EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

Living Ink: Implementation of a Prototype Sketching Language for Real Time Authoring of Animated Line Drawings

Bill Rogers

The University of Waikato, Hamilton, New Zealand

Abstract

Sketching with pen on paper is often used as a way of augmenting spoken descriptions: to help explain how some mechanical device works; to show the flow of information in an organisation, to show the action in a story, and so on. Sketches serve as a focus for the attention of viewers and help make abstract concepts more concrete. But images on paper don't move. Where the time and resources are available, as in preparing a lecture in advance, or producing a television program, authors recognize that moving images are often superior at conveying concepts and holding people's attention. An electronic display is capable of generating moving images in real time. We argue that such a display need not be used just to mimic pen and paper. After all paper is cheap, plentiful, has a wide viewing angle, and doesn't have batteries to run flat. Our 'Living Ink' system is a prototype implementation of a user interface for generating animated line drawings (animated sketches). Our goal was to provide a user interface which can be used in real-time, making interesting animations while an audience watches. This paper describes progress to date towards that objective and discusses proposals for further development.

I.3.4. [Computer Graphics: Graphics Utilities]: Graphics Editors; Picture Description Languages

1. Introduction

There are many circumstances in which people use informal sketches. When explaining how a mechanical gadget works, how we would like a user interface to function, or when telling stories, we often quickly draw pictures, to support our descriptions. This might happen in a teaching situation, where a teacher draws on a overhead projector screen or board. It might happen in a formal meeting or brainstorming session where participants draw on boards or pieces of paper. It might happen in a restaurant where ideas are jotted onto table napkins. The quality of the pictures does not have to be high, because the accompanying explanation compensates for their limitations. Indeed, in many situations the audience can ask for further clarification, and the sketch author can immediately make extensions to a picture, or invite their questioner to indicate or even draw themselves on the image to explain what aspects they do and do not understand. The crucial aspect is the immediacy. In a very short space of time it is possible to get onto paper some information that will significantly improve the level of communication being achieved between participants. Of course, given more time, sketches can be improved, and there is a continuum of circumstances between those in which a quick sketch is the most useful, and those in which a well planned and carefully constructed diagram is the most useful. In this paper we are interested in the use of rapid sketching for communicating ideas, or telling stories.

Sketch interfaces of one kind or another are available in many kinds of computer technology. PDA's and Smart Phones support pen input, and we can write sketch applications for them. Notebooks and desktop computers can have graphics tablets or 'write-on' screens for pencil and paper like sketching. It is also possible, albeit awkward, to draw with a mouse. The current technology most like the pen and paper though is the tablet computer, where software not only supports drawing strokes 'directly' on to a surface, but also simulates other aspects of real pens, like smooth anti-aliased lines, line weight that depends on pen pressure and the pen top eraser. This kind of technology is clearly well designed for use in the informal sketching domain. Our approach to animation sketching is designed for technology like a tablet computer, although our prototype implementation functions perfectly well with a mouse and a desktop PC.

Computer based sketching is significantly different to using pencil or pen and paper. The disadvantages of computer systems lie in limited resolution, imperfect pen devices, restricted drawing space, and in many cases restricted viewing angles. The advantages usually offered by the computer are the ability to store sketches, restore previous work, transmit information to other locations, and the capability of erasing, editing and transforming sketch data. In either case some kind of projection equipment can be used to make sketches visible to a large audience. Overall it is not clear that the advantages of computer

sketching outweigh the disadvantages in most circumstances. Paper is cheap, plentiful and doesn't have batteries that run flat. In order to make computer equipment more attractive in this role we need to offer compelling capabilities that paper cannot. The most obvious capability of the electronic display is that of rapid update. Sketching software already makes use of this by allowing lines to be erased or changed, which is a largely unsatisfactory process on paper. However we still have software that is oriented mostly to generating static finished sketches. In the kinds of scenario in which we see informal sketching being used, we argue that animated sketches would be of value, especially when telling stories or explaining how gadgets work.

Imagine telling a story to a child about the sea. On paper you might show a wavy line to represent the water surface, and draw a boat above it and a fish below. On a computer display such a line could really move. Figure 1 is a screen shot from our system of such a 'water surface' line. (For the purposes of presentation in print, the software has been modified to fade the lines from previous frames rather than erase them – giving an onion skin (ghost) effect. The real system just shows one moving line. In Figure 1, because of the small movement, the onion skin lines show as shaded grey patterns.)

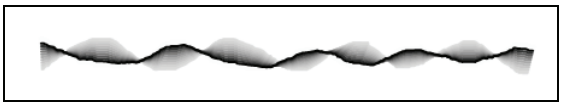

Figure 1: Animated wave in Living Ink

Of course this only works if it is possible to generate animations quickly – in real time – while observers are watching. If this could be achieved then electronic sketches would offer capabilities that really were not possible with paper, and we really might expect to see tablet computers replacing table napkins as the information exchange medium of choice in restaurants.

We take as inspiration for our work the sketch style animations produced by Big Time Pictures [BTP05], included in the British Television Channel 4 series 'Scrapheap Challenge'. In that series, contestants build mechanical devices. A programme host explains, with the aid of the animations, just how planned devices are expected to work or might fail. The animations are in a sketch style (see Figure 2) with enough detail to support the explanations. Some of the content shows devices operating. In Figure 2, the truck moves and the axe rotates. Some is background only – the clouds drift across the sky, and the truck driver has a worried expression. The animations provide a beautifully appropriate basis for explanation, showing moving parts, but including humorous elements that support a commentary that is entertaining as well as informative. Of course, these animations were not produced as the commentator spoke. They were probably the result of hours or days of effort. However, even if we cannot match them fully, we are interested in seeing how close we can get with real time authoring.

Figure 2: 'Scrapheap Challenge' animation still

The remainder of this paper is organized as follows. Section two reviews related work. Section three outlines our interface design, and shows examples of the interface in action. Section four gives some extended examples, showing how our primitive commands combine to produce complex effects. Section five reports some qualitative results from a limited informal user evaluation, and section six presents our conclusions and plans for further work.

2. Related Work

There is a compelling body of literature, eg: [LM01, PA03] that emphasizes the importance of using and retaining the imprecise sketch style in design documents. Any attempt to tidy documents can lead to authors being sidetracked into spending time on perfecting position and appearance at the expense of content. In the design context this spending of time can lead to premature commitment to early design ideas. In our authoring context sketching must be done quickly. We interpret the evidence in favour of design sketching as implying that adding precision controls to our interface would pressure authors towards time wasting adjustments of their work. Largely then, we need an interface that is as completely sketch based as possible.

The term 'sketch animation' is used in a number of ways in the literature. First there is the kind of animation shown in Figure 2. The overall appearance of the animation is of a sketched line drawing. The animation process itself seems to vary in these animations. Sometimes objects move with no internal change, suggesting that a single sketch is being moved and imaged in different positions. At other times it looks as though frames have been sketched independently, and irregularities between frames in the drawn strokes give the animation a slight 'squirminess'. So, even within this one set of examples we can distinguish sketch as a source of objects that are smoothly animated, and actual sketching constituting the animation itself.

The term ‘sketch animation’ has also been used to describe line animation in which the lines are found by searching for edges in the output of 3D scene rendering software. The goal there being to generate a sketch style of animation from scenes modeled fully in 3D. Examples are “Loose and Sketchy Animation” [CUR98] and the sketch animation feature of Lightwave 3D [NT05]. Although the result is sketch like, these systems are not built by sketch input.

A third use of the term is in progressive sketch animation [DIC05]. In these systems the strokes drawn by an artist are recorded and replayed. Usually the goal is a single, often quite detailed, sketch; for example, a portrait. The result is interesting to watch, and gives insight into the way the artist develops their work. It is usual to play back the drawing at a much higher rate than drawing speed. However this kind of animation is quite similar to the form we are trying to design. If the drawing is simple enough, it can be played at drawing speed. For example in describing a user interface it makes sense to describe each element as it is sketched – the animated drawing focusing audience attention on items, one at a time, as they are described. This is a common style of sketching used by teachers at a whiteboard.

Sketching has also been used in 3D animation systems, to specify the animation. In ‘Motion Doodles’ [THO03] sketch lines are used to quickly specify motion of articulated characters. The characters and their basic animations are modeled in a standard 3D graphics package manner: characters with skeletons and 3D meshes; basic animations elements, such as walk cycles, done using skeleton poses in keyframe sequences. Sketching is then used to define longer animation sequences. This is a gesture recognition system in which gestures are translated into motion sequences. A sketched loop, for example, specifies that the character perform a summersault; a sequence of arcs commands a sequence of steps. The system is interesting from a sketch viewpoint because it makes more use of the sketched information than just extracting gestures as abstract symbols. The position of the lines specifies location and direction of motion; the size of gestures dictates the scale of movements (large steps), and the timing of drawing controls the speed of motion. The system thus leverages the richness of detail in a sketch line to allow quick delivery of instructions that would otherwise require extensive parameter setting.

The same idea is also developed in “Animation Sketching”, Moskovich et al [MH04]. They address the problem of animating 2D sketched shapes using “motion by example”. In their system sketched items can be grabbed and moved. The track and timing of that motion is recorded. Elaborate motion, in which movements must be coordinated is handled by layering motion tracks. For example to have a bird’s two wings flap, the user first moves one wing, then rewinds and records the movement of the second wing. Where two objects must move in a coordinated way Moskovich’s system supports time warping. For example, to make a door open just before a character passes through, the user records the movements of door and character with approximate timing, then marks a coordination point on the time line of each movement, and lets the system warp the timing of each motion to make coordination points occur simultaneously.

Poliakov’s MorphInk [POL01] uses automatic morphing from shape to shape. In this system the primary motivation is achieving good compression of animation information for transmission over networks. However, its automatic morphing is a valuable approach to specifying animation. Computer animation easily provides tween frames for animation actions that are simple transformations. MorphInk extends the range of tweens that can be produced automatically, reducing authoring effort (and data size).

In [KST04] Kato et al, describe a system using sketched ‘Effect Lines’. In their system, line gestures, styled to be similar to the motion lines used in comic book cartoons, are interpreted as motion commands (Figure 3). A variety of gestures are used to specify rotation, translation and oscillatory motions. A particular advantage of this system in our context would be the appropriate and familiar style of the lines. They are commands that would communicate well to a real time observer, as well as to the animation system. In contrast to our planned system however, Kato’s system uses sketching only to specify animation. More conventional modeling techniques are used to generate the scenes being animated. This modal use of sketching avoids any difficulty that might arise in distinguishing motion commands from sketch content.

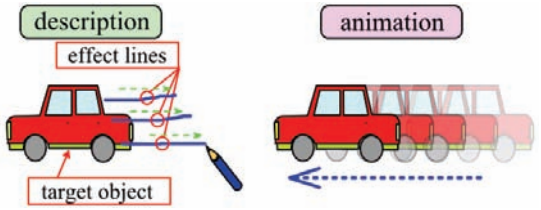

Figure 3: Effect lines from [KST04]

The K-Sketch project [DAV05] has as its goal, drastic reduction of the time taken to produce simple animations. Their project report includes a useful survey of the effects used in a collection of animations by a number of artists. Their prototype system animates sketched (line drawn) content. Animation is controlled partly by gesture and partly by pen operated controls. Gestures are used for selection and to specify motion tracks. Controls include a time slider and a multi-transformation widget (Figure 4). The transformation widget is popped up whenever content is selected. The user places their pen in the appropriate part of the widget for the kind of transformation they require (translate, rotate, etc), and drags to initiate and specify movement (Figure 4). The widget then disappears. The user sees the content move and also a low intensity motion line that grows with the motion and remains on screen to allow edit access to the motion later (Figure 5). As in Moskovich’s system, position and speed of motion are recorded for exact reproduction in replay. The system also

appears to run a clock continuously, so that successive animation actions each occur at the time at which they were entered (modulo explicit rewinding).

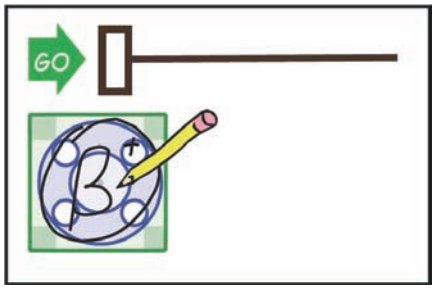

Figure 4: K-Sketch's multi-transformation widget, over sketch content (circled beta plus).

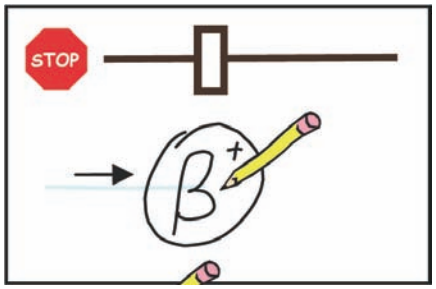

Figure 5: K-Sketch content moving with low intensity motion line showing track.

Perlin, eloquently puts the case for "The Animated Napkin Sketch" [PER02]. His goal is *allowing people to capture the immediacy and power of rapid creation of "drawings over time"*. His prototype system is called DrawPad. The concept is of a sequence of drawings or 'shots'. The timing and sequence of stroke drawing is retained so that a shot can be replayed as an animation. A simple gesture allows linking of 'sub-drawings'. The program can 'zoom' in and out of sub-drawings, thus providing a means of structuring the presentation/recording of ideas. An application of zooming occurs, for example, when the primary document is a mind map. Associated with each concept can be a (hierarchy of) sub-documents. The system is not really an animation system in the normal sense. Rather it is a way of retaining and structuring a sequence of sketch items, so that they can be presented or explored later. Its strength is the simple user interface which can be used to capture in real time a complex presentation of ideas.

In summary, a great deal of work has been done on using sketching interfaces to generate animations of different kinds. Considerable success has been achieved in reducing authoring time and ease of interaction, sometimes at the expense of reduction in precision. From these successes, we formed an optimistic view of the prospects for doing real time animation. In the following section, as part of the description of our prototype design, we comment on the relationship to the systems surveyed.

3. Living Ink Interface Design

In most graphic design software there is a strong element of selection and markup with tools. Usually there is a currently selected shape, highlighted in some way and decorated with tool handles: like corners to be used for scaling, or handles for curve adjustment. The usage process is to: add a primitive, complete with control scaffolding; and adjust until it looks right. Sketch interfaces can avoid much of this with stroke primitives. The artist draws lines and it can be assumed that they are correct immediately they are drawn, thus avoiding any adjustment process (except sometimes removing a stroke altogether and trying again). This allows us to avoid scaffolding associated with item construction. The design issues that remained are to do with specifying animation actions.

Our goal was to produce an animation program which could be used in front of an audience. This imposed strong constrains on the nature of the interface. Initially we took the view that we should avoid having any 'construction marks' or 'scaffolding' on the display. Ideally all that would ever appear would be the sketch, appropriately animated. As design proceeded we relaxed that constraint in favour of having the artist sketch 'track' lines for animation actions.

It would have been possible to achieve a 'no visible scaffolding' effect by using two displays – one of which was visible to the artist and one to the viewer. This would be viable in a teaching situation in which the artist could use a computer with local display attached to a projector, providing the viewer's image. Then it would be possible to decorate the artists view in any way desired. Of course, the dual display solution is not viable for using a shared tablet computer at a restaurant table, but more significantly, we felt that having the artist work on invisible controls would not support the 'draw and explain' task very well. It would be very easy for there to be periods of time during which the viewer would experience visual inactivity while the artist was busy with invisible adjustments and therefore probably not commenting either. For both reasons we decided that a common view for artist and viewer was necessary.

Specifying animation actions involves: selecting objects to be transformed, choosing the transformation, and specifying the track and timing that will be applied.

In K-Sketch, users select objects by drawing around them. Other systems allow the user to click on an object to select it, and then highlight the object to show that it has been successfully selected. Highlighting seemed undesirable, particularly if it persisted for an indefinite time (as is usually the case). Drawing around an object was less objectionable, as it would fit naturally with explanation – "and the fish (draw around it) swam to the …" After some initial experimentation we observed that objects were often manipulated immediately after being drawn, and adopted the idea of implicit selection. Actions apply to the last

object(s) created. Our system does have a selection operator (drag an outline with side button on pen held down), but it is rarely used. Explicit selection is acknowledged by briefly flashing the selected object red (for approx 1/3rd of a second). If several objects are selected in one operation, they are flashed in sequence.

Selecting a transformation, its track and timing were addressed together. Of other systems we have considered, the one that seems best suited to working in front of an audience is Kato's Effect Lines. At the time we were designing our system we were unaware of that work. In retrospect, though, using the idea would have been problematic for us in two ways. Firstly it would have been difficult to distinguish effect markup from object sketch lines. Secondly, whilst the idea of setting an object (like the car in Figure 3) moving by drawing effect lines is appealing, it would have put our artists under pressure to catch and stop the object in a timely manner. Instead we opted for a more conservative solution. We decided that movements could be specified with visible sketched strokes, and that there would be a toolbar holding buttons for each kind of animation action (operator buttons).

The basic interaction paradigm is stack based. The artist draws lines (strokes) which are put into a first-in last-out stack. Strokes can be either sketch data, or operator 'track' specifications. Each operator pops its operand(s) from the top of the stroke stack, and pushes its result(s) back. Most operators have two operands – data and track. For example, to draw an object and specify that it move over some track, the artist sketches the object, sketches the track and then clicks the 'path' (translation) operator. This is illustrated in Figure 6.

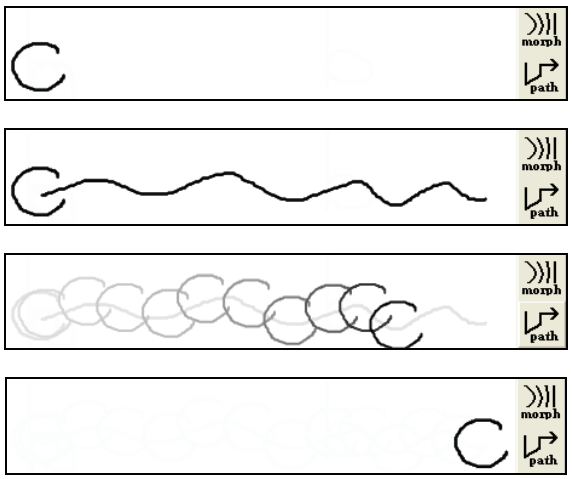

Figure 6: Using the path operator. (Part of a screen shot positioned to make the relevant part of the toolbar visible.)

The stack principle allows operations to be combined in a way that is not possible in a system in which tracks are specified directly by dragging. Almost any object can be used as a track. In particular it is possible to use an animated line as a track.

In terms of our design goals we think that this works reasonably well. The track lines have some contextual meaning. It is acceptable for the artist to say "and now the fish moves along this way (drawing the track)", then click the operator button and let it happen. In this way it is acceptable that track lines are visible to the audience. In fact, the system seems to result in more 'positive' animation. If the artist grabs and moves an object it tends to look as though it is floating about purposelessly. Showing where it is going and then letting it go there quickly and smoothly looks better. The toolbar doesn't occupy very much screen space, and the artist doesn't spend much time working with it.

That just leaves the problem of specifying timing. We were suspicious of the idea of recording detailed timing of pen movement – of animation by example – even though it was the main metaphor used in all the sketch controlled systems reviewed, other than Kato's Effect Lines. In our early experiments we found that it was difficult to adjust objects with any kind of timing accuracy. When people draw lines they are thinking more about shape than time, and speed of drawing tends to be dictated mostly by the complexity of the shape. It therefore seemed unnatural to reproduce drawing timing exactly. We also observed that having a running clock determine the timing at which actions began was 'oppressive' and that pauses which seemed appropriate while creating the animation were often annoying on playback.

Our timing is largely implicit. Initially, animation actions all take one second to perform. There is a system clock, but it doesn't move except when an action is entered, in which case it moves forward by one second. There are two timing buttons: 'Meanwhile' and 'Then'. 'Meanwhile' moves the system clock back in time to the start of the last action. 'Then' takes it forward. There is still, however, an element of timing retained from tracks. Actions do not follow tracks at a fixed speed (in the pixels per frame sense). When the pen is moved slowly, the system samples its position more often than when it is moving rapidly. Each animation action runs by using a uniform number of samples per frame, chosen to finish the given track in one second. This wasn't really a design decision. It was just what the first implementation was coded to do. Because it retains some of the 'sketchy' feel of imprecise pen movement we decided to leave it that way. As explained later, it was also helpful in matching strokes appropriately during morph operations. The one second time for actions can be adjusted after the animation is complete. Each action has start and end handles on a timeline. These can be moved to speed or slow the animation. A number of other controls are also provided in the timeline area, to permit adjustment of other aspects of actions.

The operators provided in the current version of the program design are: hide, show, bits, delete, morph, path, break, turn, size, group, one and split. They are invoked by buttons down the right hand edge of the screen, as shown in the full screen dump of Figure 14.

'Hide' and 'Show' make objects invisible and invisible. The transition occurs at the current time, but can be moved on the timeline if needed. 'Delete' completely removes an object from the system. 'Path' has been described already and illustrated in Figure 6.

'Turn' and 'Size' provide rotation and scaling respectively. Size is not implemented in the current prototype. 'Turn' works in a similar manner to 'Path', except that the track except that the track is used differently. Its starting point is the centre of rotation, and rotation 'follows' the tangent of the track (see Figure 7). A spiral track allows rotations of more than 360°.

Figure 7: Rotation with the 'Turn' operator

'Morph' takes two operands. It animates the transformation of one into the other. The normal mode of operation for Morph is to 'cycle' – the object transforms backward and forward. It can be set (radio buttons on the timeline) to 'one-shot' or 'repeat'. The 'repeat' option is like 'cycle' except that the shape changes from first to second form, then reverts instantly to first form and morphs again. The 'one-shot', 'cycle', 'repeat' options are actually available for other operators, but 'one-shot' is the default in those cases. Morph is illustrated in Figure 8.

In contrast to the MorphInk system we have not implemented any special algorithm for matching points in morphing. Our system can morph between composite (grouped) objects. In that case the component strokes of each composite object are matched pair-wise in sequence between the first and last morph positions, giving the artist some control of the process. With single objects, the way that pen motion happens usually leads to quite good morph point matching. Because pen movement usually slows in drawing cusps, there are many sample points on cusps and few on long straight line segments. For this reason morphing tends to map cusp to cusp in reasonably similar shapes. This happens in Figure 8. In addition the system forms a new cusp in the middle of the bottom line of the triangle to complete the square.

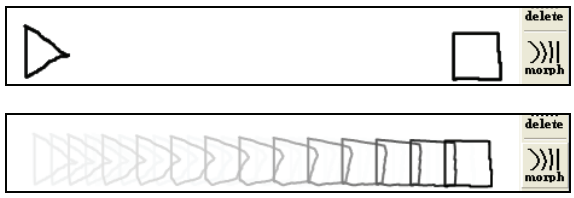

Figure 8: Morph operation.

'Group' combines two items to form a composite item. It can be applied repeatedly for form larger composites. The grouped object is flashed briefly to confirm the grouping. 'Split' separates a composite object into its components, flashing them in sequence to confirm. 'One' is a variant of 'Group'. It also makes two items into a composite. However, only one of the pair is displayed – ie: the composite is a set of items from which one is chosen for display every time the 'group' is instantiated. This is of value when the composite is used as a particle emitter.

The 'Break' operator allows an item to be cut in two. The track is used as a 'knife'. In Figure 9, an egg shape is 'cut' with a zig-zag line. In the third image of Figure 9 the two halves have been move apart by a small distance to better show the sections.

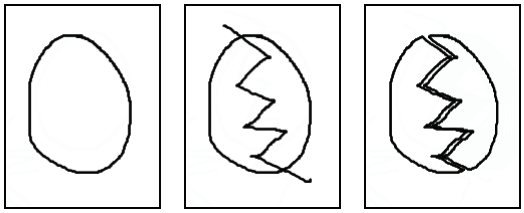

Figure 9: 'Break' cutting an egg

Finally the 'Bits' operator makes an item into a particle emitter. Controls are provided to set the set of directions and speeds, particle lifetime and emission frequency. Figure 10 starts with a sketch of a ship. The second image shows a single smoke puff above the funnel. The 'Bits' operator is applied to make it into a particle emitter, and in the last image a line of smoke puffs can be seen rising from the funnel.

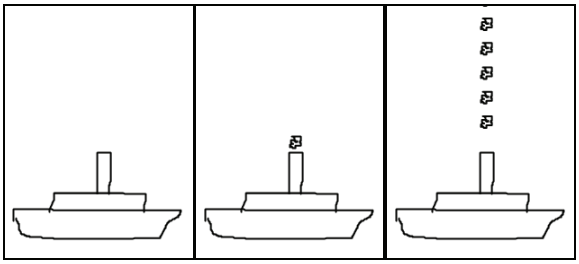

Figure 10: 'Bits' operator makes a particle emitter

4. Composite Operations

The stack style of command makes it possible to apply almost any kind of object as a control track to any other object (the main exception is that it is not possible to use a particle emitter as a track). In this section we will give three examples to provide an indication of the possibilities.

The first example is a small extension to Figure 10. The smoke emitter is grouped to the ship (which is itself a group of three lines), and a path is applied to move the ship across the screen to the right. The result is in Figure 11. The second example shows the 'One' operator in action. The letters 'a' to 'e' were drawn and grouped together

using the 'One' operator. The resulting group was made into a particle emitter. The direction of emission was set to all upward angles. The result is a screen full of letters, some of which is shown in Figure 12.

The final example is a very crude bee fluttering from flower to flower. The bee is just a circle with loops for wings. The wings are drawn in the top position, grouped to form a pair; then drawn again in the bottom position and grouped again. Clicking the 'morph' button makes the wings flap. An adjustment to the timeline speeds the flapping. Once the wings are grouped to the body, the whole bee can follow a 'Path' from flower to flower.

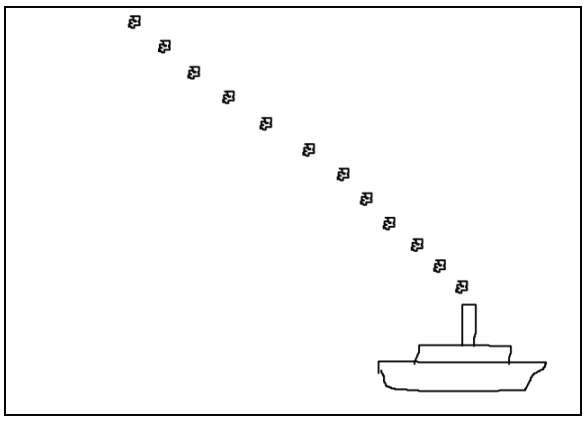

Figure 11. Ship moving with 'smoke' particle emitter

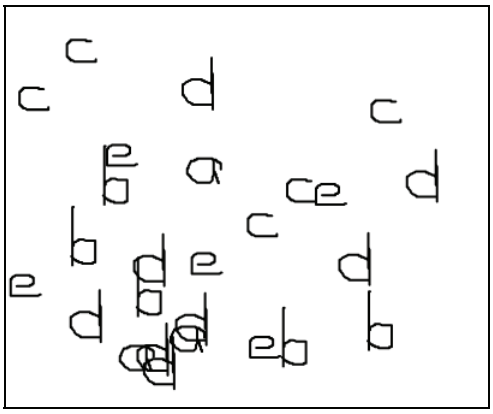

Figure 12. Particle emitter from 'One' grouped letters

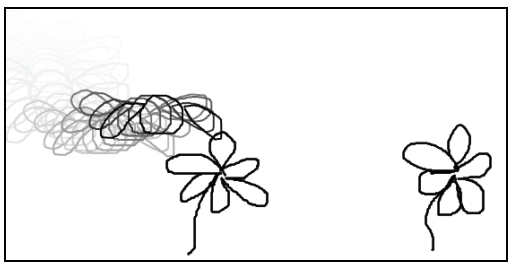

Figure 13. Bee flying

Clicking 'Bits' converts the flying bee into a swarm, in this case all following the same path (Figure 14.). If a number of different paths were needed, then those paths could be drawn, grouped with 'One' and applied with 'Path' to the bee. Each bee particle instantiated would itself instantiate a randomly chosen path. Note also that Figure 14 shows the whole application window with operator toolbar at the right, and timeline at the bottom including controls for adjusting particle emitter settings.

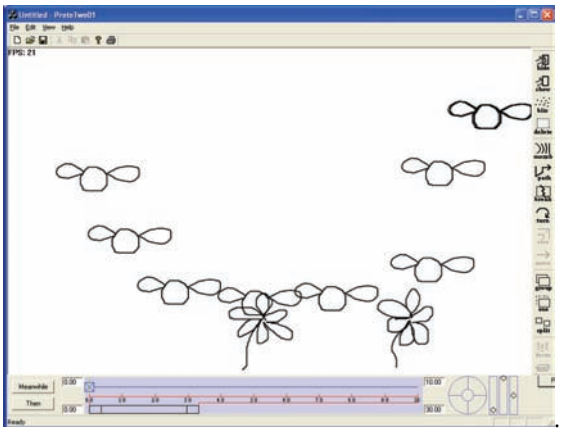

Figure 14: Line of bees following a path

5. Evaluation

We can report on the current state of the project in two ways. Firstly, the goal of producing a reasonably powerful animation system that can be used rapidly. Davis [DAV05] undertook a survey of features used in a number of animations. From their list of features Living Ink is currently capable of generating animations including: Translation, Appearing/Disappearing objects, Rotation, Repeating Sequences, Translation combined with rotation, Motion Hierarchies, and Morphing. It is not currently capable of Sequence reuse, Animation cells, Keyframing, Physical modeling or Sound. Scaling is included in the program design, but not yet implemented. The number of features included is the same as that included in K-Sketch, but the features sets are different.

Secondly, we undertook a small informal user evaluation. Five people took part: 3 children ranging in age from 11 to 16, and 2 adults. All were able to operate the program quite quickly. However, the kinds of results achieved varied considerably. Each person was allowed to experiment for as long as they wanted. All, especially the children, found the program engaging and spent an hour or more working with it. The bee visiting flowers was an invention of the 16 year old. Her experience was typical. She did not find the exact sequence of operations to make the animation obvious. She devised the idea, and set about experimenting to see how the effects could be achieved. It took 10 attempts to get a result she was satisfied with. Early attempts mostly failed because of operations applied in the wrong order, for example applying a path to the bee before grouping it to its wings (which led to the wings

flying off without the body). This experience partly indicates that the command set is complex, and that users have to work quite hard to translate their objectives into the command set provided. It can also be interpreted partly as a limitation of the current implementation. At present there is no 'undo' command, and once a control track has been bound to an object, there is no way to reverse the operation and change the track or the underlying object.

Finally, we were surprised by the number of lines that the participants put into drawings. In our own testing we had mostly used very simple drawings, usually no more than 10 lines. Participants put more items into each picture, and often used lines to shade areas. Because the rendering of digital ink is not very fast, animations with a large number (100's) of lines slow down significantly. Presumably this occurs because digital ink is rendered by the CPU rather than the graphics card. On a 2.16GHz Athlon processor with GEForce 2 graphics card, rendering 130 ink strokes (averaging approximately 100 pixels in length) with the Tablet PC ink library, occupied 100% of CPU time at 20 frames per second. In contrast, rendering the same 130 strokes as sequences of straight line segments ten times per frame at 30 frames per second involved only 50% of CPU time. We have done some preliminary experiments with alternative rendering methods.

6. Conclusions and Continuing Work

We have designed a notation/command set that allows production of a wide range of simple sketch animations. Operations include standard geometric transformations as well as breaking an object into pieces and using objects as exemplars for particle generation. Our goal was to support the production of animations in real time during teaching, design and story-telling sessions. Early user testing supports the assertion that our system allows very rapid authoring; however the evidence suggests that it is essentially an expert tool. It is simple enough for users to discover ways of achieving effects by themselves. It is fast enough for users to generate in real time effects that they have previously practiced, but it seems unlikely that people could create new effects in real time. This is not an unsatisfactory result, and is perhaps as much as we should have expected.

The current prototype has limitations that make it difficult to modify an animation. When trying to work in real time, this may not matter, because it would not be appropriate to tinker with an incorrect animation in front of an audience. However, it may be possible to improve the ease with which people learn to achieve effects with the system, by making it easier to correct faults. The next prototyping iteration will therefore include an undo facility. We are also working towards providing a way of 'looking inside' groups and operator bindings to change strokes without undoing the structures they belong to. The current proposal is a 'focus' command which would focus the editor on a single composite object, perhaps fading out other parts of the animation. Applying 'unfocus' would then return to the outside view.

References

[BTP05] BIG TIME PICTURES, http://www.bigtimepicturescom/examples/scrapheap.aspx

[CUR98] CURTIS, C.: Loose and Sketchy Animation http://www.otherthings.com/uw/loose/sketch.html

[DAV05] DAVIS, R: Informal Animation Sketching with K-Sketch, *Doctoral Consortium UIST 2005*, Seattle, WA, October, 2005

[DIC05] DECLERICO, D, Sketch Animation, http://www.chrisdiclerico.com/2005/11/12/sketch-animation

[KST04] KATO, Y., SHIBAVAMA, E., & TAKAHASHI, S. Effect lines for specifying animation effects. In *Proc IEEE Symposium on Visual Languages and Human-Centric Computing.* Rome, Sept 2004, pp. 27-34.

[LM01] LANDAY, J. A. and MYERS, B. A.: Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*. 34, 3 (2001), pp. 56-64.

[MH04] MOSCOVICH, T. & HUGHES, J.F.: Animation Sketching: An Approach to Accessible Animation. *Technical Report, CS-04-03*, Computer Science Department, Brown Univ. 2004

[NT05] VAUGHAN, W.: Sketch animation with Lightwave 3D, http://www.newtek.com/products/lightwave/tutorials/rendering/sketch/index.html

[PA03] PLIMMER, B.E., APPERLEY, M.: Software for Students to Sketch Interface Designs, Interact, Zurich, 2003.

[PER02] PERLIN, K: The Animated Napkin Sketch, http://www.mrl.nyu.edu/~perlin/draw/ (2002)

[POL01] POLIAKOV, V: MorphInk: Morphing Technology for Web and Wireless Animation, http://www.morphink.com/e/tools/WhitePaper1.pdf

[THO03] THORNE, C.; Motion Doodles: A Sketch Based Interface for Character Animation; *Masters Thesis*, University of British Columbia, 2003.