A New Static Depiction and Input Technique for 2D Animation

Shin Takahashi
Dept. of Computer Science
University of Tsukuba, JAPAN
shin@cs.tsukuba.ac.jp
http://www.iplab.cs.tsukuba.ac.jp/~shin/

Yoshikazu Kato Etsuya Shibayama
Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology, JAPAN
{yoshika0,etsuya}@is.titech.ac.jp

Abstract

This paper describes an extension of our previous technique for making 2D animation by drawing effect lines, which is a popular technique that represents motions and emotions in cartoons. It incorporates multi-step effect lines in one keyframe and constraints on the motions of objects. The new features improve the user's expressive power for specifying 2D animation interactively.

1. Introduction

We previously proposed a technique for making 2D animation by drawing *effect lines* [3]. An effect line is a popular technique that is used in cartoons to represent motions and emotions. KOKA is a prototype system for making 2D animations interactively by drawing effect lines. In KOKA, the user can set animation effects and their parameters by drawing primitive effect lines alone or in combination around target objects. These effect lines are recognized by the system and displayed in each keyframe so that the user can easily understand the animation effects that are assigned to the objects in keyframes, even from a static representation. Figure 1 shows a snapshot of the KOKA system. We designed eight primitive effect lines, and conducted informal user tests.

Through the experience of making various animations with KOKA, we found that effect lines and their combinations are not enough to deal with the wider range of animations used in various applications. For example, it is sometimes tedious to draw effect lines when assigning similar motions to similar objects in a keyframe. A simple animation sometimes requires a number of keyframes, which makes it difficult for the user to develop long animations. It is also difficult to draw exact lines with pen strokes to set precise motions.

This paper describes new techniques for depicting motions statically based on our technique using effect lines.

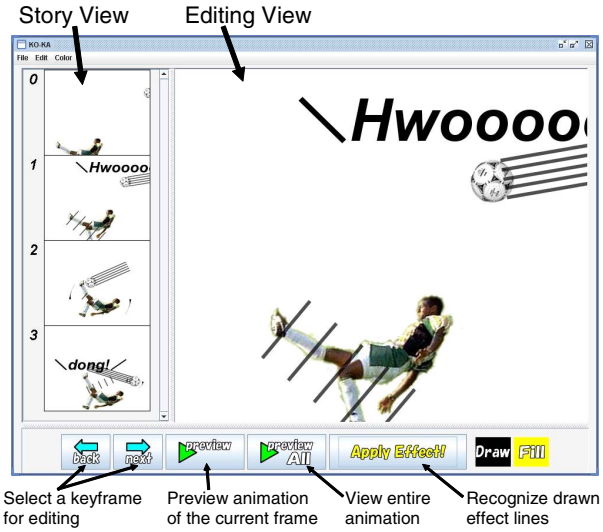

Figure 1. A snapshot of the KOKA system

Our ultimate goal is a pen-based programmable 2D-animation environment for interactive animation. As a step towards this goal, we designed two new techniques: multi-step effect lines in one keyframe, and constraints on the motions of objects. These techniques increase the expressive power of effect lines, and reduce the number of keyframes required to construct complex animations. In the following, we describe these two features, and discuss their expressive power.

2 Multi-step effects in a keyframe

KOKA is implemented as a keyframe-based system. The storyboard contains and shows the keyframes generated when the user draws effect lines. The animation between two successive keyframes corresponds to the effect lines in the second keyframe.

Since a single animation effect is a simple fragment of

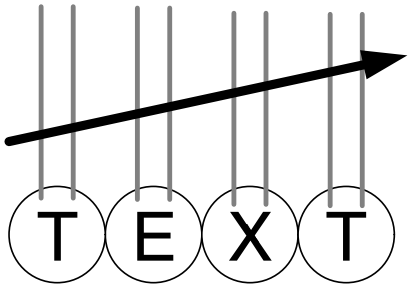

Figure 2. An example of multi-step motion

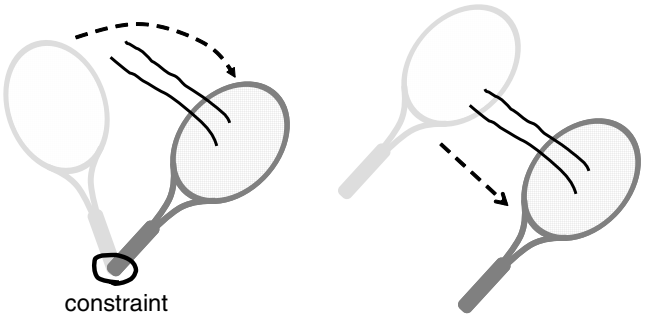

Figure 3. Constrained motion

animation, a long animation requires a long sequence of keyframes and effect lines. This makes it difficult for the user to understand the overall animation. It would be convenient to show several sequential actions in a keyframe. This could be regarded as incorporating some abstraction into the sequence of keyframes, making it easier to grasp an overview of a long animation.

To enable the user to assign multi-step effects to the objects in the keyframe, the user must be able to specify the order of actions. To do this, we provide *sequencing arrows*. A sequencing arrow is drawn over the effect lines. The timing of the actions represented by the effects is ordered according to the sequencing arrows. Figure 2 shows an example of a sequencing arrow. The arrow in the figure is drawn on the movement effect lines. This specifies the order in which the effects are to be executed. That is, the movement effects are executed sequentially in order from left to right. In this example, four characters fall downward sequentially from left to right. Without a sequencing arrow, this simple sequence requires five keyframes, but it can be represented in one frame with a sequencing arrow. Sequencing arrows help the user to organize the keyframes of long animations.

3 Constraints

We incorporate *constraints* into our system. Constraints are useful in two ways. First, a constraint mechanism can improve the expressive power of effect lines for depicting animations. Second, it is helpful to specify various parameters of motions precisely and exactly.

Concerning the first point, the user can specify constraints that change the motion of objects with effect lines. For example, Figure 3 shows an example of constrained motion. The racket to the right in the figure moves in a straight line because its movement effect line is set. By contrast, the racket to the left is constrained at its end. The small circle at its end means that this point should not move. Therefore, although both rackets use the same kind of effect line, one moves circularly, as shown in the figure. Strokes for constraints are drawn in a different mode. They are represented as thick strokes in the figures.

Constraints can also be used to group the motions of objects. In Figure 4, the user draws a stroke over the three stars and the effect lines for a movement. This means that they should move in the same way, as if they are assigned the same effect lines. A thick stroke over multiple objects means they should move together. This may be achieved

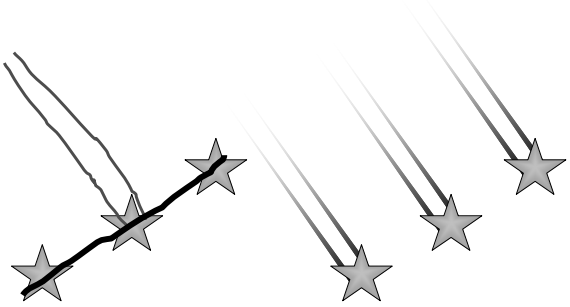

Figure 4. grouping

by the grouping function usually provided in most drawing editors. Since this group movement is often temporary, i.e., objects often move in different ways in subsequent scenes, grouping using an editor is inconvenient.

Another type of constraint is a timing constraint. The original KOKA system had a very simple timing strategy. Actions in a frame were executed at once, and then actions in the following frames were executed in order. However, a sequencing arrow enables multi-step motions in a frame, which demands the coordination of action timing. For this purpose, we designed a timing constraint. For example, in Figure 5, two balls collide at the center. The timing of the collision is specified by the markings at the center. Each circle specifies the timing in the movement of the objects and the line connecting two circles indicates that they are the same point in time. Figure 6 shows another example. In this figure, the bat hits the ball. The motion of the bat is expressed as a constrained motion. The movement of the ball is depicted as multi-step effects on the ball. The point

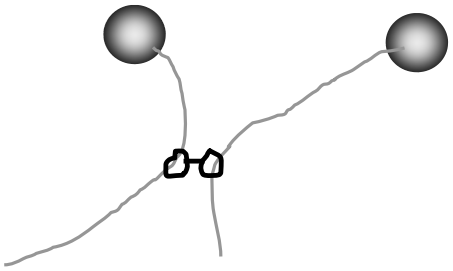

Figure 5. Time constraint

of contact is specified by the timing constraint. Each circle specifies a time in the movement of the objects and a line connecting them indicates that they are at the same point in time.

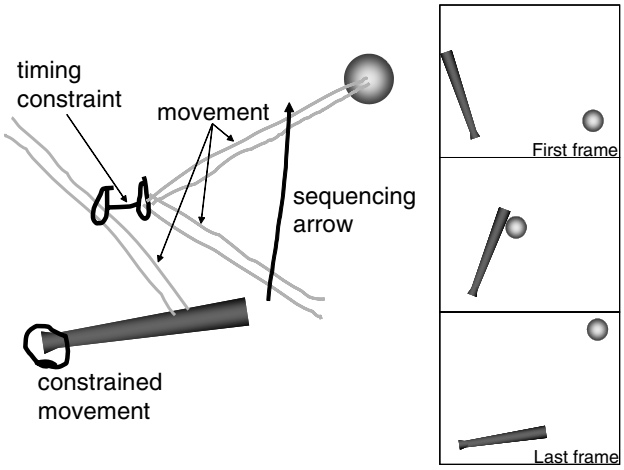

Figure 6. Hitting a ball with a bat

4 Discussion and Related Work

We are considering how to extend our effect-line-based animation authoring system to a kind of visual programming system that can *program* complex animation. Our effect-line-based method is suited to programming animation; because effect lines explicitly represent motions and changes of objects in static representation, we can annotate them to refer to the motions directly. For example, sequencing arrows are drawn on effect lines to order the specified actions. We can also add constraints on motions by drawing constraint lines on effect lines. Such meta-level operations are possible because motions are represented explicitly as annotatable visual objects. This is an important feature of our technique. For example, in rule-based systems, such as AgentSheet [4] and Viscuit [1], users specify animation with before and after pictures that describe how objects in the picture change. It is difficult to annotate motions in such systems directly. This is also true for other methods, such as animated icons, onionskins, and motion blurs.

Motion Doodles [5] is a sketching system for making 3D character animations. The user first sketches characters in a specific manner, and then inputs their motions by sketching a line. The system defines a cursive alphabet for motions. Each gesture in the alphabet is associated with a predefined motion so that the user can sketch various motions by sketching the defined gestures. This is similar to our work in that it uses a gestural line to specify the motion. It differs in that Motion Doodles makes animations instantly and does not aim to depict animations statically. Our work aims to integrate the gestural input and static representation of animations. This feature is important when extending our system to a programming environment for interactive animations.

5 Summary and Future Work

We have proposed a new input and depiction technique for 2D animation based on effect lines, which we are extending to a more programmable 2D animation authoring system. As a step toward this, we added two functions to the KOKA system: multi-step effects and constraints. We are also designing a user-definable macro function, but have not discussed it here owing to space restrictions. These techniques are useful for making various kinds of animation. We are currently implementing the new KOKA system based on the Chorus constraint solver [2], which can solve hierarchical non-linear constraints at interactive speed.

References

[1] Y. Harada and R. Potter. Fuzzy rewriting - soft program semantics for children. In *IEEE Symposium on Human Centric Computing Languages and Environments*, pages 39–46, 2003.

[2] H. Hosobe. A modular geometric constraint solver for user interface applications. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST2001)*, pages 91–100, 2001.

[3] Y. Kato, E. Shibayama, and S. Takahashi. Effect lines for specifying animation effects. In *the Proceedings of The IEEE Sysmposium on Visual Languages and Human-Centric Computing (VL/HCC'04)*, pages 27–34, September 2004.

[4] A. Repenning and T. Samner. Agentsheets: A medium for creating domain-oriented visual languages. *IEEE Computer*, 28:17–25, 1995.

[5] M. Thome, D. Burke, and M. van de Panne. Motion doodles: an interface for sketching character motion. In *Proceedings of the 2004 SIGGRAPH Conference*, pages 424–431, 2004.