect.

since stretch depends on the speed, but not direction, of movement.

While we can analytically, hence automatically, derive the control waveform needed for stretch, slightly more work is needed to implement a combined squash and stretch. Squash applies when entering and leaving a point of impact. The exact details of the squash portion of the control waveform depend on the type of materials being simulated ("firm" vs. "soft") and/or the "liveliness" that is desired of the bouncing object. However, as illustrated in Figure 10, the overall form of a squash waveform is clear (and can be programmatically applied based on a few parameters). Prior to leaving an impact point, the waveform is negative, reflecting the negative geometric scaling of the squash. After departure, it fairly rapidly climbs to meet a point on the original waveform for the stretch. How negative the squash scaling is and how rapidly it fades into stretch can be controlled by simple parameters.

A number of forms of anticipation and follow through can also be supported as waveform manipulations. Counter movement for anticipation can be done by adding an initial waveform segment that slowly goes negative, holds there for a short time, then rapidly joins the original forward (positive) motion. Again, the magnitude, holdTime, and duration of this added segment become properties that the animator can set for detailed control over the result.

Finally, overshoot movements useful for follow through and secondary action can similarly be performed by waveform manipulation at the trailing end of a typical secondary action curve. Here the secondary action waveform should extend (stay positive) past the duration of the original primary motion. A more exaggerated version of this motion might also end with a damped vibration (similar to Figure 7).

COVERAGE OF THE DOMAIN

Over the last six years, several of our colleagues in the CMU School of Design have taught a class in kinetic typography techniques that considers many examples from commercial use. The typically very talented students in the class are also asked to create new expressions. From this work, a corpus of about 30 primary examples[‡], and many more supporting examples, has been built up. While the primary examples from this corpus may contain biases or omissions, we believe it represents a wide-ranging sample of kinetic typography techniques in actual practice.

To validate that our tool provides good coverage of the domain – ensuring that a wide range of expression is practically possible and important expressions are not hindered – we have tested it against the corpus primary examples. About ⅓ of the examples have been fully re-implemented in our tool, critical effects from others have been implemented, and we have carefully cataloged the set of effects used in the remainder.

With one exception, we found that all effects used in the examples are readily supported by the kinetic typography engine using only its current library. The exception concerns blur and motion blur effects. Although our architecture can support these effects, they are currently not part of the library because they cannot yet be delivered in real-time (the original works in question were done with offline tools which could spend arbitrary time rendering each frame).

COMPOSITE EFFECTS

The behavior library described thus far is low-level in nature – it allows control of all aspects of motion, but it also requires specification of many details. For many uses, this detailed control is not needed and higher-level approaches will be preferred. To support this, the system provides for *composite effects* – the equivalent of macros or subroutines that can be automatically applied to a body of text. Such composite effects take a text string along with a small set of parameters, and build the structure needed to carry out the effect with the text.

One common example of a composite effect is *RSVP*, which arranges for the rapid sequential delivery of each word of its text. A related but more interesting composite effect is *HopIn*.

As illustrated in Figure 11, *HopIn* words jump up from the bottom of the screen using squash and stretch. Each word rises to a random height (within specified limits), and then falls as if in gravity, fading out on the decent, so that it seems to disappear behind more recent words. Words travel a random horizontal distance (constrained to keep them within specified limits). However, each new word appears where the previous word would have landed. Hence, each word carries the viewer's eye to the position where the next word will appear. Finally, the appearance of the next word is delayed from the start of the current word by a time proportional to the square root of the length of the current word, adjusted by extra delays for punctuation. (Our informal experiments with overall

[‡] Parts of this corpus are available on the web starting at: [Ishi97] and several of the designs can be seen re-implemented in the video proceedings.

pacing seem to indicate that rates deemed comfortable for reading with standard RSVP delivery, also feel comfortable for HopIn delivery. However, considerably more study would be needed to verify this.)

Several additional composite effects are included in our current library. However, while we feel that the basic behavior and time filter library supplied with the kinetic typography engine is nearing a usably complete set which covers most needs, this is not the case for composite effects. Substantially more exploration is anticipated, and in fact, we expect to be inventing and adding new effects indefinitely.

EDITING INTERFACES

While this paper has considered work at an infrastructure level, it is important to also consider the editing interfaces that will be needed to make this infrastructure generally usable, and we are currently working on such interfaces. We see at least two distinct user populations for kinetic typography tools. One is the visual and interaction designers creating rich content. The other is the more general population who might want to make use of a tool to, for example, create email messages that convey more emotive content. Currently, we believe that two very different interfaces will be needed for these two populations, but that both will be supported well by our kinetic typography engine infrastructure.

Our early contextual investigation indicates that "professional" users desire a high degree of detailed control to achieve exactly the effects they want. For example, we recently observed a design student who spent three hours implementing the animation of a single word in order to achieve the effect he wanted. (This is a testament both to the tenacity of the student and the paucity of current tools.). For this population, we believe that the full power of our engine and library needs to be made available. At the same time, however, this population tends not to have strong programming skills and do not most naturally think in mathematical terms. This makes the "composition of functions" approach we describe in this paper problematical. However, we believe this can be easily overcome by structuring an interface around the manipulation of waveforms (which can be presented and manipulated in visual form).

For our second and more general population, we believe that users will need to be able to apply effects very quickly, but that the exact details will not need to be manipulable. For this population we are considering the use of libraries of parameterized schemas that text can be "plugged into" to achieve a fixed set of "canned" effects.

CONCLUSION

In this paper we have considered the range of expressive techniques useful for animating moving typographic forms. We have introduced an architecture for a kinetic typography engine, and a library of behaviors to accompany this tool. Through composition mechanisms, this relatively small library is able to cover much of the domain of kinetic typography (as demonstrated with respect to a substantial corpus of collected works.)

By taking a signal processing approach to animation, we were able to treat all time driven changes in a uniformly modular and extensible manner. Since each behavior can be viewed as a simple filter that modifies an incoming waveform to generate a new output waveform, behaviors can be combined and interconnected in multiple ways to produce highly complex animations. Each behavior can be quite simple and naïve making the library of behaviors easily extensible by a programmer. This approach also allows us to easily utilize the mathematical properties of real-time waveforms to assist the animator in generating effects such as secondary motion.

REFERENCES

[Bass59] Bass, S. (1959). Opening Credits to *North by Northwest*, Alfred Hitchcock, MGM, 1959. Available on the web at: http://www.twenty4.co.uk/on-line/issue001/project01/clips/nbynwest.rm

[Bass60] Bass, S. (1960). Opening Credits to *Psycho*, Alfred Hitchcock, Universal, 1960. Available on the web at: http://www.twenty4.co.uk/on-line/issue001/project01/clips/psycho.rm

[Bork83] Bork, A. (1983). "A Preliminary Taxonomy of Ways of Displaying Text on Screens." *Information Design Journal*, v3 n3, pp. 206-214.

[Brin01] Bringhurst, R. (2001) *The Elements of Typographic Style ver 2.4*. Hartely and Marks, Publishers. Point Roberts, WA. 2001.

[Chan93] Chang, B.-W., Ungar, D. (1993). "Animation: From Cartoons to the User Interface", *UIST93 Conference Proceedings*, pp.45-55.

[Chan98] Chang, B-W., Mackinlay, J., Zelleweger, P.T., and Igarashi, T. (1998). "A Negotiation Architecture for Fluid Documents." *UIST98 Conference Proceedings*, pp. 123-133.

[Cho99] Cho, Peter. (1999) "Pliant Type: Development and temporal manipulations of expressive, malleable typography". Masters thesis, Massachusetts Institute of Technology.

[Crys75] Crystal, D. (1975). *The English Tone of Voice: Essays in intonation, prosody, and paralanguage*. St. Martin's Press : New York.

[Duis86] Duisberg, Robert, (1986) "Animated Graphical Interfaces Using Temporal Constraints", *CHI'86 Conference Proceedings*, pp.131-136.

[Ford97] Ford, S., Forlizzi, J., and Ishizaki, S. (1997). "Kinetic Typography: Issues in time-based presentation of text." *CHI97 Conference Extended Abstracts*, pp. 269-270.

[Gren96] Greenberg, S., Gutwin, C., and Cockburn, A.

(1996). “Awareness through fisheye views in relaxed WYSIWIS groupware. *Graphics Interface 96 Conference Proceedings*, pp. 28-38.

[Hieb92] Hiebert, K. (1992). *Graphic Design Processes: Universal to Unique*. Van Nostrand Reinhold: New York.

[Huds91] Hudson, Scott. (1991) “Incremental Attribute Evaluation: A Flexible Algorithm for Lazy Update”, *ACM Transactions on Programming Languages and Systems*, v13, n3, pp. 315-341.

[Huds93] Hudson, S.E. and Stasko, J.T. (1993). “Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Abstractions.” UIST93 Conference Proceedings, pp. 57-67.

[Ishi96] Ishizaki, S. (1996). “Multiagent Design Systems: Visualization as an emergent behavior of active design agents.” *CHI96 Conference Proceedings*, pp. 347-354.

[Ishi97] Ishizaki, S. (1997) “Kinetic Typography.” Available on the web at: http://www.cmu.edu/cfa/design/kdg/kt/

[Ishi98] Ishizaki, S. (1998). “On Kinetic Typography”, *Statements, the Newsletter for the American Center for Design*, v12 n1, pp. 7-9.

[Kand98] Kandogan, E. and Schneiderman, B. (1998). “Elastic Windows: A hierarchical multi-window World-Wide Web browser”. *UIST 98 Conference Proceedings*, pp. 169-177.

[Lass87] Lasseter, J. (1987). “Principles of Traditional Animation Applied to 3D Computer Animation”, *SIGGRAPH’87 Conference Proceedings*, pp. 35-44.

[Lewi99] Lewis, J.E. and Weyers, A. (1999). “ActiveText: a method for creating dynamic and interactive texts.” *UIST’99 Conference Proceedings*, pp. 131-140.

[Masu95] Masui, T., Minakuchi, M., Borden IV G.R., and Kashiwagi, K. (1995). “Multiple View approach for smooth information retrieval.” *UIST’95 Conference Proceedings*, pp. 199-206.

[Mill87] Mills, C. and Weldon, L. (1987). “Reading Text From Computer Screens.” *ACM Computing Surveys*, v19 n4, pp. 329-358.

[Myer96] Myers, B. A., Miller, R.C., McDaniel, R., Ferrency, A. (1996). “Easily Adding Animations to Interfaces Using Constraints”, *UIST’96 Conference Proceedings*, pp. 119-128.

[Neuw92] Neuwirth, C.M., Chandhok, R., Kaufer, D.S., Erion, P., Morris, J., and Miller, D. (1992). “Flexible diff-ing in a collaborative writing system.” *CSCW’92 Conference Proceedings*, pp. 147-154.

[Pash98] Pashler, H., (1998). *The psychology of attention*. MIT Press : Boston, MA.

[Pott84] Potter, M. (1984). “Rapid Serial Visual Presentation (RSVP): A Method for Studying Language Processing.” *New Methods in Reading Comprehension Research.* Mahwah, NJ: Lawrence Erlbaum, pp. 91-118.

[Rose98] Rosenberger, Tara. (1998) “Prosodic Font: The space between the spoken and the written”. Masters thesis, Massachusetts Institute of Technology.

[Smal89] Small, D. (1989). “Expressive Typography”. Master’s Thesis, Media Arts and Sciences, Massachusetts Institute of Technology.

[Smit95] Smith, M. (1995). *Engaging Characters — Fiction, Emotion and The Cinema*. : Oxford, England.

[Thom81] Thomas, F., Johnston, O. (1981). *The Illusion of Life: Disney Animation*. Hyperion Books : New York, NY:.

[Wong95] Wong, Y.Y. (1995). “Temporal Typography: Characterization of time-varying typographic forms.” Master’s Thesis, Media Arts and Sciences, Massachusetts Institute of Technology.