

# GestureBar: Making Gestures Browseable, Discoverable, Learnable and Training-Free

---

*Andrew Bragdon*

**Master's Project** 2008

*Department of Computer Science  
Brown University*

## Abstract

The design of GestureBar is presented, a system for disclosing and teaching pen-based gestures, with the goals of making gesture-based applications approachable and intuitive, and eliminating the need for introductory training, videos, crib sheets, or other assistance – even for users who have no prior Tablet PC or pen-based gesture experience. The design draws from user interface concepts users are experienced with, such as toolbars, icons and tooltips, to create a top-level interface which is familiar and inviting. This top-level interface can be used to launch the Gesture Explorer, an interface which presents, and allows a user to browse, procedurally-animated gesture demonstrations, interactive practice areas, and additional feature information. The results of a preliminary pilot user study is presented, which indicate that the GestureBar approach outperforms a state of the art crib sheet, and that GestureBar is effective for teaching a variety of single-stroke and multi-stroke gestures from the Line-o-grammer diagramming application.

## Introduction

Pen-based, and multi-touch gestures have many benefits. They are inherently parameterized,

effectively allowing the user to specify “what to apply the command on” or “how much” for example, without additional widgets or dialog boxes. This physical unification implies that no additional explicit mode selections, for example clicking a zoom or move tool, are required to switch between commands. Because of these, and other advantages, gestures are arguably more “cognitively lightweight” and perhaps physically more efficient than conventional WIMP interfaces. The user does not have to leave their work area to select command buttons on the periphery of the screen, and many tasks can be simplified into a single interaction that has been committed to muscle memory.

Nonetheless gestures have historically always had one major drawback: they are not self-disclosing. WIMP interfaces display all of the available commands onscreen, allowing the user to browse for the command that they need when they need it. Gestures, on the other hand, are fundamentally different in that they do not involve buttons, menus or other onscreen UI, and so they are not inherently discoverable.

Thus, many gesture-based applications require the user to watch a training video, or review a gesture “crib sheet.” Other applications require a user to step through an introductory tutorial in which the user performs each gesture until they can execute it accurately.

However, these approaches are not satisfactory for a number of reasons. These approaches create a training phase that a user must get through, which creates a barrier of entry to the adoption of such software. Users have come to expect software to be so intuitive that no explicit training is required. Moreover, mandatory, automated introductory training

screens are not practical or efficient for large and complex applications. Many large applications have hundreds, even thousands, of commands. In addition, a given user is typically only interested in a subset of the available commands that is relevant to them. Even if a user were able to get through the necessary training, it is quite possible that they will have forgotten the precise gesture and its nuances by the time they want to use it.

But even more importantly, users are familiar with the pervasive WIMP interface paradigm, in which they first form a mental goal of what they want to do, then search for a command to accomplish this task, and lastly once the command is found, make use of it. The essence of this process works well. In fact, it is often taken for granted that most users can begin using popular WIMP applications without any training.

Why should this be any different for gestural commands? Users should be able to learn gestural commands on the fly as they are needed. They should be able to browse for the command that they need. They should not have to refer to help files or crib sheets. Absolutely no training, informational videos or coaching should be required or needed. And they should feel comfortable learning a gesture-based application even if they have no prior experience with pen computing or gestural commands.

Users should be able to walk up to a gestural application, and without having ever used a Tablet PC before, begin using it right away – just as they might begin using a WIMP application right away.

The goal of the GestureBar approach is to make pen-based gestures as intuitive, approachable,

browseable, learnable and self-disclosing as WIMP commands – while still retaining all of the advantages of gestures.

## Prior Work

GestureBar builds on parts of the work, “Contextual Animation of Gestural Commands” by Kurtenbach, et al. (1). In (1), a user may perform a press-and-hold operation with the pen, at which point a crib sheet containing contextually relevant gestures is presented. If a gesture in the crib sheet, displayed as a static image of the gesture, is pressed on, a series of animations illustrating examples of the gesture will play over the document. Users may then trace the gestures to execute the corresponding commands. There are a number key differences between, and extensions by GestureBar of this system, however. The approach used in (1) requires the user to know the press-and-hold “gesture” that brings up contextually relevant commands; a significant handicap for first-time pen computing users. The approach taken with GestureBar utilizes a top-level toolbar-like interface instead, eliminating the need to know the press-and-hold gesture. GestureBar instead integrates context by using canonical examples, rather than contextual examples used in (1). Another key difference is that in (1), a user learns/practices a gesture by tracing it after it has been demoed. However, this is potentially counterproductive if a user does not perform a gesture correctly – causing adverse effects in their document. The approach taken in GestureBar is to instead provide a dedicated, scoped Practice area for each gesture. Another drawback of the contextual animations approach used in (1) is that more complex “approaches” or “strategies” cannot be readily explained – as canonical examples are better suited to illustrate overall methods, approaches

and strategies. A number of other additions and extensions to (1) exist in GestureBar as well (see Final System Design, below).

Another approach, taken by Forsberg et al. in “Tablet PC Music Composition Tool,” (2) based on their music notepad work (3) is to provide a tutorial in the form of a crib sheet which the user can open by pressing a “Demo” button. The user can then copy/trace each note gesture to learn them. However, this approach forces the user to undergo training. A user cannot sit down and begin using the application by looking for relevant commands; instead they must first review and train with the crib sheet. In addition, the crib sheet cannot be quickly open and closed for reference as opening it necessitates clearing the existing document.

Zelevnik et al. took a different approach in the *MathPaper* application (4), which makes use of an interactive training tutorial. The user is asked to perform a series of actions via a text description, and can click on a “Show me” button to receive an animated example. Once they correctly perform an action, they may advance to the next task. This approach, while probably effective for teaching a user to use the system, again requires a rigorous, up-front training process for a new user; effectively requiring a user to invest time in the application before they can use it. In addition, if a user forgets how to use a specific feature, they will have to go through the training process again rather than being able to reference/practice that one gesture or action.

The *Mouse Gestures* add-on for the Mozilla Firefox web browser (5) makes use of a static, monochrome cheat sheet docked to the side of the screen, which users can open (and leave open) to learn, and reference mouse gestures for executing various browser commands.

Gestures are grouped into categories, and each gesture is shown with a text label and a black and white gesture diagram, with a black dot indicating the start of the gesture and a black line indicating the path of the gesture.

GestureBar improves on this design in a number of key ways. The animated disclosure in GestureBar allows for multi-stroke gestures, and helps to reinforce the ordered nature of even a single-stroke gesture. Since gestures are fundamentally a motion of the hand, the added time information in an animation is valuable. In addition, the cheat sheet approach does not include a practice area and so the user cannot try a gesture in a scoped environment, receiving feedback on their performance from the system. The user must instead experiment in their real browser, with the very real possibility of invoking the wrong command by accident – a very jarring experience, as not only is the user confused as to why the expected operation did not take place, they will be confused as to why a different, probably unrelated operation, did in fact take place. Because the docked crib sheet does not include familiar icons, but instead the images of the gestures themselves (which are often abstract in nature, for example a line down and to the right for “close window”), the crib sheet is not as inviting as the GestureBar which can make use of familiar, ubiquitous icons. Along these lines, the list of gestures is quite long when fully expanded and so it actually scrolls off the screen at the standard resolution of 1024x768. This makes the list harder to navigate, and also harder to comprehend as a whole. The GestureBar, on the otherhand, is designed to fit onscreen at standard resolutions, and makes use of groupings to help users understand the space of commands available. In addition, the additional space afforded by the Gesture Explorer drop-down interface provides additional room for a

text description of the command and the gesture, and also a larger, richer gesture diagram which can point out essential features (such as “Right Angle” or “Closed Loop”).

Marking menus have been shown to be an effective way to disclose, and teach over time, a specific class of menu-like unparamaterized gestures (6). GestureBar is intended to address a different, broader class of gestures – and in fact, can be used in conjunction with marking menus, as is the case in Line-o-grammer (7).

## Final System Design

GestureBar is a middleware platform for creating user interface content specific to an application. Thus, these two aspects – the middleware platform, and the application-specific content – form the crux of GestureBar. The middleware platform governs all aspects of the way the GestureBar works, looks and feels – and the application-specific content takes the form of XML markup (using the XAML standard) which defines gesture information, icons, tool tip text, and so on.

In order to test GestureBar effectively, an underlying application with gestural commands was needed. Thus, GestureBar was tested within a diagramming application called Line-o-grammer (7), and all of the content was tailored for this application.

GestureBar is comprised of a top-level interface which is always visible, and a drop-down Gesture Explorer interface which can be opened on demand. Application-specific content is displayed in both.

## Top-Level Interface

GestureBar functions, at the top level, just like a toolbar or ribbon (8). Moreover, because the

commands presented in the GestureBar are always visible, the user can browse all of the commands available in the application from one place no matter what command they are looking for. The contextual design of the approach taken in (1) was avoided because users must guess the correct context to press-and-hold in to find a command; for example a user looking for the scrolling gesture mentioned in (1) would not be able to find these gestures unless they discovered that they must press-and-hold in the margins of the drawing area; thus the contextual approach effectively makes the available commands harder to browse. In a large application, many possible contexts may exist – making it very difficult for a user to find the command they are looking for due to the aforementioned press-and-hold command search process.

The top-level interface is comprised of a series of tabs. Each tab is divided into sections, and each section contains one or more buttons. These logical groupings were introduced to help users understand the space of commands available. The user may switch tabs by clicking on the appropriate tab, changing the buttons displayed.

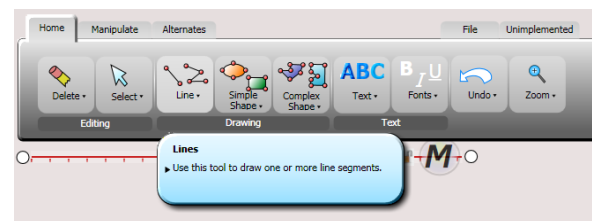


Figure 1 – The top-level interface of GestureBar, shown here with content from Line-o-grammer

Each button consists of an icon and text. If the user hovers the pointer (using the pen or a mouse) over a button, a tooltip will appear. Tooltips display the name of the command and

a description of the feature(s) associated with the button.

GestureBar's use of icons rather than static images of gestures, such as those used in (1), can be advantageous to new users for many commands. Consider for example, the command "Zoom" which is often represented by the ubiquitous "magnifying glass" icon – this image is more likely to be familiar/meaningful to a new user than an image of the double circle gesture that might be used for zoom in a given application. Moreover, the overall approach of using a toolbar/ribbon-like top-level interface makes GestureBar more approachable due to its familiar appearance.

Many applications have a need for both gestural and WIMP commands. For example, in Line-o-grammer the delete command is gestural as this creates a fluid and cognitively lightweight interface for deleting objects and parts of objects. Page Setup, on the other hand, is a command which is seldom used on average and does not require parameters – since the command simply launches the standard Page Setup dialog box. Thus, a unique gesture specific to Page Setup is not necessary or beneficial, and so a simple WIMP command can be used instead.

Thus, there are two types of buttons: gesture buttons and WIMP buttons. Gesture buttons include a dropdown "chevron" on the right-hand side of the text. When a user clicks a gesture button, the Gesture Explorer interface appears via a transition animation below the button. When a user clicks a WIMP button, the corresponding command (specified in content XML) is executed.

## Gesture Explorer Interface

The Gesture Explorer interface consists of a series of tabs for each gesture variant, a gesture demonstration unit, a gesture practice unit, and a text description of the feature.

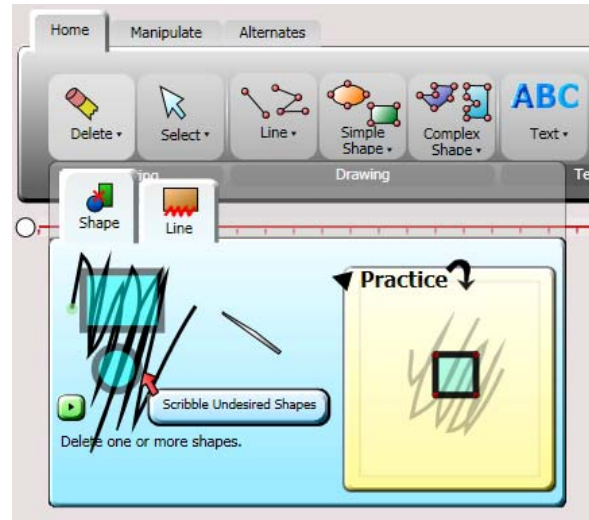


Figure 2 – The Gesture Explorer interface drops down when a user taps on a button in the top-level UI, and is comprised of gesture variation tabs, the Gesture Demonstration Unit, the Gesture Practice Unit, and a feature description

Each gesture variant tab consists of an icon and a text label. A user may switch gesture variants by tapping on a tab. The approach used by (1) was considered, in which each gesture variation animation is played sequentially, but this approach does not allow the user to see the completed gesture after the animation has finished, and it also forces users to skip animations to access the desired variation (or simply watch a potentially long series of animations). The gesture variant tab approach allows the user to both see the completed gesture and also easily browse to the desired gesture variation.

## Gesture Demonstration Unit

Gestures are presented to the user via a procedurally generated demonstration animation of the gesture. Recorded stroke data



is used to animate a virtual pen icon as if an instructor were demonstrating the gesture. The pen animates “down” at the start of a stroke, and “up” at the end to help communicate the nature of multi-stroke gestures. A green dot is displayed at the beginning of a stroke to indicate its starting location. A context image is displayed as well, to help communicate the contextual or parameterized nature of some gestures. For example, in the delete gesture from Line-o-grammer, the animation depicts a scribble scribble gesture over a background context of several shapes.

GestureBar shows context in-place by integrating context into the animations shown in the Gesture Explorer, and by using canonical examples – rather than the contextual examples used in (1). This allows all commands to be displayed all the time, and moreover, it allows gestures which use parameters to be more clearly presented. Consider, for example a lasso-like gesture for “Select” which asks a user to draw a lasso around the items they wish to select. In GestureBar this can be shown directly through one or more canonical examples, shown in the Gesture Explorer. However, in (1) such a parameter becomes increasingly difficult to show with a static set of example animations. Consider a case in which a number of complex shapes are arranged together, and the user performs a press-and-hold on one of the shapes. If the user then taps on the “Select” gesture, how will the system dynamically show a lasso animation which only lassoes one shape, and not part (or all of) a smaller neighboring shape which fits into it, when using a static set of animations? It is straightforward to conclude that it would be very difficult to create contextual gesture examples in the general case. This problem is avoided through the GestureBar’s use of canonical examples.

In initial pilot testing, it was found that many users were not noticing essential aspects of certain gestures. For example, the Recognize Text gesture in Line-o-grammer requires a user to underline the handwritten text and then perform a vertical flick at the end of the underline at a sharp angle. Some users did not understand that the sharp angle was necessary and drew it at a more relaxed angle, resulting in recognition failure. As a result gesture detail tooltips were added; after the gesture animation is complete, gesture detail tooltips slide in from the right and point out essential aspects of certain gestures. Each detail tooltip includes a red arrow and text. As an example, the tooltip “Right Angle” transitions in pointing out this essential attribute of the Recognize Text gesture in Line-o-grammer.

A green replay button is included below the animation, as during pilot testing many users expressed the need to “watch it again.”

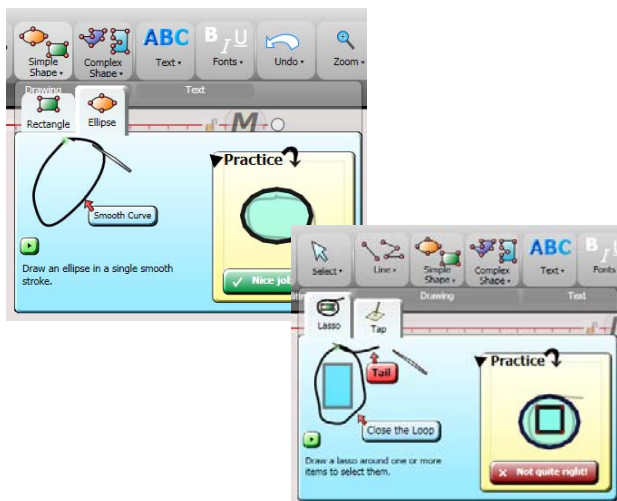
During pilot testing, some users mentioned that they missed the first part of the animations because they started immediately. As a result, a brief transition animation was added when the user first opens a Gesture Explorer (switching gesture variation tabs does not trigger the animation). The animation simply consists of an expanding rectangle, which serves to introduce a short delay and also catches a user’s attention.

### **Gesture Practice Unit**

A user’s understanding of a gesture, and the overall concept of gestural commands, can be reinforced by using the Practice Unit, after watching the demonstration animation. Gestures can be tried in the Practice Unit by tracing the example overlay, and the user receives feedback via the result tooltip. If the user successfully executes the gesture, a green

“Nice job” button is displayed via an animation. If the user is unsuccessful, a red “Not quite right.” Button is displayed instead. Tapping the button will reset the practice area; allowing them to try again.

In (1) a user may perform any gesture on any element, which could quite conceivably cause problems and frustration if the user does not perform the gesture correctly, and another action than the one intended takes place – as the actions taken while tracing directly affect the user’s document. For example, consider a scenario in which a user attempts to perform a move gesture but instead performs a delete. The GestureBar approach of an integrated practice area as part of the Gesture Explorer interface avoids this problem. Because the practice area is scoped to the specific command the user has chosen, feedback on the user’s execution of the gesture can be provided. Moreover, the practice area is separate from their document which means that they may safely experiment in it.



**Figure 3 – The Gesture Practice Unit gives the user feedback on their performance of a gesture, shown here for the Ellipse gesture (top left) and Lasso Select gesture (bottom right) from Line-o-grammer**

The try it area instantiates an instance of the main application’s interaction canvas inside the practice area; thus the practice unit behaves exactly as the real application does. However, actions taken inside the practice unit by the user will not affect their real document, giving them the ability to experiment without fear of accidentally altering their work.

The practice unit also incorporates context as well as a tracing overlay. For example, the try it unit for the Lasso Select gesture in Line-o-grammer incorporates a rectangle for the user to select, and a semi-transparent overlay of the Lasso Select gesture around it.

### Gesture Invocation Notifications

To help reinforce the connection between gestures executed in the application’s interaction canvas and the GestureBar – and to give feedback to users that a command has been successfully invoked, the icon and name of a command is faded when a gestural command is invoked. Thus, for example, if the user performs the delete gesture on a shape, the delete icon will be faded out next to the user’s pen-up position (on the right side for left-handed users, and on the left side for right-handed users; the operating system-level setting for handedness is used).

### GestureBar Content

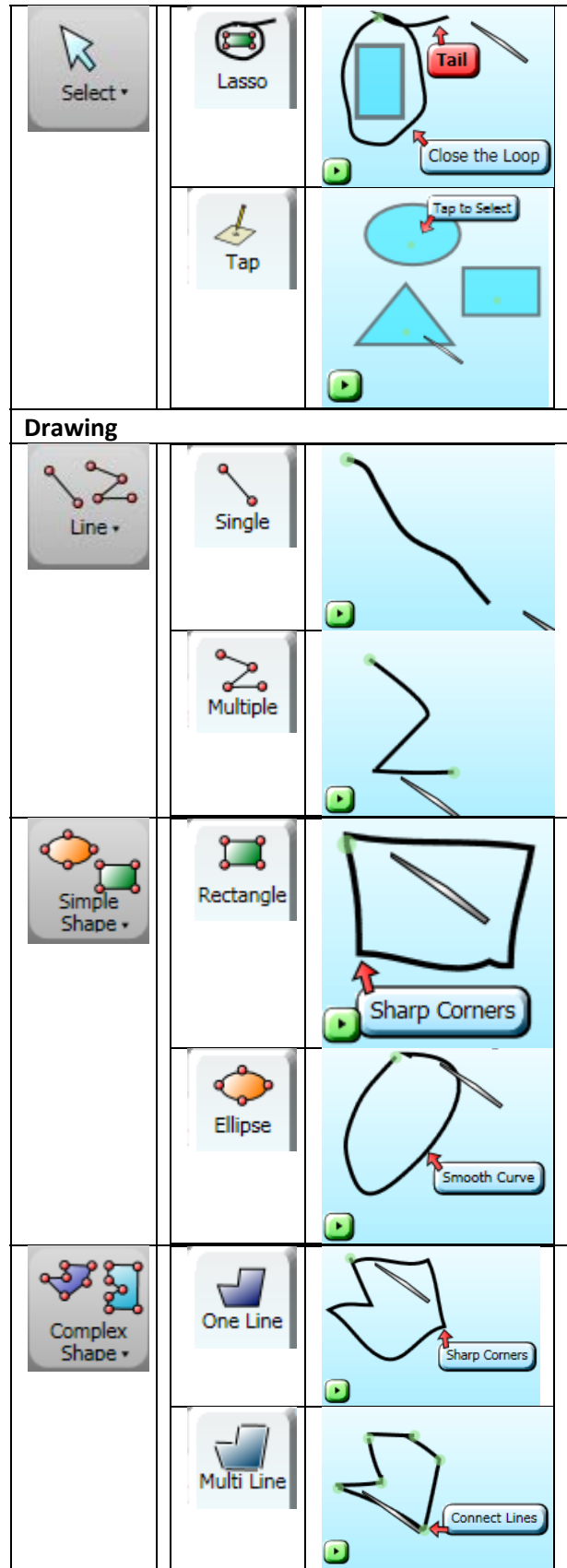
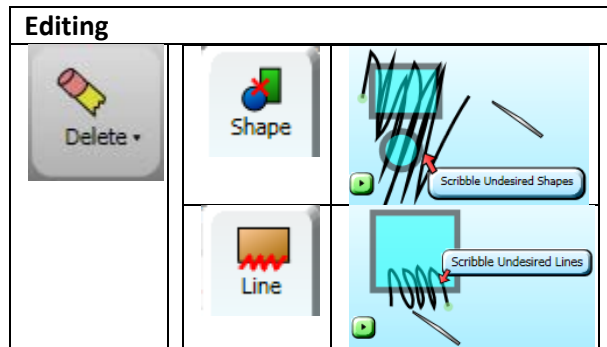
The GestureBar platform displays content based on XAML (a Windows Presentation Foundation standard) property definitions stored in XML within an application. The GestureBar supports Microsoft Expression Blend integration, making it a straightforward process to create content for the GestureBar, as the WYSIWYG designer in Expression Blend can be used to create the icons, context information, text, gesture stroke paths, and tab sections and layout.

## Designing Content for Line-o-grammer

Because the initial content created for GestureBar is based on the features of Line-o-grammer, there were a range of gestures to disclose to the user, including single-stroke gestures, parameterized gestures, and multi-stroke gestures. It is important to note that the overall effectiveness of GestureBar for a specific application is impacted by the quality of the content created for GestureBar.

As the design of GestureBar evolved based on pilot testing, so did the content design of GestureBar. Specific changes were made to the organization of gestures, and the gesture examples were revised. Additional content was added to provide more examples to communicate the nuances of some features, and additional detail was added in the form of Practice Unit traceable overlays and detail tooltips to point out the essential aspects of the gestures.

The GestureBar content created for the Home tab of Line-o-grammer is presented below. The left-most column shows the top-level button, the center column shows each gesture variation tab, and the right column shows the Gesture Demonstration Unit after the animation is completed.






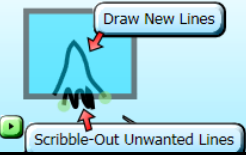


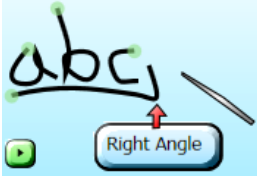

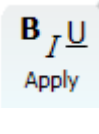
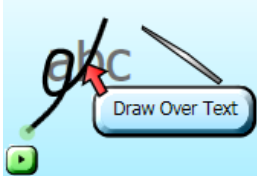

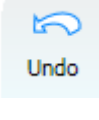

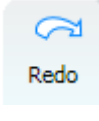
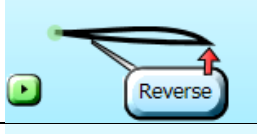



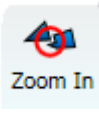
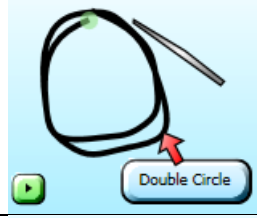
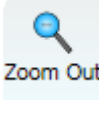
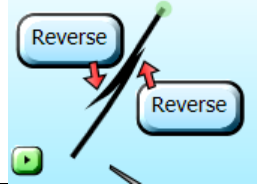
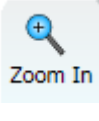
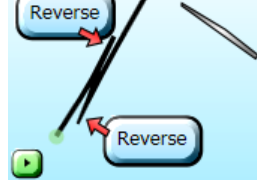
		
<b>Text</b>		
		
		
<b>(Untitled Section)</b>		
		
		
		
		
		
		

Table 1 – GestureBar content for the Home tab in Line-o-grammer

In addition, the content of the GestureBar was also tailored to communicate a high-level “strategy” for using Line-o-grammer. In Line-o-grammer, lines are the fundamental primitives, rather than shapes, which means that users can draw in additional lines and remove partial line segments from existing shapes. To emphasize this difference to new users, additional variations of the Delete gesture and Complex Shape gesture were added. A variation of Delete showing how to delete one line segment from a square was added to Delete, and a variation of Complex Shape titled “Modify Shape” was added, showing a notch being drawn onto the edge of a rectangle, and then the excess line being erased to form a closed polygon.

This approach of illustrating higher-level strategies can be shown with canonical examples, and is an advantage of the GestureBar approach. However, because the example given may not be the result a user desires, the contextual approach used in (1) would run into difficulties. The system would have to make an assumption about what geometry the user would want to add, and what geometry the user would want to remove. Since the user is then expected to trace the animation, the user would end up with a result that is very likely to not be the desired one. Secondly, the system will have to handle all possible input geometries via some sort of procedural animation system (which may not be as simple as a rectangle) – a hard problem in its own right.

## Evaluation

To evaluate the final design of GestureBar, a pilot user study was conducted with four test subjects. Two of the subjects were tested solely with GestureBar, and two of the subjects were tested on both a crib sheet and GestureBar.

The overall goal of the evaluation was to determine whether a user, who had no prior experience with Tablet PCs or pen-based gestures, could accomplish a specific series of tasks with no training, introduction or assistance. This training-free approach was critical to evaluating the goal set forth for the project. In all tests, subjects were told that they would be using an application for creating diagrams, that they would be given a series of tasks to accomplish, that the person running the study would not be able to answer questions related to the software, and that their use of the software would be observed.

### First Test Suite

The first subject was tested against a near-final version of GestureBar which did not include tracing overlays in the Practice Urea, several changes to content, the addition of the delay animation in the demonstration unit, the addition of the replay button, or the addition of the attract animation in the Practice Unit. The test suite included a series of specific tasks such as “draw a rectangle,” “draw an ellipse,” “zoom in on a corner of the rectangle,” “label the diagram with the characters ‘abc’,” “create the following diagram,” etc. For a full list of the tasks, see Appendix A. Both subjects had no prior experience with a Tablet PC or performing pen-based gestures.

The first test subject was a “novice” computer user, and a student at Brown University. She was able to perform all of the tasks successfully

with no assistance. However, she did encounter some problems. For the Zoom In animation, she missed the beginning of the animation and had to watch the animation several times to learn the gesture. In addition, she had trouble replaying animations as the replay button had been removed in the build. She also had difficulty learning the alternate Zoom In command which uses a double-hitch gesture. She also did not notice the Practice Unit for most of the test. Overall, though – with the exception of the alternate Zoom In gesture – she was able to accomplish all of the tasks successfully and on the first or second attempt.

The second subject was an experienced computer user and a recent graduate of Brown University. The second test subject was run against a further refined version of GestureBar which included a replay button, improved GestureBar content, traceable overlays in the Practice Unit, and green dots which indicated the start of a gesture in the Demonstration Unit. She was able to accomplish all tasks on her first try. She noticed and made use of the Practice area right away and was even able to learn the alternate “double-hitch” Zoom In gesture easily, possibly due to the improvements, which the first subject had had so much trouble with. Overall, the second subject was able to accomplish the tasks very quickly and smoothly.

### Second Test Suite

After the initial success of the first two subjects, the test suite was expanded to include more tasks, as well as a set of descriptive general tasks where a user was given a specific goal such as “Make a simple house with a door and window, and label the house ‘House’ in underlined text. Don’t forget the chimney.” See Appendix A for a full list of tasks.

In addition, to create a baseline for a comparison, a crib sheet in the format of the Firefox Mouse Gesture crib sheet was created. This crib sheet format was chosen as the current “state of the art” in gesture crib sheets, after examining a number of other similar approaches. This format is similar to the format used by Graffiti in the Palm Pilot handhelds, and other gesture systems. The approach used in InkSeine (9) was also considered, but it does not appear to directly translate to general applications, as the gestures in that system have a contextual (and often localized) nature. The Firefox Mouse Gesture approach appeared to be the best example available of a general gesture crib sheet. A “crib sheet mode” was added to Line-o-grammer in which the crib sheet was displayed with content for Line-o-grammer on the left-hand side of the application, just as in the Firefox add-on. Comparable/analogous content was used to the GestureBar, and command names were kept the same as much as possible. All gestures were displayed in the black-and-white diagram style used in the Firefox add-on. The crib sheet was not interactive, except for a vertical scrollbar.

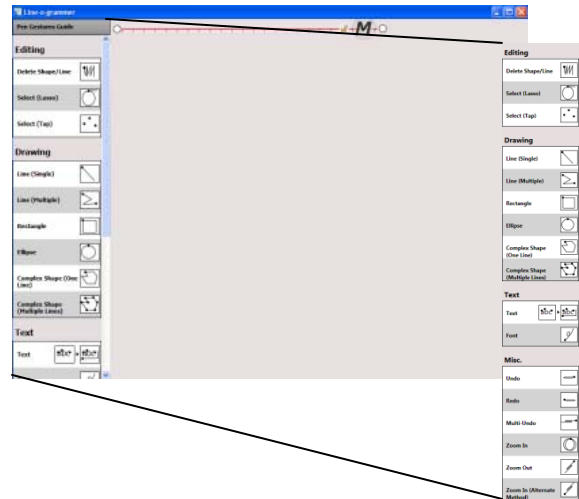


Figure 5 – Line-o-grammer in crib sheet-mode (left), full crib sheet contents (right)

Each test subject was first tested against Line-o-grammer in crib sheet mode first, and then against Line-o-grammer in GestureBar mode – with the same set of tasks. Initially, it was planned to use separate subjects for the crib sheet and GestureBar modes, however, subjects ended up performing very poorly with the crib sheet mode – and were unable to complete many tasks. Thus, as this was still a pilot study, it seemed worthwhile to run the same subjects again on the GestureBar mode to see if they were able to learn the same gestures and accomplish the same tasks which they failed on in the crib sheet mode.



Figure 4 – Gesture crib sheet from the Mouse Gesture add-on for Mozilla Firefox

The third subject was a computer user of “intermediate to expert” experience and is a student at the Rhode Island School of Design. He ran into a very large number of problems while using crib sheet mode. He did not appear to understand, and later confirmed, that he believed the crib sheet was a series of buttons used to perform mode switches. It is worth noting here that the crib sheet does not respond in any way to taps, and can only be scrolled vertically. It seems, however, that because no introduction was given explaining

what the purpose of the crib sheet was, he had a great deal of difficulty understanding its purpose. After many false starts, he did eventually determine that he needed to draw certain gestures onscreen to execute commands, but he still appeared to believe that he needed to tap on the crib sheet to execute the command. It also seemed apparent that he did not fully understand the concept of a gesture, as in many cases he would first try to use a command by tapping on it in the crib sheet and then circling what he wanted it to apply to. Most commands took a minimum of 4 or 5 attempts, and much tapping on the crib sheet – and many commands required more attempts than this. He was unable to successfully complete any tasks related to text even after numerous attempts, forcing us to move on to the next task when this occurred. He appeared to have difficulty locating commands, and often scrolled the list all the way up and then all the way back down repeatedly while searching for a command. In one particular instance, he said out loud that he was looking for the Undo command and after searching the list he was unable to find it. Later, however, he looked again and he was able to locate the Undo command in the list. By the end of the test, he still did not appear to understand that the images in the crib sheet were gesture diagrams – and in fact afterward, he said that he thought they were command icons. He also complained that the crib sheet was “hard to scroll” and that it was “difficult” to find a command. Overall, it seemed clear that he did not understand the crib sheet/gesture UI metaphor.

Thus, the study was continued by running him against Line-o-grammer in GestureBar mode to determine if similar problems occurred. The study started by running him against the family

tree task (see Appendix A for details), which involves drawing boxes, lines and typeset text. He immediately began using the GestureBar and learned the Rectangle gesture by using the Simple Shape button and its Practice Unit. He then learned the Select gesture by using its Practice Unit (a gesture he had a lot of trouble with, and which he did not appear to have fully learned before) and was able to rearrange some of the rectangles he had created after selecting them. He then used the Text Practice Unit to learn the Text gesture and was able to immediately start adding text to his diagram, a task he had tried repeatedly before and had never succeeded with. He was able to complete all of the tasks he was given on the first try. Afterward, he said that the “practice area was very helpful,” and that it “feels safe” to try a gesture in. He also said that the “animation demos were very helpful” and that “the categories helped a lot” when looking for a command. He also said that the icons in the GestureBar were helpful for finding a command.

The fourth test subject is a computer user of “intermediate” experience and is an employee of Brown University. While using Line-o-grammer in crib sheet mode, she was unable to successfully complete any tasks – even after repeated attempts, with the exception of the first two (“make a rectangle,” and “make an ellipse inside the rectangle”). Like the first test subject, she appeared to believe that the crib sheet was actually a series of buttons and that the gesture diagrams were actually icons. She repeatedly tapped on crib sheet items and then circled shapes in the interaction canvas in an attempt to execute the commands. She did not appear to understand that the crib sheet was a list of gestural commands.

As before, she was then run against Line-o-grammer in GestureBar mode. When using the GestureBar she was now able to complete all of the tasks on her first attempt, or on the second attempt in one instance; with the exception that she was unable to perform the alternate double-hitch Zoom In gesture successfully. She was able to use the Practice unit to learn gestures, and appeared to immediately understand the concept of gestural commands (a concept which she had previously not grasped). Overall, it seemed that she was successful with the GestureBar in learning and accomplishing the tasks given.

## Discussion/Future Work

Pilot testing has shown that users with no prior experience with Tablet PCs or pen-based gestures are able to begin using a new software application and accomplish specific and general tasks with no introduction whatsoever. In addition, pilot testing also strongly suggests that users have significant problems – to the point of actually not being able to complete any tasks in the case of one subject – when put in the same situation and given a crib sheet of the gestures available in a system, instead of the GestureBar. Moreover, when these users are then switched to the same application using GestureBar they are able to successfully complete the same tasks. In addition, it is apparent from the initial testing that users understand the underlying test application, Line-o-grammer, and its user interface much more clearly when the GestureBar is used.

However, there are many opportunities for future work. The initial pilot testing included only four test subjects and is thus not statistically significant. More extensive, statistically significant user testing of

GestureBar and a competing crib sheet is warranted. New metrics for speed, understanding, and user satisfaction should be developed to measure the success in such a study.

In addition, the author believes there is an opportunity to add an additional test in which the approachability of a crib sheet-based gestural system, and a GestureBar-based system is compared. In such a test, a user could be told that they are testing an application for creating diagrams, and that at any time during the test they can request a different application. However, if they change applications they will not be able to change back. Thus, a separate set of test subjects could be brought in to measure the “give up rate” – in other words, the percentage of users who become unhappy enough with either of the two systems to request a different one. If the GestureBar “give up rate” is much lower than the crib sheet “give up rate,” this would be a significant demonstration of the value of the GestureBar approach – beyond the metrics measured above.

Future work which investigates adding multi-touch gesture support to the GestureBar is also warranted. Moreover, future work which investigates a multi-modal version of GestureBar, which incorporates voice recognition, multi-touch, pen-based gestures, keyboard interaction, and other interface paradigms together in one system is also warranted.

Issues also remain with GestureBar, and the opportunity for further refinement exists. Users do not always make use of the Practice area. In addition, the Practice area does not give detailed feedback on why a user has unsuccessfully performed a gesture. In future



studies, exploring a slightly modified crib sheet which includes more explanatory text is also warranted.

## Acknowledgements

The author wishes to acknowledge the assistance, advice and sponsorship of Andries van Dam, Robert Zeleznik, and Andrew Forsberg.

## Conclusion

GestureBar has been presented, a novel system for making gestures browseable, discoverable, learnable and training-free. The details of the approach, a toolbar-like top-level interface, a drop-down Gesture Explorer interface hosting an animated Gesture Demonstration Unit and Gesture Practice Unit, and the content created specifically for the gesture-based Line-o-grammer application were presented. In addition, the details of the iterative design process used were presented. The results of a pilot study which tested GestureBar separately, and also compared it against a “state of the art” gesture crib sheet were also presented. Preliminary testing indicates that this is a promising and successful approach, and that the GestureBar performs significantly better than a crib sheet.

## References

1. *Contextual Animation of Gestural Commands*. **Kurtenbach, G. and Moran, T.** 1994. Eurographics Computer Graphics Forum. Vol. 13(5), pp. 305-314.
2. **Microsoft.** Tablet PC Music Composition Tool. [Software Application]. 1998.
3. *The music notepad*. **Forsberg, Andrew, Dieterich, Mark and Zeleznik, Robert.** 1998. Proceedings of the 11th annual ACM symposium on User interface software and technology. pp. 203-210.
4. *Designing UI Techniques for Handwritten Mathematics*. **Zeleznik, R., Miller, T. and Li, C.** 2007. Eurographics Workshop on Sketch-Based Interfaces and Modeling.
5. Mouse Gestures 1.5.2 Add-on for Mozilla Firefox. [Software Application]. 2008.
6. **Kurtenbach, G.** The Design and Evaluation of Marking Menus. [Ph. D. Thesis]. s.l. : Department of Computer Science, University of Toronto, 1993.
7. **Zeleznik, Robert, et al.** Line-o-grammer. [Software Application]. 2008.
8. **Microsoft.** Ribbon User Interface, Office 2007. [Software Application]. 2007.
9. *InkSeine: In Situ Search for Active Note Taking*. **Hinckley, K., et al.** 2007. CHI'07. pp. 251-260.
10. *SATIN: A toolkit for informal ink-based applications*. **Hong, J. and Landay, J.** 2000. UIST'00 ACM User Interface Software & Technology. pp. 63-72.
11. *Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes*. **Wobbrock, J., Wilson, A. and Li, Y.** 2007. UIST'07 ACM User Interface Software & Technology. pp. 159-168.
12. *Fluid inking: augmenting the medium of free-form inking with gestures*. **Zeleznik, Robert and Miller, Timothy.** 2006. Graphics Interface. pp. 155-162.

13. *Can People Use Gesture Commands?* **Wolf, C. G.** 1986, ACM SIGCHI Bulletin, Vol. 18, pp. 73-74.

14. *Behavioral Experiments in Handmarks.* **Gould, J. D. and Salaun, J.** New York : ACM, 1987. Proceedings of the CHI + GI '87 Conference on Human Factors in Computing Systems and Graphics Interface. pp. 175-181.

15. *Automated explanations as a component of a computer-aided design system.* **Cullingford, R. E., et al.** 1982. IEEE Transactions on System, Man and Cybernetics. Vol. March/April, pp. 168-181.

16. *Coupling a UI framework with automatic generation of context-sensitive animated help.* **Sukaviriya, P. and Foley, J. D.** New York : ACM, 1990. Proceedings of the ACM Symposium on User Interface Software and Technology '88. pp. 152-166.

17. **Blinkenstorfer, C.** Graffiti. *Pen Computing.* 1995, 50, pp. 30-31.

18. *Modeling human performance of pen stroke gestures.* **Cao, X. and Zhai, S.** 2007. CHI'07 ACM Human Factors in Computing Systems. pp. 1495-1504.

19. *The limits of expert performance using hierarchic marking menus.* **Kurtenbach, G. and Buxton, W.** 1993. CHI'93 ACM Human Factors in Computing Systems. pp. 482-487.

20. *Visual Similarity of Pen Gestures.* **Long, A., et al.** 2000. CHI'00 ACM Human Factors in Computing Systems. pp. 360-367.

21. **Microsoft.** Pen Flicks, Windows Vista. [Operating System]. 2007.

22. *Hover widgets: using the tracking state to extend the capabilities of pen-operated devices.* **Grossman, T., et al.** 2006. CHI'06. pp. 861-870.

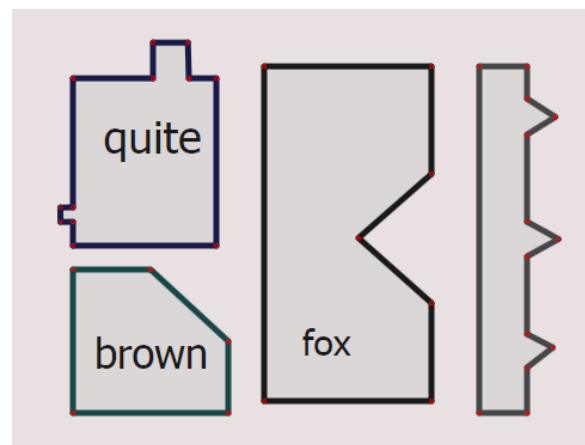
23. *Design and Analysis of Delimiters for Selection-Action Pen Gesture Phrases in Scriboli.* **Hinckley, K., et al.** 2005. CHI'05. pp. 451-460.

## Appendix A: Pilot User Study Tasks

The series of tasks given to pilot study subjects for the two tests are given below.

### First Test Suite

1. Draw a rectangle
2. Draw an ellipse inside the rectangle
3. Zoom in on a corner of the rectangle
4. Zoom out
5. Zoom in using alternate method (in this case, hitch gesture)
6. Label the circle with "abc" typeset text
7. Delete all shapes
8. Draw the following diagram:



9. Select all shapes

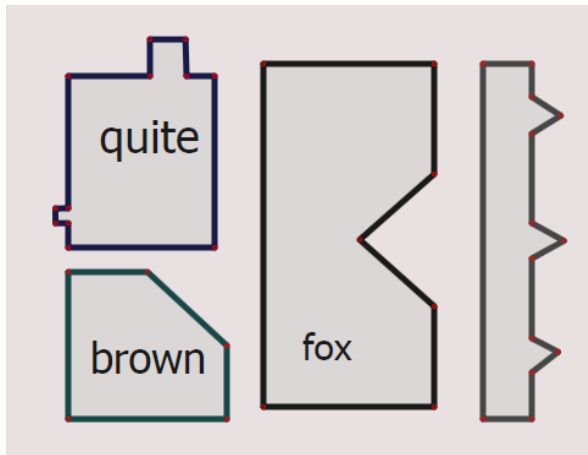
### Second Test Suite

#### Part I

1. Make a rectangle
2. Make an ellipse inside the rectangle
3. Zoom in on a corner of the rectangle
4. Zoom out
5. Zoom in using alternate method (in this case, hitch gesture)
6. Label the circle with “quick” typeset text
7. Delete all shapes
8. Draw the following shape:

1. Draw your family tree using boxes, lines and text. It doesn't have to be accurate – the goal is really to draw a tree diagram.

2. Make a simple house with a door and window, and label the house “House” in underlined text. Don't forget the chimney.



9. Select all shapes
10. Deselect all shapes
- 10.5. Delete all shapes
11. Make three triangles in a row
12. Select the first triangle and the last triangle, but not the middle triangle
13. Now select the middle triangle but not the first or last triangle
14. Delete the first and last triangle
15. Delete the remaining triangle
16. Make a simple house with a door and window, and label the house “House” in italic text
17. Add a chimney to the house
18. Undo the chimney

## Part II