Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Performance Timing for Keyframe Animation

S.C.L. Terra and R.A. Metoyer

Oregon State University, Corvallis, Oregon

Abstract

Keyframing is a standard technique for generating computer animation that typically requires artistic ability and a set of skills for the software package being used. We are interested in addressing the needs of the novice animator who is not necessarily artistically skilled or familiar with keyframing interfaces. From our experience observing novice animators, it is clear that setting keyframe values is straightforward while specifying the keyframe timing is difficult and often time consuming. We present a novel method for novice users to time keyframes using gestures without changing the motion itself. The key to our approach is the separation of specification of keyframe values from the specification of keyframe timing. Our approach allows the user to "act-out" the timing information using a simple 2D input device such as a mouse or pen-tablet. The user's input is analyzed and features of the user's input are mapped to features of the keyframed motion. The keyframes are then distributed in time according to the timing of the user's input path. We have implemented the approach as a plugin to the AliasWavefront Maya modeling and animation package. We demonstrate the approach on several example scenes and discuss its strengths and limitations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

As 3D computer animation tools become more accessible to novice users, it becomes important that our interaction techniques address the needs of these users. One of the most fundamental motion generation techniques is keyframing. Within the context of keyframing, the talented animators of Walt Disney developed the 12 principles of animation for creating compelling motion [TJ81]. John Lasseter later discussed these principles in the context of 3D computer animation [Las87]. According to Lasseter, the timing principle is extremely important

> " . . . it gives meaning to the movement–the speed of an action defines how well the idea behind the action will read to an audience. It reflects the weight and size of an object, and can even carry emotional meaning."

In our observations, timing an animation is often the most difficult part of the process for novice users. Using an animation suite such as AliasWavefront's Maya, we have found that novice users do not have much difficulty in setting the spatial values of the keyframe. For example, it's fairly easy for a novice user to set the keyframes for a bouncing ball in terms of where it will move in space. However, we have found that properly spacing the keyframes *in time* and setting the velocity profiles are often only done well by someone with an artistic sense for motion specification. We believe that this difficulty is not caused by an inability to imagine the timing, but by an inability to convey the timing using the provided interfaces. On the other hand, we note that most people find it fairly intuitive to "act out" the motion or mimic the desired motion with their hands. For example, most of us could probably mimic a bouncing ball motion by tracing the path with our fingertips in mid air. We therefore decided that the novice user needed a similar way to act out their desired animation timing.

In this paper, we present a novel method for capturing the user's desired timing, correlating it to the character's motion path, and adjusting the timing of the animation to reflect the user's desires. We first capture the user's timing by requiring her to act out the motion of a single animated object that exhibits translation. Next, we determine correspondence between the user's acted motion and the object's motion path. We then distribute all keyframes in the object's animated channels according to the acted motion to obtain a completely new timing without modifying the spatial character-

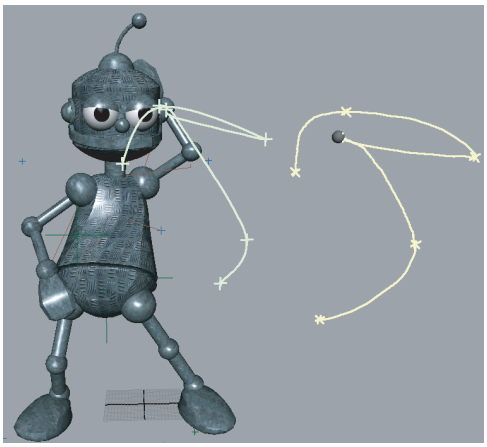

Figure 1: *Our timing technique can be applied to any motion that exhibits translation. In this case, the user has chosen the inverse kinematics handle of the arm as the target for animating Hugo. The motion path is shown along with the user's sketched timing path.*

istics of the motion. The goal is to create an intuitive tool for the novice animator and a rapid prototyping tool for skilled and non-skilled animators alike. Figure 1 shows a keyframed character alongside the user's timing performance path for the small arm movement.

2. Background

The need for intuitive animation interfaces has been recognized for many years now. In 1971, Baecker created the Genesys Computer Animation System for sketching animations. Using a tablet device, the animator sketched curves that represented the path and dynamics (timing) of the 2D animation [Bae71]. In 1995, Balaguer and Gobbetti introduced an integrated environment for creating animations using 3D sketching [BG95, GB95]. The system was designed to allow for the expressiveness of a real-time motion capture system in a general animation system. The environment allows users to sketch animations using a 3D device, controlling both spatial and timing aspects. A data reduction step fits a curve to the user data, resulting in a clean animation. In this paper, we intentionally separate spatial specification from timing in order to make use of currently effective techniques for specifying 3D motion paths through keyframe values. We then provide a method for timing the animation within a standard 2D mouse-based interface.

In recent years, several researchers have developed sketching interfaces for controlling animation. Dontcheva and her colleagues present the layered acting approach for creating character animations [DYP03]. In this system, user actions are captured and mapped to the character to create motion. The motion can be supplied in stages and layered to produce the final motion. This system also requires substantial equipment for capturing user input and again concentrates on both spatial and timing aspects of the motion. Our approach is very similar in that we allow users to directly control features of the "character" by capturing motion. However, we are primarily concerned with doing so with 2D devices and without affecting the spatial aspects of the motion.

In 2000, Laszlo et al. presented an entertaining interface for controlling interactive physical simulations by directly mapping mouse motion to the desired joint angles and by providing discrete control actions via keystrokes [LvF00]. The goal was to build on the user's intuition about how the motion should be performed and to use this intuition to drive the character directly. We have very similar goals but we are concerned with traditional keyframed motion and not physical simulation.

Most recently, Thorne and his colleagues presented a sketch-based technique for creating character motion [TBvdP04]. In this system, the user not only sketches the character to be animated, but also sketches motion curves that are segmented and mapped to a parameterized set of output motion primitives that reflect the timing of the input sketch. Our approach reflects the timing of the input sketch as well, but rather than apply it to particular characters with particular primitives, we apply our motion directly to keyframed motion values.

Popovic et al. discusses an interface for sketching the desired path and timing for rigid body simulations [PSE03]. An optimization step then produces physically plausible motion that reflects the desired path and timing from the sketch. An approach that is similar to ours in nature but not interface is Sampath's NUKE plugin for Maya. This plugin allows users to retime keyframed animations by scrubbing the animation time slider to reflect the desired timing [Sam99]. Real-time motion capture is the ultimate goal in performance animation. Unfortunately, motion capture systems are expensive and often cumbersome especially for animation of simple characters. A thorough overview of motion capture for animation is presented by Gleicher [Gle99].

3. "Acting Out" Timing Information

Using a standard animation interface, the user must first keyframe the motion that she wants to animate. In doing so, the user may completely ignore how the keyframes are spaced in time. The result is an animation where the object moves correctly in terms of the spatial channels, but for which the timing is arbitrary.

The key to our approach is allowing the user to "act out" the timing information. To do so, the scene must contain a spatial component for the user to mimic. This spatial component must exhibit values that vary in at least one of the translation channels (x, y, or z). This is the component that

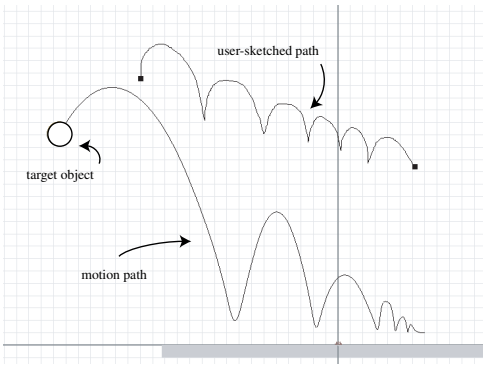

Figure 2: *In this example, the user chose the bouncing ball as the target component. Above the component's path, we see the timing path sketched out by the user when acting out the timing. Note that the user-sketched timing path is similar in shape, but not exact.*

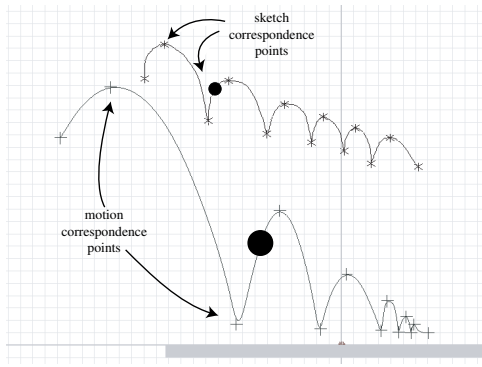

Figure 3: *An example of correspondence calculation to match features in the two paths: the target component motion path and the user's sketched timing path.*

the user will mimic while acting out the motion. This requirement does not mean that only objects with translations can be used. For example, a single rotating sphere does not exhibit translation. However, one could select a single vertex on a rotating sphere and use its translation as the motion to be "acted out". This approach could also be used to time an object's scale changes. The user begins by selecting an expressive object in the scene from any one of the camera viewpoints. For a simple scene, such as a bouncing ball, the user would simply select the ball itself as the target component. In a more complex scene, such as a maestro directing an orchestra, the animator could choose the position of the hands or the tip of the conducting wand. The user should choose a target object and view that exhibits motion that she understands and for which she can mimic the motion.

The user then acts out the scene by sketching the desired timing directly in the window in which she is viewing the spatial path of the target component. The user's timing path must be similar in shape to target motion path. As the user sketches, the system samples the input sketch resulting in time-stamped path samples. In effect, this is a simple form of motion capture. Figure 2 shows an example target component along with the user's sketched path as seen in the actual interface.

4. Mapping the Motion

Now that we have the user's desired timing in the form of the sketched path with time-stamped samples, the goal is to use this data to control the timing of the target component's motion. A naive approach would be to apply the resulting timed samples directly to the target component's motion. This, of course, would modify the spatial motion of the object. Instead, we match features of the sketched path to the target component motion path. We then distribute the keyframes in time according to the distribution of the features in the user's sketched timing path.

This matching is done for the purpose of determining which point on the motion path the user was mimicking when she was sketching out a particular point on the screen during the performance. Since the sketch is in no way guaranteed to be the same length, shape, or size as the motion path (indeed, that would be the very rare exception), we must correlate the two paths in a way that seems natural to the user. We will present two methods that we have experimented with for matching the paths.

4.1. Path Matching

The first method involves finding the local minimum and maximum points in both the horizontal and vertical directions in the current view. We call these features the peaks of the path and we use these peaks as our correspondence points between the two paths. We examine three samples at a time to determine whether the path has changed direction along a particular axis. If we find a directional change, and if the change is larger than a threshold value, we place a peak at the location (Figure 3). The threshold value allows us to ignore noise from the user where noise is defined as small motion adjustments that are not intended to be peaks. This peak finding method is simple and computationally efficient.

We have also experimented with more robust methods of shape matching such as a discrete, dynamic programming solution for curve matching [SB94]. This technique is typically more robust but also more computationally expensive and therefore less responsive than the peak finding algorithm. For this solution, we first align the two paths in space. Our cost function to be minimized is defined as the distance that a sample must be moved from the first path in order to place it on the second path.

In cases where neither the peak method nor the dynamic

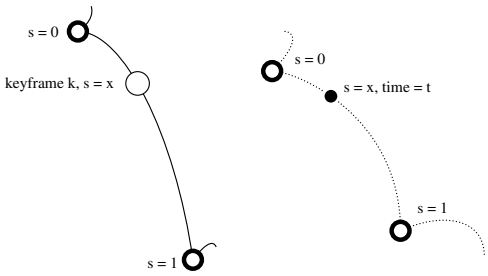

Figure 4: *In the example shown above, the path on the left represents the target component path. The arclength, s, is normalized for each segment where a segment is defined as the portion of the path between two peaks. The path on the right represents the user's sketched timing path. To find the correct time for the keyframe, k, shown on the target path, we first determine the arclength traveled to that keyframe* $(s = x)$ *We then use this arclength value to index into the timing path and find the time stamp value, t that corresponds to arclength,* $s = x$*. The keyframe in the target component motion path is then assigned this time value.*

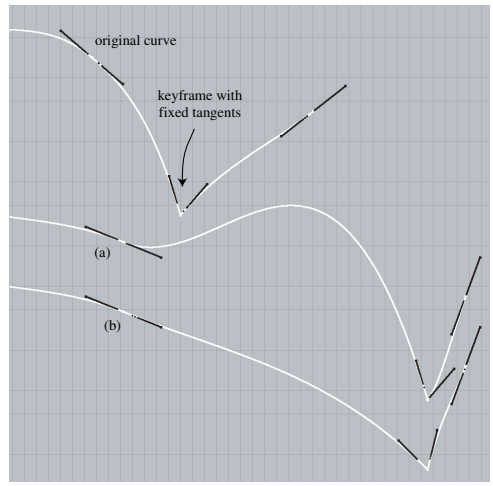

Figure 5: *The top path represents the original animation path for the target object. The horizontal axis is time while the vertical axis is the value of the animation channel (ie. translationX). In this example, imagine that the user has specified fixed tangents for the spline that represents the animation curve. Path a shows the results of moving the keyframes in time without adjusting the tangents–the motion is clearly changed. Path b shows the results after we modify the tangents. The tangents are modified by adjusting the angles to reflect the relationship between the similar triangles.*

programming approach produce the desired results, the user always has the option to manually edit and assign peaks as we will discuss in Section 5.

4.2. Mapping the Timing

Now that we have established correspondence between the target component motion path and the user's sketched timing path, we must adjust the target component's keyframes to reflect the user's timing while preserving the original spatial motion path. We first parameterize each path according to arclength. The paths are broken into segments, where a path segment is defined as the portion of a path between two peaks. For each segment of the motion path we identify the existing keyframes. For each of the keyframes, we compute the arclength traveled on the target component motion path to that particular keyframe. This arclength value is then used to index into the user's sketched timing path to find the time stamp value that corresponds to this particular arclength. This time value is then assigned to the keyframe in the target component motion path (Figure 4). This approach distributes the keys across the component motion path according to the user's desired timing.

The underlying motion curve is represented with a spline. In some animation systems, spline tangents can be fixed such that moving the keyframes in time will cause the overall motion to be changed. If this is the case, we must adjust the tangent vectors so that the overall motion path remains unchanged (Figure 5). The final step involves adjusting the tangent angle of shifted keyframes to maintain the original motion given the adjustment in time.

To synchronize the motion for all parts of an articulated character, the user must simply apply the same mapping to each attribute that she wishes to animate. In our system, this is done by adding those attributes to a combined character, performing the timing for a single target object in the combined character group, and mapping the timing of that object to the entire group.

5. Editing

Because the feature correspondences cannot always be identified, we also provide the ability to edit the peak information. Users can remove peaks as well as add peaks.

We also give the user the ability to apply timing to various keyframed objects independently. This means that the sketched motion can be applied to all objects simultaneously as described above, or, the user can provide different timing sketches for different objects. This allows the user to create the animation in a layered fashion.

6. Results

To demonstrate the effectiveness of our approach, we created a plugin for AliasWavefront's Maya modeling and animation package. Using Maya's keyframing facilities and our plugin, we created several animated sequences. We were able to quickly and easily create timing for various scenes that had already been keyframed.

For the first example, the user was asked to time a bouncing ball to be physically correct. In the second example, the user was asked to time a bouncing cursor to annotate the words of the song "Jingle Bells". In the third example, the user modifies a keyframed articulated figure to walk with a limp. For the fourth example, the user animates a pencil character to exhibit several different timings for a scene where the pencil is looking around the environment. Finally, we demonstrate the use of our approach for animating a longer animated scene with a complex character.

In all cases, we found that this system made it possible to quickly and easily generate many timings for the characters in the scene. However, these studies were informal. We are currently in the process of designing a more formal user study to establish the effectiveness of the interface as compared to a traditional keyframing interface.

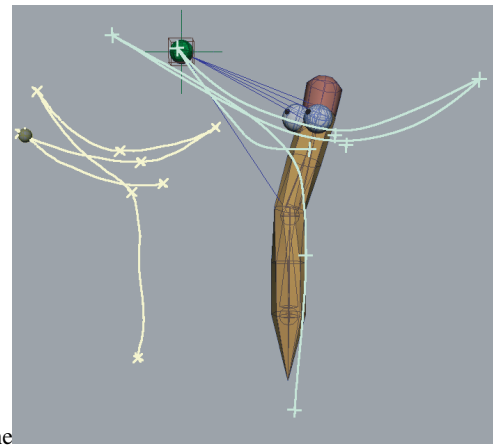

Figure 6: *In this example, the character's head is animated by translating an inverse kinematics handle. The user has chosen this handle as the target object for the timing performance. One could also choose to "act out" the motion of a single vertex on the surface of a character.*

7. Discussion and Future Work

We have presented a novel approach for generating timing information for keyframed motion. This approach is designed to help a novice user generate animations without necessarily being artistically talented and to allow even skilled animators to prototype motion quickly. We have demonstrated the resulting motion in several animated scenes with varying complexity.

Finding the correspondence points between the target object motion path and the user's sketched timing path is the most computationally complex part of our algorithm. The min/max peak finding algorithm is $O(n+m)$ where n is the number of samples in the target component motion path and m is the number of samples in the sketched timing path. For the dynamic programming solution, the running time is $O(n*m^2)$.

Longer animations present an interesting problem. In particular, the user should not be required to animate an entire scene in one performance. In our current implementation, the user can time and map several portions of the animation. In doing so, the user must be sure that she picks good ending points so that there are no apparent discontinuities in the motion. If the user overshoots an ending point or starts before a start point, the unwanted parts of the timing curve can be removed with the peak editing facility. Another option is to allow the user to specify an overlap window for two parts of an animation and apply a simple blend. This addition is left for future work.

The system performs well when the user's timing sketch is similar to the target component motion path. The simple peak finding algorithm and our dynamic programming solution are both rotation dependent and therefore will fail for sketches that are rotationally different from the target component motion path. However, we find that users do not object to using an orientation similar to that of the target component path when sketching their timing. There are also cases where the user's input does not necessarily resemble the shape of the target object. For example, consider a situation where the user visualizes the motion as having features that occur at velocity peaks or locations of similar curvature between the paths. We are investigating dynamic programming solutions with cost functions that match various properties such as acceleration or velocity to the peaks in the target object's motion.

We currently require an object to have some kind of spatial signal to "act out". For example, in Figure 6, the user is animating the rotation of the character's head via the translation of a bone's inverse kinematics handle. This makes "acting out" the change in an attribute such as color currently impossible. One extension would be to map an attribute, such as color, to an artificial spatial signal that exhibited peaks at the maximum and minimum values of the attribute signal. The user could then act out the timing for this artificial curve and map it to the attribute. This is left as future work.

In this paper, we have presented several examples created using 2D input devices such as a mouse or a pen tablet. Although we are mostly interested in this domain, we can also apply our algorithms to 3D timing paths. For example, one could imagine a vision based system for tracking a single marker to be placed at the end of a timing wand. The user could act out the timing by manipulating the wand as an extension of her hand. This approach is similar to work in the audio synthesis domain [SKG00]

An alternative to acting out the timing by drawing paths is to "act out" the motion by clicking the input device, such as a mouse, when the peaks should occur. This approach would give us the relevant information needed to redistribute the keyframes. However, possibly valuable information is lost

in this approach. For example, with our current technique, we could also analyze the users input to understand velocity changes between the keyframes and use this information to adjust the animation curves to create effects such as ease-in ease-out. For example, in a scene with a single sphere and two key frames, currently, we can only generate a constant velocity motion between those two keys. If we were to analyze the velocity profile, we could adjust the tangents of the motion curve to reflect the velocity profile of the user between those two keyframes. This is also left for future work.

Although we have demonstrated success with the interface we are currently conducting usability studies with novice users to identify weaknesses, refine the interface, and observe how well the interface facilitates animation by novice users. User studies for comparing our approach to traditional keyframe timing techniques are also planned.

8. Acknowledgments

The authors would like to thank Laurence Boissieux and INRIA for use of the Hugo model (Figure 1). The present 3D Model was created by Laurence Boissieux within the French National Institute for Research in Computer Science and Control (INRIA), Domaine de Voluceau, Rocquencourt - BP 105, 78153 Le Chesnay Cedex, France.

This work was supported in part by NSF CCR-0237706.

References

[Bae71] BAECKER R.: A demonstration of the genesys computer animation system. In a 60 minute colour sound film, The Dynamic Image, 1971. Dynamic Graphics Project, University of Toronto, 1987.

[BG95] BALAGUER J., GOBBETTI E.: Sketching 3D animations. *Computer Graphics Forum 14*, 3 (1995), 241–258.

[DYP03] DONTCHEVA M., YNGVE G., POPOVIC Z.: Layered acting for character animation. *ACM Transactions on Graphics 22*, 3 (2003), 409–416.

[GB95] GOBBETTI E., BALAGUER J. F.: An integrated environment to visually construct 3d animations. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM Press, pp. 395–398.

[Gle99] GLEICHER M.: Animation from observation: Motion capture and motion editing. *Computer Graphics 33*, 4 (1999), 51–54.

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 35–44.

[LvF00] LASZLO J., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 201–208.

[PSE03] POPOVIC; J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 4 (2003), 1034–1054.

[Sam99] SAMPATH J.: Nuke. http://members.tripod.com/sjagannathan/nuke/, 1999.

[SB94] SERRA B., BERTHOD M.: Subpixel contour matching using continuous dynamic programming. In *Proceedings of the International Conference onComputer Vision and Pattern Recognition* (1994), pp. 202 –207.

[SKG00] SEGEN J., KUMAN S., GLUCKMAN J.: Visual interface for conducting virtual orchestra. *IEEE International Conference on Pattern Recognition 1* (2000), 1276–1279.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: An interface for sketching character motion. *To appear in ACM Trans. Graph.* (2004).

[TJ81] THOMAS F., JOHNSTON O.: *Disney Animation - The Illusion of Life*. Abbeville Press, 1981.