EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2010), pp. 1–4
M. Alexa and E. Do (Editors)

Transient Mode Interaction Techniques for Sketch-Based Interfaces

Anon[1]

[1]Mysterious University

Abstract

This paper describes an exploration of transient mode interaction methods suitable for use in sketch-based interfaces. Each technique is triggered by a sketched gesture that invokes a transient mode change, enabling users to seamlessly transition between drawing, forming selections, and issuing commands. These techniques are demonstrated in a simple cartoon sketching program called SKRUI Draw. Three idioms are described: flow selection, scribble-fill, and structural ink. While SKRUI Draw remains an experimental system, the sketch-based interaction techniques it provides show promise.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—

1. Introduction

Sketch-based applications are often used to support visual thinking and informal reasoning. It is important to support these activities without needlessly imposing structure. Typically, software tools present functions using persistent modes, such as pencil mode, fill mode, select mode, and so on. To perform an action, users temporarily shift focus to the editing controls to find and activate the appropriate widget before returning to work.

This paper presents several mode switching techniques that address the problem of persistent mode. The techniques presented here are *transient* because input reverts back to the default sketching mode when it is no longer needed. Each transient technique is triggered by interpreting sketched gestures.

This work presents three techniques: flow selection, scribble-fill, and structured ink. They are implemented in a simple cartoon drawing environment called SKRUI Draw. Because they are implemented together it is possible to evaluate them as an ensemble, from both interaction and engineering perspectives.

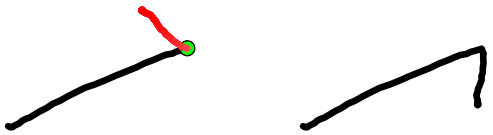
2. Related Work

The Inferred Mode Protocol [SL03] provides a framework for interacting with a drawing without persistent modes. The system recognizes pen input using context to mediate user intent. The system might quietly wait for more information, or it could ask the user to resolve the problem.

Some have developed GUI widgets that are appropriate for pen applications, making it easier to invoke modes and commands. CrossY demonstrates several pen-centric widgets based on *goal crossing* [AG04], based on the insight is that it is easier to move a pointer through a goal than it is to stop inside a target [AZ02].

Thierjung *et. al* studied methods for recognizing filled or hatched regions by analyzing stroke features [TBM09]. The current approach is designed to be interactive and is recognized before as the stroke is made. Thierjung's approach is more extensible but it is not interactive.

A portion of this system extends prior work on Flow Selection [JGD06]. It is a method for selecting portions of strokes, allowing users to quickly fix portions of a drawn stroke. It is issued with a dwell gesture, and a selection will begin to 'grow' outward from the nearest point on the line. Each point is partially selected. The longer the dwell con-



(a) Creating a hinge by selecting near the end of a stroke.

(b) The corrected stroke has been rotated and scaled.

Figure 1: *Flow-selecting near the end of a stroke enables users to use corners as a hinge, allowing the selected portion of a stroke to be rotated and scaled.*

(a) (b) (c)

Figure 2: *Multiple stroke flow-selection: If the user taps the pen immediately before beginning a flow selection, multiple strokes are selected. Each tap adds a constant distance to the area of effect. Each stroke is equally effected by modifications.*

tinues, the more strongly each point is selected. If the user drags the pen points are translated some amount depending on how strongly they are selected and how much the pen has moved.

3. Transient Interaction Techniques

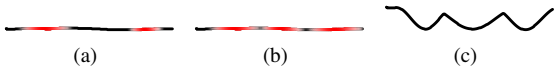
The following sections present three experimental transient interaction techniques, developed on the principle that *the user should never have to set down their pen*. These techniques aim to minimize the cognitive overhead associated with sketch-recognition based user interfaces. Transient modes should require minimal effort to enter and no effort to exit.

The methods are implemented in a simple cartoon sketching environment called SKRUI Draw, written in Java 1.6. These techniques were deliberately developed together to get a sense for how well (or not) they work as an ensemble.

SKRUI Draw contains a sketch recognition engine based on constraint satisfaction and is similar to LADDER [HD05]. This recognizer identifies time-dependent gestures as well as domain-specific symbols.

3.1. Flow Selection

The current system extends prior work on Flow Selection [JGD06] in three ways. First, if the user begins a flow selection near the end of a stroke, the selection will pause momentarily when it reaches a corner. When the corner point



(a) (b) (c)

Figure 3: *Selecting and editing multiple regions of the same stroke. Panels 3(a) and 3(b) show the user making a disjoint selection of the same stroke; in 3(c) the user manipulates the selection.*

is 70% selected and one neighbor is less than 35% selected, it acts as a hinge and is drawn as a large colored dot. If the user drags the stylus, selected points are scaled and rotated about the hinge. Figure 1 illustrates this process. Corners are found using the MergeCF algorithm [WPH09].

Second, users can select more than one stroke by tapping before starting the flow selection (See Figure 2). In this implementation, each tap expands the radius of influence by 30 pixels.

Last, selections persist for a short time after they are made, enabling users to select and operate on more than one region of the same stroke (see Figure 3). Flow selection can be thought of as applying *heat* to a stroke; heated regions are malleable but *cool* over time until they are no longer selected.

A point's selection strength is a product of two functions: *effort* and *strength*. Effort is calculated using a message passing scheme. Messages originate at the selection's *epicenter*—the point on the stroke that is closest to the stylus.

Effort represents how long the user must dwell for a point to become fully selected. In this implementation, effort is simply the accumulated curvilinear distance from the selection epicenter to the point in question, plus penalties imposed by intervening corners. The effort message passing function is:

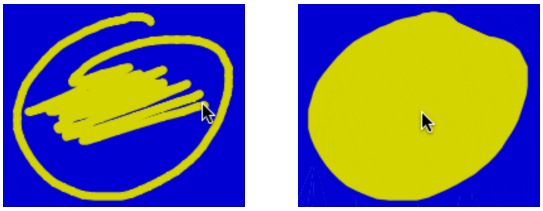
$$E[next] = E[i] + distance(i, next) + penalty(i) \quad (1)$$

Variables i and $next$ are neighboring points of the same stroke. The effort value for the epicenter point is defined as zero. The penalty function returns 130 for corners, and zero for other points.

The strength function depends on a point's effort value $E[i]$, the dwell duration t, and the existing selection strength $S[i]$ from any previous selections. An experimentally found constant, C, is used to control how quickly the selection spreads (in this implementation it is 0.1.) The following presents the strength function in two steps.

$$v[i] = \begin{cases} (cos(\frac{\pi \cdot E[i]}{t \cdot C}) + 1)/2 & : E[i] < t \cdot C \\ 0 & : otherwise \end{cases} \quad (2)$$

$$S[i] = max(v[i], S[i]) \quad (3)$$



(a) A scribble-like gesture causes the SKRUI Draw to enter a fill-mode.

(b) Moments later the region is filled using the same pen color.

Figure 4: *Scribble-fill: to color a region, the user issues a scribble gesture, characterized by lines with very sharp corners between them. Once a scribble is recognized, it fills in the convex hull of the pen path, and continues to add points to the hull as the user optionally drags the stylus.*

Note that the updated strength value can not decrease until the selection begins cooling, or until the user selects a different stroke.

3.2. Scribble Fill

People often make colored or hatched regions in a sketch. Using pencil and paper, one way of shading a region is to scribble back and forth, creating the visual suggestion of a shaded region. The back-and-forth motion (illustrated in Figure 4(a)) serves as a gesture that initiates a mode change to create a filled region.

Because the gesture must be detected as the user draws a stroke, we can not use the MergeCF corner finding algorithm used in other parts of SKRUI Draw. Instead, corners are resolved as the stroke progresses. For each point, two vectors are created that extend a constant distance outward along the pen path in both directions (in this implementation, the distance is 10 pixels in each direction). If the angle between these two vectors is less than 1 radian, it is a corner candidate. Typically, this creates clusters of candidates. The accepted corner is the one with the smallest angle in a candidate cluster.

A scribble-fill gesture is identified by a series of corners separated by straight segments. The ink between corners is considered straight if its ideal length (the line joining its endpoints) is no less than 90% of its actual curvilinear path length. To trigger a scribble-fill, the current implementation requires six consecutive corners and line segments that satisfy these constraints.

Once a scribble-fill gesture has been recognized, the pen path is used to form a convex hull, which is filled using the current pen color (shown in Figure 4(b)). Each additional pen point is added to the hull.

Lifting the pen ends the fill operation, and the user may continue sketching with the familiar pencil tool. However, if the next pen drag begins in the recently created filled region, it triggers another scribble-fill. This lets the user easily create filled regions that are not convex.

3.3. Structured Ink

Designers often make conceptual drawings with marks that guide production of important geometric properties. For example, when drawing a cylindrical object, the designer might begin by drawing a bounding parallelepiped to anchor production of subsequent marks. While this part of the drawing is not strictly part of the object under consideration, it is certainly part of the designer's thinking and production process. Such 'scaffolding lines' [CV09] appear in many design domains, particularly those concerning physical form.

SKRUI Draw supports users to create scaffolding via drawn gestures. This *structured ink* allows users to place points and lines and use them to guide production of new sketch input. They also serve as editing controls for working with nearby elements.

A structured point is created with a drawn gesture consisting of a dot followed by two short intersecting lines. The structured point is created at the intersection of the two lines. The ink associated with the gesture is removed. Structured point gestures are identified with a constraint-based recognizer conceptually similar to LADDER [HD05]. Additionally, the stroke components of a gesture must be made in a particular sequence, which helps reduce false positives.

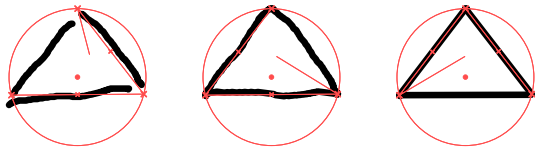
Structured points provide visual guides. In order to remove reduce clutter, only the three most recently created or activated points are used to show guides. After 30 seconds, structured points are no longer considered active. Tapping a structured point once reactivates it.

There are several types of guides. A straight line is drawn from the most recent point to the current pen location. Straight lines are drawn between the three most recently active points. Their midpoints are also shown. If there are three active points, they are used to fit a circle. Figure 5(c) shows all guide types.

Structured points may be used to control nearby sketched ink by tapping them. Two taps cause nearby lines move. If a line's endpoint is near the structured point, it is rotated and scaled to coincide with the structured point, using the opposing endpoint as a hinge. When the middle section of line passes near a structured point, the whole line is translated such to meet the structured point. Figure 5(b) illustrates double-tapping. Tapping three times will rectify nearby ink (see Figure 5(c)).

4. Future Work

The transient mode-switching techniques described in this paper are experimental prototypes. They will be improved



(a) Rough triangle with three nearby structured points. (b) Two taps: Rotate and scale lines. (c) Three taps: Straightening lines.

Figure 5: *Structured points can serve as editing nodes. Tapping twice causes nearby ink to scale and rotate to end at the structured point. Tapping three times rectifies those same lines. The dangling line in each figure is the guide line connecting the pen location to the most recently activated point.*

and new ones will be added. Progress will be guided in part by user testing, which will supply feedback about recognition errors caused by different user drawing styles and expectations, as well as preferences about which methods feel intuitive.

Further work must be done to simplify SKRUI Draw's software architecture to accommodate additional transient modes. Adding new modes increases user choice but substantially complicates the underlying system architecture. Adding new transient modes also increases user cognitive load: gestures must learn and remembered. Gestures should be 'different' enough that users are unlikely to trigger them by accident but simple enough that they can be remembered and issued without error. While the set of gestures involved in the current work are different enough that they are not easily confused, there are only three. Scriboli's *delimiters* provide a point of departure in this regard [HBRG05]. The delimiters are gestures that appear at the end of a selection stroke that people are unlikely to make otherwise, but are easily recognized and enable users to give commands on the selected region.

There are specific improvements to each technique discussed earlier. Flow selection can be used at the end of a selection sequence (as in Scriboli's Dwell gesture) to give users the ability to perform commands on the selected region. This could afford the ability to erase and to select colors or thickness properties from the selected area. The 'heating/cooling' metaphor might support overtrace strokes to correct flow-selected areas. Currently flow selection only operates on raw ink, but it should be extended to also work on scribble-filled regions as well.

Scribble-fill could use existing elements as guides (both structured lines and raw ink.) The filled region should then 'stick' to the existing boundaries so that changing the guide also changes the filled region.

While it is possible (and relatively easy) to create filled regions that are not convex, it requires multiple pen strokes. A significant improvement to scribble fill would be to generate arbitrarily shaped filled regions based on the pen path.

It is not clear if the current method for creating structured ink is the best way. Mechanical engineers sketching on paper draw guides alongside lines to indicate the subject's shape [CV09]. These lines are differentiated by their appearance: dashed or thin, light marks for auxiliary lines; thick, overtraced lines for object boundaries. Given that sketch-based tools often aim to provide a paper-like experience, it is worthwhile to explore how structured ink might be inferred based on the visual appearance of drawn lines.

5. Conclusion

This paper has presented three transient mode interaction techniques for sketch-based design tools. While they are still quite experimental they serve as useful objects to think with and might likely lead to a set of additional interaction idioms suitable for sketch interaction. The Java code for this project is open source and freely available on the Internet.

References

[AG04] APITZ G., GUIMBRETIÈRE F.: CrossY: a crossing-based drawing application. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2004), ACM, pp. 3–12.

[AZ02] ACCOT J., ZHAI S.: More than dotting the i's — foundations for crossing-based interfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2002), ACM, pp. 73–80.

[CV09] COMPANY P., VARLEY P.: Operating modes in actual versus virtual paper-and-pencil design scenarios. In *IUI 2009 Workshop on Sketch Recognition* (2009), Hammond T., (Ed.).

[HBRG05] HINCKLEY K., BAUDISCH P., RAMOS G., GUIMBRETIERE F.: Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2005), ACM, pp. 451–460.

[HD05] HAMMOND T., DAVIS R.: LADDER, a sketching language for user interface developers. *Elsevier, Computers and Graphics 29* (2005), 518–532.

[JGD06] JOHNSON G., GROSS M. D., DO E. Y.-L.: Flow selection: A time-based selection and operation technique for sketching tools. In *2006 Conference on Advanced Visual Interfaces* (Venice, Italy, 2006), pp. 83–86.

[SL03] SAUND E., LANK E.: Stylus input and editing without prior selection of mode. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2003), ACM, pp. 213–216.

[TBM09] THIERJUNG R., BRIELER F., MINAS M.: On-line recognition of hatched and filled regions in hand-drawings. In *Proceedings of IUI 2009 Workshop on Sketch Recognition* (2009).

[WPH09] WOLIN A., PAULSON B., HAMMOND T.: Sort, merge, repeat: An algorithm for effectively finding corners in hand-sketched strokes. In *Proceedings of Eurographics 6th Annual Workshop on Sketch-Based Interfaces and Modeling* (2009).