K-Sketch: A "Kinetic" Sketch Pad for Novice Animators

Richard C. Davis
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
rcdavis@eecs.berkeley.edu

Brien Colwell, James A. Landay
Computer Science and Engineering
DUB Group, University of Washington
Seattle, WA 98195
xcolwell@gmail.com, landay@cs.washington.edu

ABSTRACT
Because most animation tools are complex and time-consuming to learn and use, most animations today are created by experts. To help novices create a wide range of animations quickly, we have developed a general-purpose, informal, 2D animation sketching system called K-Sketch. Field studies investigating the needs of animators and would-be animators helped us collect a library of usage scenarios for our tool. A novel optimization technique enabled us to design an interface that is simultaneously fast, simple, and powerful. The result is a pen-based system that relies on users' intuitive sense of space and time while still supporting a wide range of uses. In a laboratory experiment that compared K-Sketch to a more formal animation tool (PowerPoint), participants worked three times faster, needed half the learning time, and had significantly lower cognitive load with K-Sketch.

Author Keywords
Animation, sketching, pen-based, informal user interfaces.

ACM Classification Keywords
H5.m. H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

INTRODUCTION
Research into "easy" animation has produced many new tools and techniques in recent years. Some support specific tasks, such as studio-quality production [5, 17], classroom examples or exercises [1, 12]. Others have less specific tasks in mind [14, 19, 20, 22]. Unfortunately, no tool is fast enough for sketching ideas, simple enough for novices, and powerful enough to handle a wide variety of tasks.

Borrowing ideas from informal interfaces [7, 11] and demonstration-based animation systems [2], we have developed an informal, 2D animation system called K-Sketch, the "Kinetic" Sketch Pad. K-Sketch is a pen-based system that relies on users' intuitive sense of space and time while still supporting a wide range of uses. K-Sketch animations are often rough, but they are still useful in informal situations and as prototypes of formal animations.

CHI 2008, April 5–10, 2008, Florence, Italy.

The goal of this project has not been to design novel interaction techniques but rather to focus on high-level choices about tool features. Thus, we conducted field studies to find out how an informal animation tool might be used and whether or not it could be made general-purpose. From these interviews with nineteen animators and would-be animators, we compiled a library of 72 usage scenarios for an animation system. In an earlier workshop paper [4], we presented preliminary results from this fieldwork. Here, we analyze these results in more detail and describe a novel optimization technique that enabled us to make K-Sketch's interface simultaneously fast, simple, and powerful.

Our evaluations show that K-Sketch has come a long way toward accomplishing its goal. In a laboratory experiment that compared K-Sketch to a more formal novice animation tool (PowerPoint), participants worked three times faster, needed half the learning time, and reported significantly lower cognitive load with K-Sketch. Participants also reported that K-Sketch felt easier and faster, that they were no less comfortable showing their animations to others, and that they were significantly more comfortable creating animations in front of others using K-Sketch.

We begin by reviewing our interviews with animators and with non-animators. This is followed by an analysis of the library of usage scenarios we collected and a description of our interface optimization technique. We then present K-Sketch and the evaluations we conducted. We close with related work, conclusions and future work.

INTERVIEWS WITH ANIMATORS
Since many novice animators wish to do what experienced animators do, we began our field studies by interviewing eight experienced animators to see how an informal tool would fit in their work process. Six participants were professional animators (1 with Flash, 5 with other media) with an average of 10 years of experience. One of these also taught animation. The other two animators were Computer Science graduate students with much less experience who produced animated conference presentations. Though the range of participants was broad, commonalities did emerge.

Interviews were structured around the following questions:
- Describe the steps in your work process. Give more detail on the early stages and the parts that involve sketching.
- What hardware/software tools do you use in your work?

During the interviews, we recorded how participants go about the various steps in their process, collecting sketches, photographs, animations, and video of the animators at work whenever possible. There was variety in the animators' work processes. One artist did much of her work in clay with stop-motion photography, but produced early versions of her work with drawings that were animated in Adobe AfterEffects. Several other artists worked with similar tools, though some drew out all frames by hand (as in traditional, cel-based animation). Our two students took a vastly different approach, coding their animations in Slithy [24], an animated presentation language.

Six mentioned prototypes as a key step in their process. Traditional animators begin prototyping with paper sketches, then move to animated "character sketches" or other short timing tests, and then move to storyboards. Then, many animators build videos (which some call "animatics") that show storyboard frames in sequence with rough timing and sound tracks. Animatics are important for the animators' creative processes, but they are also an important tool for communicating project status to clients.

Other participants had different approaches to prototyping. Animator 7 was so experienced with Flash that she was often able to mock up animations directly in the tool after only a few sketches. Animator 4 worked exclusively in Slithy and was so proficient that he sometimes skipped the storyboarding phase entirely. Animator 5 was less proficient in Slithy, however, and expressed a need to do more prototyping, though he was not sure how.

At some point in each interview, we described possible designs for a rough animation tool, suggested ways that the animator might be able to use such a tool in their work process, and noted their reactions. Most were interested in such a tool as a prototyping aid. Animator 8 also expressed interest in using such a tool for finished works. Animator 3 taught animation classes for children and said that our demonstration-based approach matched very closely with children's intuition. Her students frequently "act out" the actions of characters in front of the camera.

Occupation	Domain	Animation Descriptions
1. Education Student	Biology	Student exercise: meiosis
2. Education Student	Physics	Planetary motion
3. Mech. Eng. Prof.	Eng.	Gears, molecular shifting
4. CS Grad. Student	Dance	Contra dance moves
5. Chemistry Prof.	Chem.	Particle collisions, rusting reaction, battery reaction
6. Researcher	Eng.	Machine tread motion
7. Math Instructor	Math	Cantor set construction
8. Reading Tutor	Reading	Fun animations to motivate
9. Aeronautical Eng.	Eng.	Robot arm following a path
10. Manager	Eng,	Box sliding into casing
11. Researcher	Geology	Etalon noise

Table 1: Summary of interviews with non-animators.

These interviews show that informal animations can play an important part in the development of more formal animations. In the following section, however, we will see that many novices do not require a formal end result.

INTERVIEWS WITH NON-ANIMATORS

As our project progressed, we encountered many people who did not create animations but were looking for fast, easy ways to create them. To better understand their needs, we recorded these conversations in a structured way. We recorded interviews with people who met these criteria:

1. They must describe animations they wish to create in sufficient detail for us to create them.
2. There must be a plausible reason why they do not already produce the animations they describe.
3. The animations must support a specific task.
4. There must be a plausible reason why the animation is *necessary* to accomplish that task.

We believed these criteria would lead us to people who could give us a clear picture of the needs of inexperienced animators. Table 1 describes those who met our criteria. Most of the participants were teachers (or education students), engineers, or scientists seeking to explain a concept. Some teachers did not know where to find animations that fit their needs, and others wanted to customize the animations that they found. Participant 1 wanted her students to produce animations as a visualization exercise. Participant 8 wanted animations to entertain and motivate her students. Engineers and scientists were seeking to explain ideas in small, informal meetings (as in Figure 1). Participant 4 wanted to visualize a set of dance moves before directing dancers (Figure 2).

Most of our participants did not know how to go about creating the animations they envisioned. Only two (Participants 3 and 4) knew of useful domain-specific animation tools, but their need did not justify the effort required to learn these tools. Some participants believed general purpose animation tools would be prohibitively complex. Teachers had trouble devoting lesson planning time to learning or using these animation tools. Participant 1 was concerned that her students would waste studying time learning to use animation tools. Participants who worked in science and engineering said that the need for an animation would arise suddenly in meetings, and there was no hope of creating an animation at the last minute.

These interviews further convinced us that there is a need for informal animation tools that require very little time to learn or use. However, as the next section shows, our participants described a wide variety of usage scenarios, presenting us with a considerable design challenge.

LIBRARY OF USAGE SCENARIOS

As we looked more closely at the tasks suggested by our potential users, we saw a wide variety of subject matter,

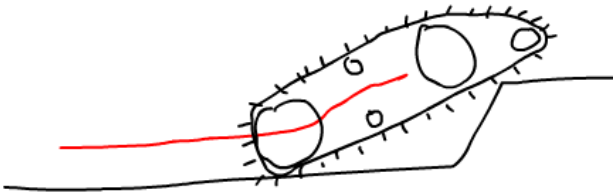


Figure 1: Construction equipment tread motion (non-animator 6). The tread traces out the red line as it moves.

level of complexity, and usage contexts. It seemed that the complexity of general-purpose tools might be necessary. In hopes that we could identify a small set of capabilities that would still support a wide range of animations, we gathered task scenarios into a library for deeper analysis.

Each scenario contains a description of objects and actions as well as a detailed description of the user and goal. Currently, the library includes 16 scenarios from non-animators and 27 scenarios from animators. Animator 3 also gave us 22 animations produced by children in her class. Many of these were rough animations. Finally, we created 7 scenarios that were significantly different from any in our library, giving 72 scenarios total.

In the remainder of this section, we discuss the patterns that emerged in these scenarios, starting with the users.

User and Goal Categories

All users fell into one of the following five categories. Each title is followed by the number (and percentage) of scenarios with a user in this category.

Amateurs: 31 (43%). These users are undertaking small creative tasks that may not involve animation directly. Amateur animators would use an informal animation tool to tell short stories, solve a problem, or share an idea.

Artists: 24 (33%). These users are undertaking larger creative tasks that involve animation. They would use informal animation to try out new ideas quickly, to prepare animated storyboards, and to share these with clients.

Teachers: 11 (15%). These users are working to impart knowledge to their students. It is likely that animation will be used to explain a dynamic concept, or to make course material more engaging. Education research is conclusive in showing these uses have positive educational impact [16].

Students: 1 (1%). These users have a teacher who has asked them to create an animation as a learning exercise. This class of users may be growing [21].

Professionals: 5 (7%). These are knowledge workers who are working on a variety of complex tasks. They may use an animation tool to explain a concept to a colleague, think through a problem, or prototype a more formal presentation.

When we look at users' goals in these scenarios, a similar picture emerges. *Prototyping* is the goal in 35% of scenarios, and nearly all of these scenarios come from artists. Amateurs wanted to *Entertain* others (21%) or just *Doodle* for themselves (21%). *Explaining* (21%) was

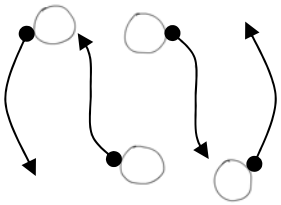


Figure 2: Sequence of contra dance moves (non-animator 4).

usually the goal of teachers and professionals. Finally, two scenarios had a goal of *Thinking* through some problem.

Informality can be useful in all of the above situations, and most categories of users can benefit from reduced learning time. Of the above categories, all but "Artists" have very limited time to learn about animation tools. This is why reducing learning time has been a major focus of our work.

Animation Operations

We now turn our attention to the more challenging problem of analyzing the content of each animation. Each animation tool provides a set of operations for specifying animation, and the complexity of a tool is determined by the number of operations and how the interface supports each one.

Following the tradition of informal tools [7, 11], we wanted our interface to match users' intuition as closely as possible. As part of our analysis of each animation, we noted how a user might intuitively express the events that took place. We found many different ways to express events, but over time they fell into a relatively small number of categories. These categories can serve as the primitive language elements, or operations, of an informal animation tool.

After a detailed analysis of the 72 usage scenarios in our library, we defined the following 18 animation operations:

Translate, Rotate, Scale: Common, simple operations.

Set Timing: Specify the speed and acceleration of a motion, rather than moving it at a constant speed.

Move Relative: Add a motion on top of another, so that the new motion is relative to the old motion's reference frame.

Appear, Disappear: An object appears or disappears. We count them separately because some tools support only one.

Trace: Animate a line over time, as if traced by a pen.

Repeat Motion: Repeat an event sequence.

Copy Motion: Move an object in the same way or a way similar to another object.

Define Cels: Create alternate appearances for an object, as in traditional 2D animation cels.

Morph: Turn one object into another over time.

Physically Simulate: Move objects as in "real life."

Interpolate: Define the start and end states of a change and animate the transition between the two.

Move Forward/Back: Change the stacking order of objects, so objects that were covered up are now uncovered.

Deform: Stretch an object out of its current state.

Move Limb: Define object skeleton & move a segment.

Orient to Path: Translate an object while pointing it in the direction it is moving.

In addition to these 18 animation operations, we defined five variants of *Translate*, *Rotate*, *Scale*, and *Set Timing*, but we do not discuss them here, because they added little to our analysis. We also defined eight other operations that are fairly orthogonal to the operations above (*Repeat Playback*, *Add Scene*, and *Play Sound*) or common to graphical editors (*Occlude*, *Zoom*, *Copy Object*, *Import*, and *Define Background*). These also added little to our analysis, and we assume that all would be present in a product.

This left us with 18 animation operations to choose from. The length of this list helps to explain why general-purpose animation tools are so complex, and it was a major design obstacle for us. We knew that we could not support all operations, but we lacked a method for choosing between them. This led us to develop a new analysis and optimization technique, described in the following section.

INTERFACE OPTIMIZATION

Our goal was to make K-Sketch fast enough to accomplish most tasks in minutes or seconds, simple enough for novices to learn after a short demonstration, but powerful enough to handle most of the scenarios in our library. We expressed this as an optimization problem, maximizing the number of animations we support (powerful), while minimizing the number of steps needed to complete a task (fast) and the number of animation operations available (simple). This led us first to enumerate all the possible ways of representing each animation in the library. We then built an optimization program that computes small, fast, and powerful operation sets and displays them to help a design team understand the tradeoffs.

For each animation in our library, we enumerated the "features" that a user would have to represent to accomplish their goal. For each feature, we listed one or more "approaches" to representing that feature and noted the operations required by each approach. We can then say that a set *O* of operations "supports" scenario *S* if all the features of *S* can be represented with one or more approaches for which all operations are contained in *O*.

Not all approaches are equivalent. Preference was given to those that gave the best results for the task and that could be performed the most quickly and easily. Subsequent approaches might produce animations that are less precise, but acceptable. If an approach required the user to perform more work than the preferred approach, we noted the number of extra steps. For example, if an animation had two objects that moved along curved paths, each of which could be approximated by 4 straight paths, then using straight paths would result in 8-2 = 6 extra steps.

Given this data, our goal was to compute the minimal sets of animation operations that would support every size subset of scenarios. We computed these results with a Python script that tested every combination of animation operations against combinations of sufficiently fast feature approaches. The definition of "sufficient" was configurable. We considered several possibilities, but decided that approaches with four or fewer extra steps were acceptable.

We ran the script with these options on a 2.8 GHz Intel Xeon CPU with 2 GB RAM, and after 18 minutes, it produced the results in Table 2. There are many solutions in

Scenarios Supported	100%	99%	97%	94%	93%	90%	86%	83%	79%	75%	68%	51%	46%	36%	28%	21%	15%
Num. of Operations	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Translate	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Scale	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	
Rotate	■	■	■	■	■	■	■	■	■	■	■	■	■	■		▒	
Set Timing	■	■	■	■	■	■	■	■	■	■	■	■	■		■		
Move Relative	■	■	■	■	■	■	■	■	■	■	■		■	■			
Appear	■	■	■	■	■	■	■	■	■	■	■	■					
Disappear	■	■	■	■	■	■	■	■	■	■	■	■					
Trace	■	■	■	■	■	■	■	■	▒								
Repeat Motion	■	■	■	■	■	■	▒	▒	▒	■							
Copy Motion	■	■	■	■	■	■	▒	▒	▒								
Define Cels	■	■	■	■	■	■	▒	▒	▒								
Morph	■	■	■	▒	■	■	▒	▒									
Physically Simulate	▒	▒	▒	▒	▒		▒									▒	
Interpolate	■	▒	▒	▒	▒												
Move Forward/Back	■	■	■	▒			▒										
Deform	▒	▒	▒	▒			▒	▒									
Move Limb	■	▒	▒	▒												▒	
Orient To Path	■	▒		▒													

Table 2: The data produced by our interface optimization technique (simplified). Each column shows the minimal sets of operations that would support the percentage of scenarios at the top. If an operation is needed in all minimal sets, a dark gray box appears in its cell. Light gray boxes show those operations that are in some, but not all minimal sets.

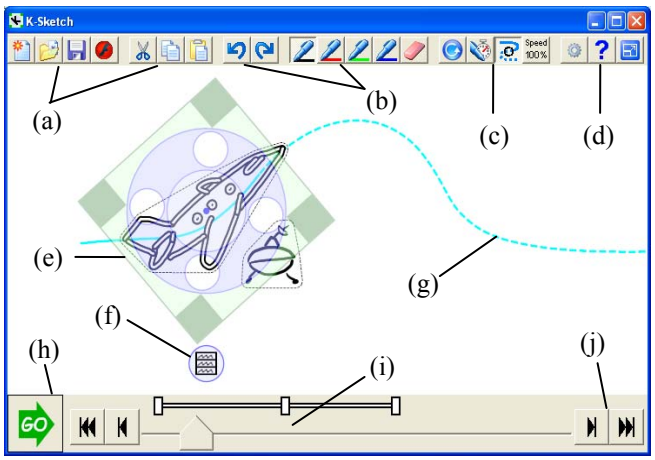


Figure 3: K-Sketch User Interface. (a) New, Open, Save, Export Flash, and Cut/Copy/Paste buttons, (b) Undo/Redo and Pen/Eraser buttons, (c) Repeat Playback, Record Drawings, Show Motion Paths, and Speed buttons, (d) Options, Help, and Full Screen buttons, (e) Object Manipulator, (f) Context Menu button, (g) Motion Path, (h) Go/Stop button, (i) Time slider bar, (j) Nudge Forward/Go to End buttons.

this table, but the visualization helps us to identify clear trends. The operations toward the top are clearly more important than those toward the bottom, which helps us to optimize the user interface for the more common cases.

We set our target for K-Sketch aggressively at 80% of scenarios. Because our method finds only operation sets that allow tasks to be completed quickly, we can "support" more scenarios than commercial tools such as Flash (56%) and PowerPoint (39%), which have slower precision-focused operations. We can reach 79% by providing the operations *Disappear* and above in the table plus two of the following: *Trace, Repeat Motion, Copy Motion, and Define Cels.* We chose *Trace* and *Copy Motion*, since they fit naturally into our design idea. We also included *Orient to Path* to bring us over 80%, because it also fit naturally.

This technique is general enough for a variety of domains. Our optimization program has exponential complexity and is impractical for large numbers of operations, but it parallelizes easily. The technique should not be used to compute one "optimal" solution, because designers' intuitions may reasonably conflict with computed results (as in our choice to support *Orient to Path*).

K-SKETCH: THE "KINETIC" SKETCH PAD

K-Sketch currently supports all ten desired animation operations: *Translate, Scale, Rotate, Set Timing, Move Relative, Appear, Disappear, Trace, Copy Motion, and Orient to Path.* Also recall that our analysis assumed the presence of eight other operations. Of these, we currently support only *Repeat Playback* and *Copy Object.* The others are straightforward to add and have little research value.

The K-Sketch user interface appears in Figure 3. The design is intended for pen-based computers and is visually divided into three parts: a tool bar on top, a drawing canvas in the center, and time controls on the bottom. We assume the presence of a physical "Alternate" button that users operate with their non-dominant hand to access alternate modes. We allow Tablet PC bezel buttons, handheld remote controls, and keyboard keys to be Alt. buttons. This form of mode switching has been shown to be effective [13].

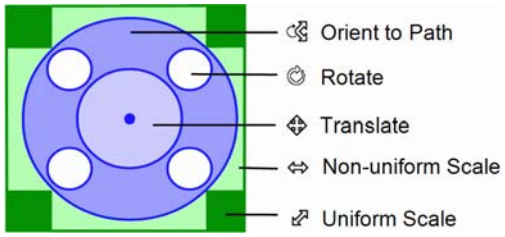


Figure 4: Object Manipulator control zones.

K-Sketch models animation as a sequence of editing steps over time. By default, any edit operation that the user performs happens instantaneously at the current time index and is visible from that time forward. Drawing and erasing are treated as any other edit, and this is how K-Sketch provides the *Appear* and *Disappear* animation operations. To record an animation, users perform these edit operations while time is advancing. When they do this, their edits are recorded in real time as they are performed. The following sections describe these editing and time control features and show how they support the remaining animation operations.

Selecting and Manipulating Objects

Objects are selected in K-Sketch by drawing a loop around them while holding the Alt. button. If 60% of a stroke lies inside the selection loop, it is selected and rendered in outline (as is the plane in Figure 3). Individual strokes can be selected by holding the Alt. button and tapping on them.

When an object is selected, a manipulator (Figure 3e) appears on top of it. This manipulator is designed for pen interaction, and it allows objects to be animated in a variety of ways, depending on where the user touches it (see Figure 4). A similar manipulator appears in Integrated Manipulation [9]. Tracking Menus [6] are also similar, but they follow the pen instead of hovering over selected objects. Whenever strokes are edited with this manipulator, they are implicitly grouped so that they can be easily selected in the future.

Using the manipulator inserts an instantaneous change into the animation at the current time. If the user holds the Alt. button when the pen touches the manipulator, time advances as long as the pen touches the screen, allowing edits to be recorded over time. This is how K-Sketch supports *Translate, Rotate, Scale,* and *Orient to Path*. Because the timing of these motions is taken directly from the user's pen movement, this supports *Set Timing* as well.

The manipulator initially appears centered, axis-aligned on the selected strokes. If users wish to move or rotate the manipulator relative to a selected object, they can use the object's context menu, which is accessed through a button

that appears just below the manipulator (Figure 3f). This allows users to rotate objects about a different center, scale them non-uniformly along different axes, or grab the manipulator in more convenient places.

Coordinating Motions

There is no difference between recording and playing in K-Sketch–edits to a playing animation are recorded, and this means that all objects move simultaneously when time advances. Users must rely on their intuitive sense of timing to coordinate the motion of objects, but K-Sketch provides three features to assist them: motion paths, a global speed control, and ghosts. Figure 5 shows some of these features in action as a user coordinates the collision of two particles.

Motion paths (Figure 3g) are pen traces that appear whenever the user records an edit with the object manipulator. These paths can help users coordinate movements by showing where objects will go in the future. The line is rendered solid for that portion of the motion that has already occurred and dashed for that portion that is yet to come. A motion path is visible whenever its object is visible. If the canvas becomes too cluttered, the user can turn motion paths off through a button in Figure 3c.

If objects are moving too fast for users to respond to them, they can slow down the animation through the Speed button (Figure 3c). This button shows a slider allowing users to speed up or slow down the global clock. The speed is initially 100%, and ranges from 2% to 50x.

Ghosts (Figure 5g) are transparent views of objects that appear at the moment in time when an object is erased to help users remember where it was. They are useful for coordinating the position of a drawing that replaces another, such as the explosion in Figure 5h. Ghosts can also appear under the pen when adding relative motion (see below).

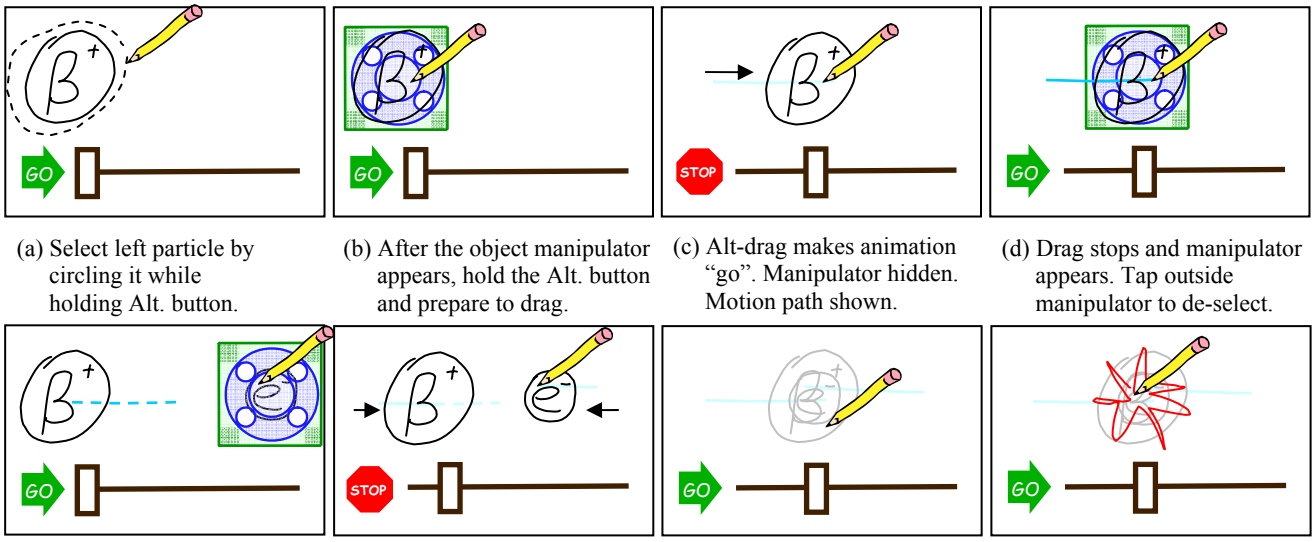
Overwriting and Adding Motions

Users can modify an existing motion path by going back in time and demonstrating the motion over again. By default, new motions overwrite existing motions. Any existing motion that was in progress is truncated, and any motions that started during the new motion are removed. Motions that end before or start after the new motion are unaffected.

Users may not always wish to overwrite motions, however. To support the *Move Relative* animation operation, it is necessary to add a motion on top of an existing motion. For example, a rolling wheel might be created by translating the wheel and then adding a rotation motion that occurs simultaneously in a new reference frame. It is easy for novices to imagine adding motions in this way, but we found that performing such an operation can be difficult if users must add motions in a particular order or explicitly specify the added motion's reference frame.

Our solution is to use heuristics to predict the reference frame that users intend to modify and to provide a correction interface when our prediction is incorrect. When only one reference frame exists, new motions overwrite existing motions. When multiple reference frames exist, the new motion overwrites existing motions of the same type (*e.g.* translates overwrite translates). If the user does not like the result of adding a motion, she can select "Fix Last Motion" in the context menu. This displays animated thumbnails of the resulting animations from all possible reference frame choices and allows the user to intuitively pick the correct one.

In practice, the correction interface is quite fast. For most animations, it is needed only once for each new reference frame. Because few animations in our studies required more than three reference frames, this list is usually short. Also, since most relative motions fall into a few types, we are often able to put the most likely alternative first in the list.



(a) Select left particle by circling it while holding Alt. button.

(b) After the object manipulator appears, hold the Alt. button and prepare to drag.

(c) Alt-drag makes animation "go". Manipulator hidden. Motion path shown.

(d) Drag stops and manipulator appears. Tap outside manipulator to de-select.

(e) Rewind, draw & select right particle, hold Alt. button, & prepare to drag.

(f) Positron moves as electron is dragged. Time collision by hand.

(g) Erasing hides particles. Objects that disappear are shown as ghosts.

(h) Draw an explosion where the particles disappeared.

Figure 5: Creating a particle collision animation from non-animator 5 with K-Sketch

Figure 6: Edit history feedback. (a) Main feedback (b) Local feedback for selected object.

Cut, Copy, and Paste

K-Sketch also provides the standard editing controls Cut, Copy, and Paste (Figure 3a), and these can be used to perform the *Copy Motion* animation operation. When an object is selected, users can click "Select Motions" in the context menu to select the motions applied to that object. When motions are selected, a *Copy* command will copy them to the clipboard. When another object is selected, a *Paste* command will apply those motions to the new object.

Recording Drawings

Users can perform the *Trace* animation operation in K-Sketch by pressing "Go" and drawing a line. The appearance of the line will be animated over time as would any other edit operation. However, users cannot hold the Alt. button to advance time as they do with the object manipulator, because this button is used for drawing selection loops. Instead, K-Sketch provides a "Record Drawings" mode (Figure 3c) that advances time whenever the pen touches the screen for a draw or erase operation.

Simplified Time Navigation Controls

Instead of a complex timeline that shows the history of every moving object separately, K-Sketch compresses time navigation into a single slider with an iconic overview of history (Figure 3i). Every edit event adds a tic mark at that moment in history (Figure 6a). When an object is selected, the slider highlights the edits related to that object (Figure 6b). Users can move through time by dragging the slider thumb or by tapping on either side of the thumb, which jumps to the next event. There are also buttons (Figure 3j) that jump to the beginning and end of the animation and that "nudge" time forward and back by $1/15^{th}$ of a second.

Users can tweak the start or end time of a motion by sliding the edit history tics along the timeline. By default, moving an event also moves any others that occur after it, but holding the Alt. button allows an event to be moved independently. The order of events is always preserved.

Simplified Recording Controls

Instead of using standard "VCR-like" recording controls with recording, playing, and stopped modes, we chose to limit K-sketch to two modes (going and stopped) controlled with a single button (Figure 3h). We did this to reduce the number of controls and the possibility of confusion between playing and recording modes. It is possible, therefore, for a user to play an animation and wait for an appropriate time before manipulating an object. As long as the animation is going, the edit will be recorded over time.

Implementation

K-Sketch is implemented in C#. The implementation makes heavy use of the Piccolo.NET graphical interface toolkit [3], which we modified to use the ink collection, rendering, and selection methods provided by the Microsoft.Ink API. K-Sketch totals 67 classes with 28,000 lines of code, plus 7 classes with 3,900 lines of code added to Piccolo.NET.

EVALUATIONS

We conducted three small user studies with K-Sketch as part of an iterative design process. These studies helped us to refine K-Sketch's recording controls and selection controls in numerous ways. They also helped us refine our support for the *Trace* animation operation and demonstrated the need for a global speed control. These studies also convinced us that novices could use K-Sketch to do real-world tasks after about 30 minutes of practice.

These user tests were helpful, but we needed a comparative evaluation with another tool to evaluate our claim that K-Sketch makes animation more accessible to novices. In our first attempt at such an evaluation, the first author produced 10 animations from our field studies with both K-Sketch and Flash. This user was a K-Sketch expert, but was also quite experienced with Flash. In this evaluation, most animations took 4–8 times longer to produce with Flash.

We believed these results to be promising, but we knew that the system needed to be evaluated with true novices. Also, we realized that Flash is a poor tool for comparison, because it is too complex for novices. For these reasons, we planned a larger laboratory experiment that compared K-Sketch to an animation tool for novices.

Laboratory Experiment

We decided to compare K-Sketch to Microsoft PowerPoint and focus on the advantages of informality. PowerPoint has powerful "Custom Animation" features targeted at allowing novices to make animated presentations. The tool provides most of the animation operations provided by K-Sketch; but is missing *Trace* and *Orient to Path*, and its support for *Set Timing* is limited. It is a good example of a formal, general-purpose animation tool for novices.

Method

Our study was a within-subjects comparison of PowerPoint and K-Sketch. Participants completed a practice task followed by two experimental tasks with one tool, and then repeated the process for another tool. Our independent variables were the tool used and the task performed, and both were counterbalanced across participants. The tasks were simplified versions of tasks from our field studies, a particle collision (Figure 5) and a dance maneuver (Figure 2). We chose these because both require multiple objects to be in motion simultaneously, the tools support the required operations, and pilot tests showed they could be completed in a 4 hour session. The practice task was designed to teach users everything needed to complete the experimental tasks.

Our primary dependent variable was the time to complete each task. Also, after each task we asked participants to fill

out two questionnaires asking how comfortable they would be showing their animation to others or creating it in front of others. We knew comfort was likely to be different depending on the audience. Therefore, for each situation (showing or creating), we asked the question for eight different audiences (no audience, 1 colleague in a meeting, 10 colleagues in a meeting, 1 student while tutoring, 30 students in a class, 300 students in a class, 30 professionals watching a presentation, 300 professionals watching a presentation). Responses were on a seven-point scale (1=extremely uncomfortable, 7=extremely comfortable).

There were also three variables dependent on tool only. After using each tool, participants took the NASA TLX cognitive load self assessment [8]. We also asked participants for subjective feedback on both tools at the end of the experiment. We asked how easy it was for them to work with both tools (7-point scale, 1=Very Easy, 7=Very Hard) and how fast they were at operating both tools (7-point scale, 1=Very Fast, 7=Very Slow).

Each task was an animation that participants needed to create. Participants viewed these animations in the QuickTime player, allowing them to replay and scan through the animation as much as they wished. These animations were formal so that participants could form a clear mental picture of the task as quickly as possible. To create a sense of time pressure, participants were instructed before each task to complete the task as quickly as possible. The instructions stressed that participants did not have to make the animations look perfect, and that it was more important for them to work quickly than it was to reproduce objects or their motions precisely. They were required, however, to keep the sequence of events the same.

Participants learned to use each tool during the practice task by working through an 8–10 page written tutorial that showed them how to use the tool to complete the practice task. They were allowed to keep this tutorial as a reference during the experimental tasks. During the experimental tasks, participants were asked to avoid asking for help unless they were stuck, and the experimenter intervened only when necessary to keep things moving.

Participants

We recruited 18 participants through a poster that called for people who are "interested in creating animation but have never done so." Of these, two were discarded from our analysis because they could not complete the tasks in the time available. Of the remaining 16, seven were men, and nine were women. Nearly half were students, and the others worked as artists, technology professionals, teachers, or dental assistants. Participants rated themselves "fair" in drawing skills using a 5-point rating scale (M = 3.22, SD = 0.88). On a 7-point scale, they rated themselves somewhat experienced with PowerPoint (M = 3.19, SD = 1.52) and very inexperienced with Tablet PCs (M = 1.44, SD = 0.89).

Participants also reported a sporadic desire to create animations, another obstacle to gaining expertise with complex tools. Most reported a desire to create animations once a year (n = 6), although some wished to make them once per week (n = 3). The purpose of these animations varied, including new works of art and animations for a company web site. Participants had little or no experience with PowerPoint Custom Animation (11 had none, 5 had 1–5 hours) or other animation tools (11 had none, 5 had 5–30 hours). On a 7-point scale (1=disagree strongly, 7=agree strongly), participants agreed somewhat that they were discouraged from creating animations. They were more discouraged by the time required (M = 5.23, SD =1.54) than by the complexity of animation tools (M = 4.43, SD = 1.83).

	K-Sketch		**PowerPoint**	
	M	*SD*	*M*	*SD*
1. Task Time (min.)	**6.76**	**4.30**	**19.57**	**12.24**
2. Practice Time (min.)	**26.36**	**4.80**	**43.52**	**11.36**
3. Fast? (1–7, 1 = fastest)	**2.75**	**1.39**	**4.50**	**1.10**
4. Easy? (1–7, 1 = easiest)	**2.13**	**1.36**	**4.25**	**0.78**
5. NASA-TLX (1–100)	**32.0**	**18.1**	**56.7**	**15.9**
6. Comfort Sharing (1–7)	5.17	1.25	4.89	1.30
7. Comfort Creating (1–7)	**5.46**	**1.14**	**4.10**	**1.35**

Table 3: Dependent variables with means & std. deviations. Results in bold are statistically significant.

Results

Test sessions took between 2½ and 4½ hours to complete. We limited our interaction with participants during experimental tasks, but it was sometimes necessary. 7 participants (44%) needed help 1–5 times to finish tasks with PowerPoint, and one needed help twice to finish a task with K-Sketch. Help was given in gradual stages, to limit interaction as much as possible.

Table 3 shows the means and standard deviations for all our dependent variables. Measures 2-5 were analyzed with pair wise t-tests, and the others were analyzed in a 2 (K-Sketch vs. PowerPoint) x 2 (Task A vs. Task B) within-subjects analysis of variance. To account for multiple tests of significance, we used a per-comparison alpha level of .007 for each t-test (we used a Bonferroni correction of .05/7 = approximately .007) to determine statistically significant effects. For ease of interpretation, p-values are reported.

Except for "Comfort Sharing", all the differences shown in the table are statistically significant. The time to complete experimental tasks was about three times lower with K-Sketch than with PowerPoint ($F_{1,15}$=24.28, p<.001). There was no other significant interaction on completion time. Participants needed about half as much time to complete the practice task with K-Sketch (t_{15}=-5.687, p<.001).

As the subjective measures in rows three and four of Table 3 show, participants thought K-Sketch felt faster (t_{15}=-4.869, p<.001) and easier (t_{15}=-4.667, p<.001) than PowerPoint. Row five shows that the NASA Task Load Index was about two times higher for PowerPoint than for K-Sketch (t_{14}=-5.443, p<.001).

To compare participants' comfort showing their animations to others and creating them in front of others, we first averaged each participant's responses across all eight possible audiences. The values shown at the bottom of Table 3 are the means and standard deviations of these averages. Oddly, participants were slightly *more* comfortable showing their K-Sketch animations to others, but this difference was not significant ($F_{1,15}$=.82, *n.s.*), and there was no other significant interaction. We then considered each possible audience separately, but still found no significant differences between tools on comfort showing animations. However, participants were significantly more comfortable creating animations in front of others with K-Sketch than they were with PowerPoint ($F_{1,15}$=14.88, p=.002). There was no other significant interaction on comfort creating animations.

When asked what they liked or disliked about K-Sketch, six participants said they liked using the pen, five commented that it was simple or easy to learn, and four said that it felt natural or intuitive. However, eleven participants commented that K-Sketch needed more tools for creating and editing precise graphics as in PowerPoint. When asked what they liked or disliked about PowerPoint, four participants said they liked PowerPoint's similarity to tools they were familiar with, and four said they liked the presence of precise graphical tools. On the other hand, seven disliked the fact that PowerPoint felt complicated or "technical", four said it was time-consuming or tedious, and three said it was inflexible or too structured.

Discussion

These results show that K-Sketch's simple interface has strong benefits. Experimental tasks took an average of one-third the time with K-Sketch. Participants' complaints about the complexity of PowerPoint indicate a major cause. Many participants were confused by PowerPoint's timeline and by its many menu options for timing control. By contrast, K-Sketch required less cognitive load, and participants felt that it was easier and faster. Participants found K-Sketch's simplified timeline to be more accessible than PowerPoint's, and we were pleased to see many participants manipulating events in K-Sketch's timeline, even though this was only briefly mentioned in the tutorial. The simplicity of K-Sketch's interface also meant less practice time was needed before tasks could be performed.

The benefits of informality are also evident in these results. The goal of informal interfaces is to allow deferring of details, but the results of this study show that informality can *help* participants to defer details when lack of time requires them to do so. The impulse to perfect in PowerPoint seemed involuntary. Participants were asked repeatedly to work fast and avoid making objects or motions perfect, but participants still spent time perfecting PowerPoint animations, which contributed to longer task time. The roughness of K-Sketch animations, on the other hand, probably contributed to the fact that K-Sketch felt easier and faster. It is also noteworthy that the extra time participants spent on PowerPoint tasks was not sufficient to make them more comfortable showing their animations to others. (Also note that this last result comes in spite of participants' mediocre self-rating of their drawing skills.)

Finally, we believe these results hint that spontaneous animation may become a practical medium in collaborative environments. Participants were more comfortable creating animations in front of others with K-Sketch, and their cognitive load was much lower.

RELATED WORK

We now look at work related to K-Sketch in the applications of animation, in informal interfaces, and in animation systems that use sketching and demonstration.

Applications of Animation

The most helpful animation research indicates when and how to use animation effectively. Rieber described conditions under which animation aids learning of concepts involving motion or trajectory [18]. Others explain why many animations fail to communicate effectively, and note that interactive playback control is important [23]. We have taken these results into account by designing K-Sketch to support the use cases that these researchers envision.

Informal Interfaces

In seeking a way to make animation more accessible to novices, we have taken inspiration from previous work in informal sketching tools [7, 11, 15]. The great insight of these systems is that much of the complexity of conventional design tools comes from their focus on precise details. When these details can be ignored or deferred, design tool interfaces can be much simpler.

Sketching and Demonstration in Animation

Sketching has often been used to simplify the animation process. Much of this prior work is geared toward producing highly polished final results. For example, sketched motion paths [17] have been used to direct the motion of 3D figures. Others have automatically generated animated movements of polished 2D character sketches [5]. K-Sketch is less concerned with polish and more concerned with accomplishing animation tasks quickly and easily.

There have been other efforts to apply sketching to rough animation. Alvarado used sketched annotations to generate imprecise animations of mechanical systems [1]. MathPad^2 generates animations from sketches of figures and mathematical equations [12]. These systems are valuable in certain domains, but K-Sketch attempts to be useful across many domains. KOKA [20] is a general-purpose animation tool that supports 17% of our scenarios. It defines a visual language for animation, which we avoid, because visual languages are difficult for novices to learn. Living Ink [19] is another general purpose animation tool which supports 51% of our scenarios. It uses static motion path sketches to generate motion and has a stack-based metaphor for combining motions that may be confusing for novices.

Our system is most similar to systems that use timing in addition to the spatial extent of strokes or gestures. Genesys [2] was the first system to use sketching for both creating objects and demonstrating motion. K-Sketch brings these ideas to novices and addresses open questions, such as which operations to support, how to select between operations, and how to navigate through time.

Most current work in animation demonstration simplifies the process of creating expressive motion for articulated figures [22] or deformable objects [10]. RaceSketch [14] is much more similar to K-Sketch than any other project listed here. It supports 25% of our scenarios with the operations *Appear, Translate, Orient to Path, Set Timing,* and *Deform*. It also provides a novel technique for timing refinement, but we believe a global speed control is sufficient and easier to apply in many cases. All of these projects contribute fast interaction techniques, but none share our goal of optimizing for both simplicity and power. Our analysis suggests that the costs of some techniques may outweigh the benefits. For example, Table 2 shows that the *Deform* operation is less important than others, and most rough tasks can do without expressive articulated figures.

CONCLUSIONS AND FUTURE WORK

We have presented K-Sketch, an informal animation sketching system for novice animators. Our efforts to reduce the complexity of animation tools while supporting a wide range of tasks caused us to carefully analyze the requirements of 72 usage scenarios. The resulting system relies on users' intuitive sense of space and time and allows tasks to be accomplished quickly and with little learning.

This paper makes the following contributions:

- The implementation of a novel tool, K-Sketch, that helps novices quickly create 2D animations.
- A novel optimization technique for designing an interface that is simultaneously fast, simple, and powerful.
- A laboratory experiment showing the benefits of K-Sketch over a more formal animation tool for novices, including a reduced task time by a factor of three.

In the near future, we plan to run another evaluation similar to the one reported here but with one-fifth as many participants and five times more experimental tasks to more fully demonstrate the generality of our tool. K-Sketch is available at k-sketch.org.

ACKNOWLEDGMENTS

This work has been supported by NSF Grants 0080562, 0205644, and 0742877. We also thank Elizabeth Sanders for help with statistical analysis, Cy Khormaee, Matt Davis, and Arpi Shaverdian for K-Sketch deployment and testing help, and John Canny for his continuing support.

REFERENCES

1. Alvarado, C. and Davis, R. Resolving Ambiguities to Create a Natural Sketch Based Interface. In *Proc. IJCAI '01*, 1365-71.
2. Baecker, R. Picture-Driven Animation. In *Proc. AFIPS Spring Joint Computer Conference*, 34 (1969), 273-288.
3. Bederson, B.B., *et al.* Toolkit Design for Interactive Structured Graphics. *IEEE Transactions on Software Engineering* 30,8 (2004), 535-546.
4. Davis, R.C. and Landay, J.A. Informal Animation Sketching: Requirements and Design. In *Proc. 2004 AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural* (2004), 42-48.
5. Di Fiore, F. and Van Reeth, F. A Multi-Level Sketching Tool for 'Pencil-and-Paper' Animation. In *Proc. 2002 AAAI Spring Symposium on Sketch Understanding* (2002), 32-36.
6. Fitzmaurice, G., *et al.* Tracking menus. In *Proc. UIST '03*, ACM Press (2003), 71-79.
7. Gross, M.D. and Do, E.Y. Ambiguous intentions: a paper-like interface for creative design. In *Proc. UIST '96*, 183-192.
8. Hart, S.G. and Staveland, L.E., Development of the NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research, in *Human Mental Workload*, P.A. Hancock and N. Meshkati, Eds. Elsevier, 1988. 139-177.
9. Honda, M., *et al.* Integrated Manipulation: Context-Aware Manipulation of 2D Diagrams. In *Proc. UIST '99*, 159-160.
10. Igarashi, T., *et al.* As-rigid-as-possible Shape Manipulation. In *Proc. SIGGRAPH '05*, ACM Press (2005), 1134-1141.
11. Landay, J.A. and Myers, B.A. Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer* 34,3 (2001), 56-64.
12. LaViola, J.J. and Zeleznik, R.C. MathPad²: A System for the Creation and Exploration of Mathematical Sketches. In *Proc. SIGGRAPH '04*, ACM Press (2004), 432-440.
13. Li, Y., *et al.* Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces. In *Proc. CHI '05*, ACM Press (2005), 461-470.
14. Moscovich, T., *Animation Sketching: An Approach to Accessible Animation*, Unpublished Master's Thesis, C. S. Department, Brown University, 2001.
15. Newman, M.W., *et al.* DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. *Human-Computer Interaction* 18,3 (2003), 259-324.
16. Park, O. and Hopkins, R. Instructional Conditions for Using Dynamic Visual Displays: A Review. *Instructional Science* 21 (1993), 427-448.
17. Popović, J., *et al.* Motion Sketching for Control of Rigid-body Simulations. *ACM Trans. Graph.* 22,4 (2003), 1034-1054.
18. Rieber, L.P., Computers, Graphics, and Learning. Brown and Benchmark, Madison, WI, USA, 1994. 144-170.
19. Rogers, B. Living Ink: Implementation of a Prototype Sketching Language for Real Time Authoring of Animated Line Drawings. In *Proc. Eurographics 2006 Workshop on Sketch-based Interfaces and Modeling* (2006), 115-122.
20. Takahashi, S., *et al.* A New Static Depiction and Input Technique for 2D Animation. In *Proc. 2005 IEEE Symposium on Visual Languages & Human-Centric Computing*, 296-98.
21. Tatar, D., *et al.* Handhelds Go to School: Lessons Learned. *IEEE Computer* 36,9 (2003), 30-37.
22. Thorne, M., *et al.* Motion Doodles: An Interface for Sketching Character Motion. In *Proc. SIGGRAPH '04*, 424-431.
23. Tversky, B., *et al.* Animation: Can it Facilitate? *International Journal of Human-Computer Studies* 57 (2002), 247-262.
24. Zongker, D.E. and Salesin, D.H. On Creating Animated Presentations. In *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 298-308.