Interaction Concepts for Fluid Freehand Sketching

Menno Nijboer* Moritz Gerl† Tobias Isenberg†

University of Groningen

Keywords: Interaction, Sketching

Abstract

In this paper, we explore a minimalistic, gesture-based interface for fluid freehand concept sketching with vector graphics. Our approach leverages the advantages of both the GUI and gestural interface paradigms. We describe how to use frame gestures to control rotation, translation, and scale of the drawing canvas and of stroke selections. Based on an implementation of this concept we evaluate our tool with both novices and experts, and report on both its benefits and drawbacks.

1 Introduction

The control of drawing and sketching systems with sketch-based interaction techniques seems natural to users and has recently received considerable attention (e. g., [BBS08, BBS09]). With the increasing complexity of drawing systems, however, the set of required command gestures for purely gestural interfaces increases as well. This requires users to learn and remember an increasing number of commands, which might compromise the usability of such sketching systems, in particular for novice users. The fluent and direct interactions that are possible with gestures, however, also bear great potential for rapid editing of drawings, and particularly of sketches.

In this paper we explore a minimalistic, gesture-based interface for fluid freehand sketching with vector graphics. In such an interface, the fluid switching between three types of basic interactions are needed: (1) normal drawing, (2) interaction with the drawn strokes, and (3) interaction with the canvas itself. We explore new interactions that enable a fluid switching between these types, based on mapping canonical transformations (translation, rotation, scaling) of the whole canvas or of stroke selections to contextual gestures that are started from the canvas border or the selection frame. Combined with elements from existing drawing systems, this frame-based interface lets us investigate contextual gestural control for the placement and orientation of the canvas. The analogous interaction with stroke selections facilitates fluent and direct rigid transformations of strokes without having to switch between dedicated operating modes.

We evaluated the proposed concepts in an informal user study and learned that the concepts were generally well-received and that our interface requires only minimal instruction for a user to become familiar with it. Artists proficient in digital drawing particularly liked the notion of a directly-manipulatable canvas, and novices were especially attracted by the ease of learning and use. We also report on some shortcomings of the proposed interface in form of restrictions by the *actual* interface border and the use of a hold gesture to make selections or to erase groups of strokes. In summary, this paper contributes novel concepts for interacting with freehand sketches, including a new way of mapping canonical transformations of both canvas and stroke selections to their frame, and an evaluation of the proposed techniques with novices and experts in digital drawing.

This document is structured as follows: In Section 2 we discuss the related work. Section 3 presents the interaction concepts of our sketching system. In Section 4 we outline the model we use for representing brush strokes. In Section 5 we report on the informal user study and discuss its results. We draw conclusions in Section 6.

2 Related Work

The interaction design of our sketching system relates to work in pen-based interaction, sketch-based interfaces, digital drawing, and interactive stroke-based NPR. Specifically, our work relates to sketch-based interfaces for concept sketching, which has been studied in detail for 3D content creation. For example, Zeleznik et al.'s SKETCH [ZHH96] used a purely gestural interface for sketching 3D scenes. In the subsequent UniCam system [ZF99], the camera can be

*e-mail: m.nijboer@student.rug.nl

†e-mail: {gerl|isenberg}@cs.rug.nl

rotated in 3D via 2D gestures starting from the border region of the viewing window. We draw upon this idea and use location-sensitive gestures on the interface border for manipulating both drawing canvas and strokes. Unlike UniCam's use of gestural 3D camera transformations in the interface's center region, we integrate all 2D canvas manipulations into the border region, leaving the center region for inking interaction. This decision relates to the mode switching problem between ink and gesture modes in pen interfaces which was analyzed experimentally by Li et al. [LHGL05]. They report that pressing a button with the non-preferred hand offers the fastest performance. We satisfy this finding by providing hotkeys in our interface, but also offer pure pen interactions for all system functionality. For the BezelSwipe [RT09] interaction, Roth et al. as well make use of gestures on the interface border to prevent mode errors in the interaction with mobile touch screen devices. In ILoveSketch [BBS08], in contrast, a non-preferred hand button is employed for switching between inking and gesturing modes. Bae et al. integrate recent advancements in sketch-based interaction and modeling in this 3D sketching program, here a purely gestural interface provides access to all sketching interactions as well as camera and drawing surface manipulations. With EverybodyLovesSketch [BBS09], they adapt the system to the needs of a broad audience. While our sketching system conceptually overlaps with Bae et al.'s work, our concept is targeted at 2D sketching, investigates the use of contextual gestures for switching modes explicitly when interacting with the 2D canvas in order to avoid having to learn a fairly large gesture vocabulary. In this respect, our design is more similar to the sketch-based implicit surface modeling tools Teddy [IMT99] and ShapeShop [SWSJ05] which also rely on both gestures and a GUI and use a toolbox for explicit mode changes.

Apart from sketching and modeling 3D content, pen-based interfaces were investigated for editing text and graphics documents. The gestural interface of Hinckley et al.'s InkSeine [HZS*07] supports active note taking tasks and in-situ search queries on tablet PCs. Although InkSeine provides visual feedback in the form of labeled gesture previews, our approach differs from the purely gestural interface of InkSeine. At the same time, it differs from menu-based gesture-enabled GUIs for pen input as found in, e. g., ScanScribe [SFLM04], a sketch-based graphics and text editing program, or Zeleznik et al.'s Fluid Inking [ZM06], an approach that augments free-form inking with gestures. In this respect, our approach lies more along the lines of the sketch-based animation tool K-Sketch [DCL08]. In this system, Davis et al. present a gesture-enabled widget for manipulating objects. The stroke manipulation in our program follows a similar concept. In contrast to a purely gestural interface, the gesture-augmented GUI approach allows us to reduce the gesture vocabulary. We make use of a small, consistent set of command gestures, so we can avoid advanced techniques for handling complex gestures such as gesture delimiters [HBRG05]. Hinckley et al. also present ideas on multiple-stroke selection [HGA*06]. Similar to these approaches we also use crossing interactions for selecting strokes, but chose an explicit mode for multi-stroke selection. Our interface design also contrasts the use of gesture-invoked implicit mode changes found in most of the described purely gestural interfaces. Instead, we make use of two basic editing modes in the form of an inking and a stroke shaping tool, which are selected via a button menu. Similar to [HGA*06], our button menu is optionally local or non-local. The crossing interactions that we provide for multi-stroke selection and erasure borrow from the drawing application CrossY [AG04].

Our work also relates to research in interactive stroke-based NPR. For instance, we employ a stroke model that builds upon Hsu and Lee's skeletal strokes [HL94] who also investigate a pen tablet as input device for their pen and ink drawing system. This idea was further investigated by Kalnins et al. [KMM*02] who presented a system for drawing strokes onto 3D models. In digital painting, Baxter et al. [BWL04] as well as Vandoren et al. [VLC*08] presented interfaces that physically emulate the painting process. Related to this work, we derive interaction metaphors from physical actions in the process of concept sketching with pen and paper and provide them within a minimalistic user interface. We employ an adjustable canvas that has just recently found its way into drawing packages, although the benefit of supporting artwork orientation in digital drawing has been investigated before [FBKB99]. In this context our system relates to the many digital drawing and painting systems that followed the seminal Paint [Smi78]. Specifically, our work directly relates to dedicated sketching and painting systems such as Painter, SketchBook Pro, or ArtRage. Similar to these programs, we make use of a minimalistic user interface and interactions that emulate physical drawing actions. In contrast to the use of menu-based GUI and hotkeys to support canvas transformations in these programs, however, we examine a means of direct canvas manipulation via gesture-sensitive interface borders.

3 Interaction Concepts

A striking difference between a traditional sketching setup and a digital one is the way the artist can interact with the actual canvas. In traditional sketching, the sketchbook can be held or placed freely by the artist and rotated to his or her liking. In digital drawing, the canvas was traditionally aligned with the computer screen. More recently, developers started

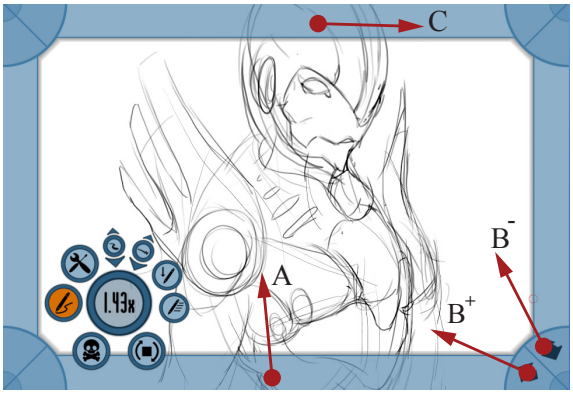


Figure 1: Canvas manipulation. The arrows depict the click and drag motions for the different frame gestures, where A is translation, B is scaling and C is rotation.

to equip drawing software with the possibility to adjust the drawing canvas to a preferred orientation, location, and size. These canvas interactions commonly are provided to the user as menu entries which can alternatively be performed with hotkeys. We examine purely pen-based interaction metaphors for canvas and stroke manipulations: the use of the screen border as an active interface element that allows manipulations of the canvas and the interaction with strokes or groups of strokes in an equivalent way.

To enable pen-based interaction with the canvas we use the interface's border as an active element enabled with contextual gestures (Fig. 1). Actions performed on this border are mapped to manipulations of the canvas. Using the interface border for canvas transformations provides direct access to these interactions from the entire central interface area as well as allows us to reserve the central area for inking and stroke interactions. Of the three 2D canvas transformations (canvas translation, canvas rotation, and canvas scaling), the first two are inspired by the affordances of non-digital sketchbooks, whilst scaling of the canvas offers the benefit of a digital sketchbook for working at arbitrary magnification levels.

In analogy to touching the border of a piece of paper with a single finger and moving the finger either along or perpendicular to the border, we use the following mappings. A *frame gesture* invoked by touching the border and dragging roughly parallel to the border (C in Fig. 1) is mapped to a *rotation* of the canvas, enabling artists to easily rotate the canvas to their liking. This gesture emulates the feeling of dragging parallel on the border of a sheet of paper to rotate it. Alternatively, a frame gesture that starts on the border but drags the pen perpendicular to it (A in Fig. 1) results in a *translation* of the canvas. Both behaviors resemble the rotate-and-translate (RNT) interaction for working with mobile objects in direct-touch settings [KCST05]. Both interactions can be performed from

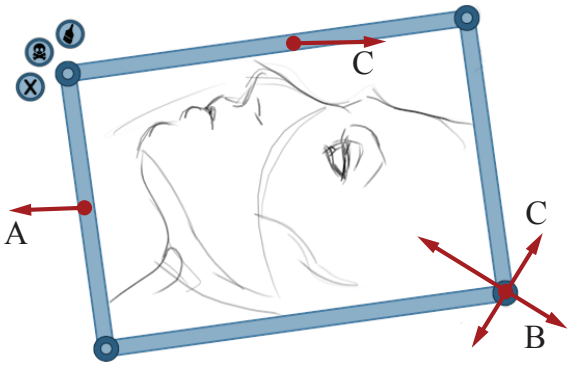
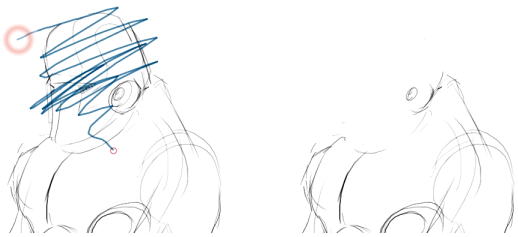


Figure 2: Stroke selection frame and transformation gestures (arrows): A—translation, B—scaling, C—rotation.

all interface borders. This results in low cursor travel distances and permits a fluent access of these functions throughout the entire interface.

Another option to distinguish rotation and translation commands would be to split the interface border in two separate regions for rotation and translation, but this would require a higher pointing precision. We designed our interface with touch displays in mind and thus opted for large interface elements, allowing for imprecise and rapid grab and drag interactions. Additionally, with our discrimination of rotation and translation by gesture direction we simulate the haptics of rotating and moving a physical canvas ontop of a drawing surface. Apart from this, we did choose not to use the frame corners for canvas rotations, as these would most naturally map to a gesture direction tangential to the corner. The lack of display space to perform such a gesture in the interface corners was a reason for us to use the center of the borders for canvas rotations. Furthermore, we implemented rotations of the canvas as a rotation around the center of the interface. This is motivated by the assumption that most commonly, users would work with an approximately centered canvas. Rotating around the interface center, however, is subject to several limitations. It could be improved upon by including more advanced rotation methods which allow to rotate around specific points, as recently proposed by Yu et al. [YSI*10].

Finally, we employ the corners to enable canvas scaling, an interaction that is not possible with the real-world counterpart. Because we cannot move further outward from the corner of the interface when it is enlarged to fill the screen, we offer two regions for each corner (B^+ and B^- in Fig. 1), one for zooming in and one for zooming out. These gestures solely depend on the starting region and not on the direction of movement, what facilitates using the interface corners for canvas scaling. All frame gestures, once they are started and recognized, are no longer restricted to the frame and users can freely move the cursor across the interface.

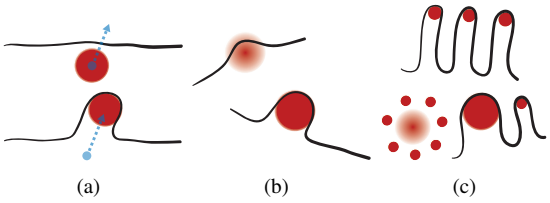


(a) (b)

Figure 3: Erase-line interaction: (a) the (fading) circular marker that started an erase-line drawn from it; (b) shows the result.

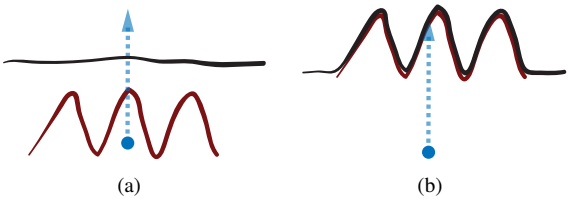
Rigid transformations of strokes or stroke selections (selections are created using a hold-and-scratch gesture which is described below) are provided to users of our system analogously to the canvas transformations, using a gesture-enabled selection frame (Fig. 2): *translations* are performed by dragging perpendicular from the frame (A), while *rotations* are possible by dragging parallel along the frame (C). *Rotations* can also be performed by dragging tangentially from the circular corners. Because there is usually enough display space around stroke selections to drag tangentially from the corners of the selection frame, we make use of this gesture as an additional means of rotating groups of strokes. *Scaling* is done by dragging a corner towards or away from the center of the selection (B; in contrast to the canvas case, here we do not need to differentiate between scaling directions because users can freely move away from selection corners). This fluid grab and drag concept allows us to make all these transformations of strokes directly accessible without explicit mode changes.

Erasing is performed with the eraser tip of the pen. 'Drawing' with the eraser tip partially erases strokes in the brush region, as commonly expected from a drawing application. Groups of strokes can be erased with a 'scratch out' gesture. To invoke this mechanism without an explicit mode change, we make use of a dedicated *hold gesture* (touching and holding for 0.5 s to prevent interference with quick partial erasing): clicking on the blank canvas and holding for half a second creates a red circular marker, drawing a line out from it invokes the multi-erase line (Fig. 3). We employ the same holding and scratching mechanism to create stroke *selections* (Fig. 2), using the pen's drawing tip. Singular strokes can also be selected by clicking and holding on them. A selection of strokes can be de-selected, erased, or defined as a brush tip for stroke deformations via clicking dedicated buttons (the three small buttons to the top left of the selection frame in Fig. 2). The crossing-based multi-stroke erasure and selection interactions not only permit fast and loose erasing or selecting with zig-zag gestures but also precise control by drawing selectively over particular strokes of interest.



(a) (b) (c)

Figure 4: Local reshaping using a radial brush (a). This reshaping tool is customizable via (b) a softness parameter , and (c) the usage of multiple radials as brush tip.



(a) (b)

Figure 5: Local reshaping using a (a) red custom stroke as a brush to shape an (b) existing black regular stroke.

Local reshaping of strokes can be accomplished with local displacement tools, using either a circle or a custom stroke as a brush. These tools directly alter a stroke's geometry. When applied to a stroke, this stroke is locally displaced and extended along the brush geometry and dragging direction. We build upon equivalent tools in existing vector drawing applications, such as the *warp* tool in Illustrator. We extend the existing methods, e. g., with a softness parameter for the radial brush (Fig. 4) and the generalization to custom strokes (Fig. 5).

4 Stroke Representation

To permit the described interactions, we employ a vector-based stroke representation. The vector-based strokes are pre-triangulated to exploit graphics hardware for rendering.

A stroke is stored in three components: its set of control points, the segments in between, and its vertex geometry. The control points are taken as basis of a uniform cubic B-spline. Along with the control point coordinates, we also store the stroke width and opacity. The width parameter can be seen as the radius of the disc of which the control point coordinate is the center. This information is used to calculate ribs of the stroke, from which the triangle geometry is generated. We denote a line segment perpendicular to the center line of the stroke as a rib, with length equating to the stroke width. Encoding a stroke shape in this manner was inspired by the work by [SWT*05], who used a similar approach to draw digital calligraphy. We extended this stroke model with a LOD mechanism, in order to achieve scalability. The chosen LOD mechanism was inspired by [ESAV99], who employ merge trees to expand and collapse vertices of triangle

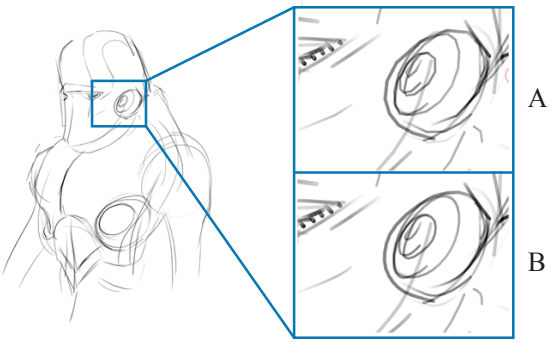


Figure 6: Stroke refinement. The insets show a magnified part of the sketch without (A) and with (B) the application of stroke refinement.

strips. We adopted this technique to our stroke model and store the ribs of a segment in a merge tree. This permits to expand and collapse ribs of strokes depending on the magnification level. This enables a view-dependent resolution of the triangulation. Fig. 6 illustrates the impact of stroke refinement achieved with this LOD support.

In addition, we down-sample the strokes' control points from a high resolution input. Only sample points with sufficient visual impact are used as control points for the strokes, using a technique similar to one used by [ABCO*08] for point set surfaces. This down-sampling allows us to both efficiently store and render strokes. Some interactions in our sketching system also require up-sampling of strokes which we accomplish using the forward differencing method by [Bru98]. As a further optimization, strokes are stored in a quadtree. This is done to speed up range queries, which are necessary for many operations in the system. In order to support range queries along the entire extent of the strokes, we store a low-resolution version of the strokes' segments in the quadtree, instead of their control points.

The described stroke model yields an advantage over a plain vector representation: the pre-triangulation enables fast visual feedback in full quality when canvas or strokes are transformed. The strokes' triangle geometry can be directly rendered. In contrast, a plain (not triangulated) vector representation of strokes would result in a lower rendering performance. Here, strokes have to be rasterized each frame, a process that consumes a considerable amount of processing resources.

5 Evaluation

The merit of our interaction concepts, of course, can only be judged by actual users. We received feedback on our techniques in an informal study which showed that the proposed interaction techniques were generally well-received. We discuss the benefit of our techniques based on insights we gained with this study.

Figure 7: An artist working with the system.

Our informal user study was conducted with two distinct groups of participants: a group that consisted of five (all male) artists who are proficient in digital drawing ('experts') and a group of five (one female, four male) people with little to no experience in digital drawing ('novices'). This distinction refers to the expertise in digital drawing, not in using our system. All participants were unfamiliar with the interaction concepts in our application. The actual study was conducted as follows: each participant was instructed to interact with the implementation using a pen tablet (Fig. 7). The participant was told to experiment with the program, while voicing his or her thoughts. During this phase, the participant received no explanation regarding the interface. After this phase, the interactions and tools were explained. The participant was then encouraged to use the tools to create a sketch.

For novices we observed that they could immediately start drawing with our tool, without any further explanation. After the new interaction concepts were explained, novices started using them, mostly in an exploratory and playful manner but also purposefully. Comments from novices included, e. g., that they enjoyed the ease of use and simplicity of our system. Expert participants, in contrast, learned and applied the new interactions even faster. Experts also directly compared our application with existing software, revealing both shortcomings and benefits of our system. The lack of an undo operation, specifically, was commonly identified as a limitation. Support for layers, usage of hotkeys, and a way to texture strokes were proposed as useful extensions. Nonetheless, the feedback was positive in general. People found the interface to be comfortable to work with. The canvas manipulations were particularly liked, especially the canvas rotation. The methods for creating stroke selections and applying rigid transformations to selections were described as comfortable and fluent. Unfortunately, we failed to better encourage the expert participants to experiment with our tools for local stroke reshaping. Thus, we could not gather valuable feed-

Figure 8: Artwork created with our sketching tool by expert study participants (left) and the first author (right).

back on the benefit of these tools. The accompanying video, however, demonstrates the efficiency and creative flexibility offered by the reshaping tools.

The analysis of the interaction and response from novice participants indicated that our interaction concepts can easily be learned and adopted and that they can facilitate the access to digital drawing for this user group. We see this suggested by the fact that people who had never worked with a drawing software before could immediately draw and execute other basic operations after a short explanation. It appeared that the reduced complexity of the interface encouraged people to use our application. Reducing the need for explicit mode changes was arguably beneficial for this purpose, judging from the fact that novices could execute various editing interactions without having knowledge about the internal operating modes of the system. In a traditional graphics program, novices would have had to learn how to select and apply different editing tools to achieve the same results. Thus, we assume the proposed concepts to be well suitable for an educational context. The techniques are similarly suited for use by experts, e. g., for rapid concept sketching, evident in the quality of images that can be created as shown in Fig. 8.

The study also helped us to identify some limitations of our system. A drawback of our design of the canvas translation interaction is that the canvas can only be translated in a direction pointing generally inward from the respective interface border. For example, a translation to the right can not be performed from the right border, as there is no screen space of moving the pen pointer to the right. This also leads to the arguably unnatural requirement to move in a inward direction to zoom the canvas both in and out from the interface corners, but this was not commented negatively by study participants. Another limitation of the system is the use of hold gestures. We initially used tap gestures for the line-select and line-erase interactions, but this lead to unintentional selections resp. erasures of strokes. We remedied this by using hold gestures, but these do not permit as fluid interactions as taps and demand people to learn how to perform them. Furthermore, our interaction concepts are not entirely self-explanatory as was evident from the explanations that participants required to start using the new methods. The reduced interface comes with the cost of limited visual indication for certain functions. Most of the interactions possible with the system require an explanation to become evident to the user, which could probably be improved upon by employing tool-tips or gesture previews.

To investigate the benefit of an orientable canvas further, we observed and questioned artists about their drawing behavior when working with a screen-aligned canvas. We learned that they typically tilt their heads, reposition their hands, or rotate the drawing tablet to facilitate drawing certain strokes. This can be interpreted as adaptations to a restriction imposed by a screen-aligned canvas, which motivates the usefulness of an adjustable one.

In a separate session, we asked a digital drawing artist to specifically compare our software with Autodesk's SketchBook Pro. The artist stated that he prefers our frame-based interaction techniques with the canvas and groups of strokes to the marking menu strategies used in the professional software. Apart from that, he also noted similar shortcomings with respect to the lack of advanced drawing functionality as noted above, but implementing a complete drawing application is beyond our scope.

6 Conclusion

With the goal of providing a minimalistic and intuitive interface for digital freehand sketching, we designed and explored new ways in which artists can sketch and interact with both strokes and the canvas (see example results in Fig. 8). In general, we use the interaction with the frame of the canvas and of stroke selections to apply canonical transformations to these elements without the need for explicit mode changes. We identify some limitations of using the interface border for interaction, emerging from the restriction to 'inward' directions for gestures, and we propose ways of dealing with this problem. We also found that using a tap gesture interferes with drawing short strokes, and that replacing it with a hold gesture does not permit as

fluid interaction as taps. Apart from that, the choices we made for mapping canvas transformations to border gestures are arbitrary to some extent. Alternative schemes of mapping those transformations could be devised, and it would be interesting to experimentally compare different schemes. In our solution, however, we aimed at providing an intuitive and consistent set of gestures which emulate the feel of working with a physical canvas. In the informal evaluation our interaction elements have shown to be well-received in general. People could successfully learn the interactions in a short period of time. Specifically the use of location-sensitive, contextual gestures rather than explicit mode changes allows us to provide a fluid transition between interaction techniques that are essential for rapid concept sketching and digital drawing. We hope that these interaction elements will inform the development of future sketching tools. Our techniques can be provided as a useful interface alternative in combination with the much more elaborate features such programs provide.

References

[ABCO*08] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point Set Surfaces. In *Proc. Visualization* (2008), IEEE Computer Society, Los Alamitos, pp. 21–28. DOI: 10.1109/VISUAL.2001.964489

[AG04] APITZ G., GUIMBRETIÈRE F.: CrossY: A Crossing-Based Drawing Application. In *Proc. UIST* (2004), ACM, New York, pp. 3–12. DOI: 10.1145/1029632.1029635

[BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: ILoveSketch: As-Natural-As-Possible Sketching System for Creating 3D Curve Models. In *Proc. UIST* (2008), ACM, New York, pp. 151–160. DOI: 10.1145/1449715.1449740

[BBS09] BAE S.-H., BALAKRISHNAN R., SINGH K.: EverybodyLovesSketch: 3D Sketching for a Broader Audience. In *Proc. UIST* (2009), ACM, New York, pp. 59–68. DOI: 10.1145/1622176.1622189

[Bru98] BRUIJNS J.: Quadratic Bézier Triangles as Drawing Primitives. In *Proc. Workshop on Graphics Hardware* (1998), ACM, New York, pp. 15–25. DOI: 10.1145/285305.285307

[BWL04] BAXTER W., WENDT J., LIN M. C.: IMPaSTo: A Realistic, Interactive Model for Paint. In *Proc. NPAR* (2004), ACM, New York, pp. 45–56. DOI: 10.1145/987657.987665

[DCL08] DAVIS R. C., COLWELL B., LANDAY J. A.: K-Sketch: a 'Kinetic' Sketch Pad for Novice Animators. In *Proc. CHI* (2008), ACM, New York, pp. 413–422. DOI: 10.1145/1357054.1357122

[ESAV99] EL-SANA J., AZANLI E., VARSHNEY A.: Skip Strips: Maintaining Triangle Strips for View-Dependent Rendering. In *Proc. Visualization* (1999), IEEE Computer Society, Los Alamitos, pp. 131–138. DOI: 10.1109/VISUAL.1999.809877

[FBKB99] FITZMAURICE G. W., BALAKRISHNAN R., KURTENBACH G., BUXTON B.: An Exploration into Supporting Artwork Orientation in the User Interface. In *Proc. CHI* (1999), ACM, New York, pp. 167–174. DOI: 10.1145/302979.303033

[HBRG05] HINCKLEY K., BAUDISCH P., RAMOS G., GUIMBRETIÈRE F.: Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli. In *Proc. CHI* (2005), ACM, New York, pp. 451–460. DOI: 10.1145/1054972.1055035

[HGA*06] HINCKLEY K., GUIMBRETIÈRE F., AGRAWALA M., APITZ G., CHEN N.: Phrasing Techniques for Multi-Stroke Selection Gestures. In *Proc. GI* (2006), Canadian Information Processing Society, Toronto, pp. 147–154.

[HL94] HSU S. C., LEE I. H. H.: Drawing and Animation Using Skeletal Strokes. In *Proc. SIGGRAPH* (1994), ACM, New York, pp. 109–118. DOI: 10.1145/192161.192186

[HZS*07] HINCKLEY K., ZHAO S., SARIN R., BAUDISCH P., CUTRELL E., SHILMAN M., TEN D.: InkSeine: In Situ Search for Active Note Taking. In *Proc. CHI* (2007), ACM, New York, pp. 251–260. DOI: 10.1145/1240624.1240666

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A Sketching Interface for 3D Freeform Design. In *Proc. SIGGRAPH* (1999), ACM, New York, pp. 409–416. DOI: 10.1145/311535.311602

[KCST05] KRUGER R., CARPENDALE S., SCOTT S. D., TANG A.: Fluid Integration of Rotation and Translation. In *Proc. CHI* (2005), ACM, New York, pp. 601–610. DOI: 10.1145/1054972.1055055

[KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A.: WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics 21*, 3 (July 2002), 755–762. DOI: 10.1145/566654.566648

[LHGL05] LI Y., HINCKLEY K., GUAN Z., LANDAY J. A.: Experimental Analysis of Mode Switching Techniques in Pen-based User Interfaces. In *Proc. CHI* (2005), ACM, New York, pp. 461–470. DOI: 10.1145/1054972.1055036

[RT09] ROTH V., TURNER T.: Bezel Swipe: Conflict-Free Scrolling and Multiple Selection on Mobile Touch Screen Devices. In *Proc. CHI* (2009), ACM, New York, pp. 1523–1526. DOI: 10.1145/1518701.1518933

[SFLM04] SAUND E., FLEET D., LARNER D., MAHONEY J.: Perceptually-Supported Image Editing of Text and Graphics. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 728–728. DOI: 10.1145/1015706.1015787

[Smi78] SMITH A. R.: *Paint*. Technical Memo 7, NYIT Computer Graphics Lab, July 1978.

[SWSJ05] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J.: ShapeShop: Sketch-Based Solid Modeling with BlobTrees. In *Proc. SBIM* (2005), Eurographics Association, Aire-la-Ville, Switzerland, pp. 53–62. DOI: 10.2312/SBM/SBM05/053-062

[SWT*05] SEAH H. S., WU Z., TIAN F., XIAO X., XIE B.: Artistic Brushstroke Representation and Animation with Disk B-Spline Curve. In *Proc. CHI* (2005), ACM, New York, pp. 88–93. DOI: 10.1145/1178477.1178489

[VLC*08] VANDOREN P., LAERHOVEN T. V., CLAESEN L., TAELMAN J., RAYMAEKERS C., REETH F. V.: IntuPaint: Bridging the Gap Between Physical and Digital Painting. In *Proc. Tabletop* (2008), IEEE Computer Society, Los Alamitos, pp. 65–72. DOI: 10.1109/TABLETOP.2008.4660185

[YSI*10] YU L., SVETACHOV P., ISENBERG P., EVERTS M. H., ISENBERG T.: FI3D: Direct-Touch Interaction for the Exploration of 3D Scientific Visualization Spaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (nov / dec 2010). To appear.

[ZF99] ZELEZNIK R., FORSBERG A.: UniCam – 2D Gestural Camera Controls for 3D Environments. In *Proc. I3D* (1999), ACM, New York, pp. 169–173. DOI: 10.1145/300523.300546

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An Interface for Sketching 3D Scenes. In *Proc. SIGGRAPH* (1996), ACM, New York, pp. 163–170. DOI: 10.1145/237170.237238

[ZM06] ZELEZNIK R., MILLER T.: Fluid Inking: Augmenting the Medium of Free-Form Inking with Gestures. In *Proc. GI* (2006), Canadian Information Processing Society, Toronto, pp. 155–162.