Tableur: Handwritten Spreadsheets

Emanuel Zgraggen
Robert Zeleznik
Philipp Eichmann
Brown University
Providence, RI 02912, USA
ez@cs.brown.edu
bcz@cs.brown.edu
peichmann@cs.brown.edu

CHI'16 Extended Abstracts, May 07-12, 2016, San Jose, CA, USA.
ACM 978-1-4503-4082-3/16/05.
http://dx.doi.org/10.1145/2851581.2892326

Abstract
The need for back-of-the-envelope calculations, such as rough projections or simple budget estimations, occurs frequently and oftentimes while being away from desktop computers. While major software vendors have optimized their spreadsheet applications for mobile environments their generality still makes them heavyweight for such tasks. We have built Tableur a spreadsheet-like pen- & touch-based system targeted towards these use cases. Our design revolves around handwriting recognition - all data is represented as digital ink - and gestural commands. Through a rethought cell referencing system and by incorporating standard math notation recognition Tableur allows for simple formula creation and we experiment with techniques that support pattern-based prefilling of cells (*Smart Fill*) and exploration of what-if scenarios (*Reverse Editing*).

Author Keywords
spreadsheet; pen and touch; handwriting recognition; user interfaces; interaction design

ACM Classification Keywords
H.5.2 [User Interfaces]: Interaction styles

Introduction
Spreadsheet applications are programs that allow users to enter, manipulate and analyze data that is represented in

tabular form and through formulas that reference values in cells. They are widely used for different tasks and support advanced data analysis through built-in scripting languages. The need for simple calculations oftentimes occur while being in situations where a desktop computer is unavailable such as during meeting or while being at lunch with a colleague. Although spreadsheets are extremely powerful their complexity can be overwhelming especially while being in such mobile environments.

In this paper we present Tableur a novel gesture-based pen & touch system that offers spreadsheet-like functionality and is targeted towards back-of-the-envelope calculations on devices found in away-from-desk situations such as phablets, tablets and interactive whiteboards. We designed our system by following the *fluid* design guidelines [4] which promote direct manipulation, minimization of user interface clutter and advocate the integration of interface components directly into the visual representation. In Tableur all data is represented, created and edited as handwritten ink. We offer a set of gestures that manipulate and organize ink and incorporate handwriting recognition such that formulas can be created through standard handwritten math notation. We present a label-based design for referencing content in formulas and discuss techniques that support fast creation of patterns (*Smart Fill*), promote freeform workflows (*Freeform Cells*) and what-if scenarios (*Reverse Editing*).

Related Work

Numerous authors [2, 7, 8, 3, 6, 9] have emphasized the benefits of pen- & touch-based interfaces with regards to data exploration and analysis. Drucker et. al. [3] for example find that users not only subjectively prefer a touch interface over a traditional WIMP interface but also perform better with it for certain data related tasks. Others built pen- & touch-based systems that simplify story telling with data [7], allow users to gesturally create, label, filter and transform charts [2, 8] or enter calculated fields through handwriting [13]. However all of these systems focus on exploring and analyzing existing data and do not support standard spreadsheet functionality like creating and editing new data or referenced-based formulas. In other domains, such as diagram drawing [12] or mathematical eduction [11, 10, 5], researchers have ported the immediacy and fluidity of paper-based pencil-drawing to digital media and augmented it with computational power. We adopted some of these approaches and techniques to the domain of spreadsheet calculations.

Tableur

We motivate our system through a use case throughout which we point to the appropriate sections that describe the techniques in more detail. Figure 1 shows the resulting spreadsheet that is assembled throughout the use case.

Use Case

Eve is the owner of a small business and wants to get a rough idea how her company will be performing over the next few months. She decides to use Tableur to assist her with this task. Tableur offers an unbounded zoom- and panable 2D canvas where ink can be placed anywhere with the use of a digital stylus. Eve starts by writing down an outline of her tabular data with the column headers “Income”, “Expenses” and “Total” and rows labeled “Jan” and “Feb” indicating months she wants to analyze. She then performs a gesture (*Gestural System*) to signal that her ink should be interpreted as a tabular structure (*Ink Segmentation*). After filling out the individual cells with estimates of the respective incomes and expenses she realizes that she wants to expand her spreadsheet beyond just January and February. Another gesture prefills the cells with ink up until June (*Smart Fill*). Eve now wants to create a formula

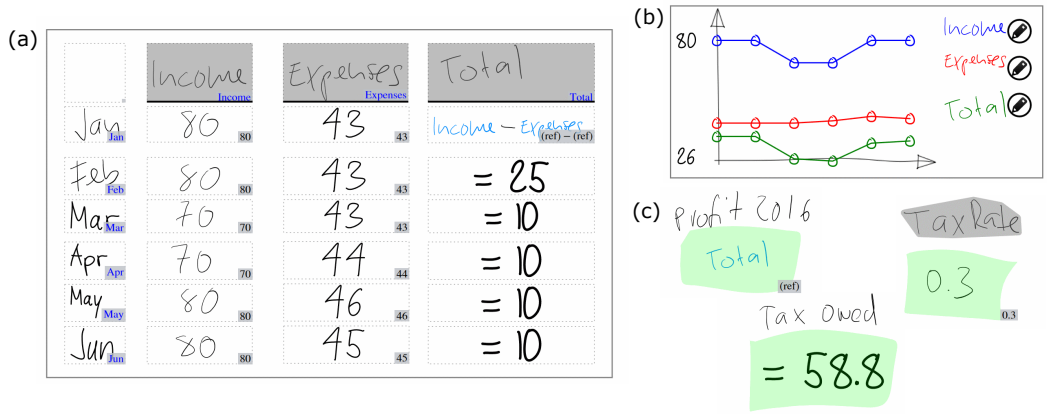

Figure 1: Screenshot of the application depicting the resulting view assembled by Eve throughout the introductory use case. a) Table with rows per month of estimates for income and expenses as well as formulas to compute the total. b) Line chart of "Income", "Expenses" and "Total" column. c) *Freeform Cells* that compute the sum over the "Total" column as well as taxes owed based on a constant "Tax Rate".

that computes the total (income minus expense) for each month. She does so by combining references to the "Income" and "Expenses" labels in the first row's "Total" column through drag and drop and handwrites a minus sign in between (*Formulas*). Propagating this formula to all cells of the "Total" column seems tedious so Eve opts to use another gesture to do so (*Smart Fill*). She fills out the rest of the spreadsheet with estimates for the respective months and then creates a free floating cell next to her table that she labels "Profit 2016" (*Freeform Cell*). She drops the "Total" label into that new cell and the system automatically calculates the sum over that column for her. In order to reinforce the numbers she just wrote down and calculated Eve decided to visualize them. She drags all the column labels out to free area of the canvas and the system shows her a simple line chart. A new supplier that Eve was in contact with offers very competitive prices that would cut down her monthly expenses from roughly $\$60$ to $\$40$. To analyze this what-if scenario she edits the "Expenses" line in the chart by over-drawing a new horizontal line (expenses are more or less constant across months) at roughly the 40 mark (*Reverse Editing*). The system updates the numbers in her spreadsheet as well as her "Profit 2016" cell and she realizes that she could more than double her profit by switching to this new vendor. Eve finalizes her analysis by creating more *Freeform Cells*. She uses those to calculate how much in taxes she estimates to owe for the first half of 2016 based on a "Tax Rate" that she sets to 30%.

Gestural System

All commands within Tableur are triggered through gestures ranging from simple drag and drop operations to ink-based scribble-erase gestures [12] that remove unwanted content. To distinguish regular ink from gestures and to be able to

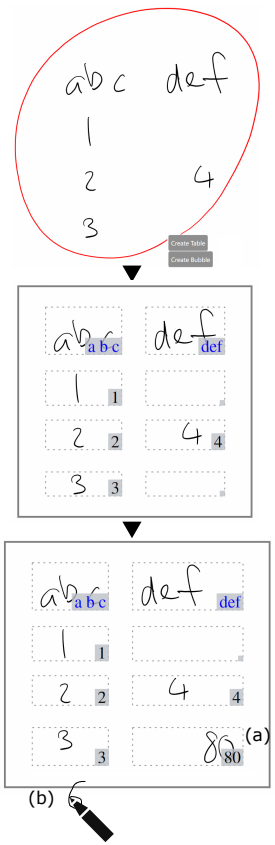

Figure 2: Screenshots from the application. Top: Unsegmented ink outside of an active object with ongoing lasso gesture. Center: Table after ink has been segmented and assigned to corresponding cells. Bottom a) Expanded table to the right after adding ink. b) Ongoing expansion of table to the bottom.

overload gestures with multiple outcomes we implemented a two-step gesturizer system. All new ink gets analyzed and as soon as a potential gesture is recognized it gets highlighted and a pop-up menu is displayed. Selecting an option then triggers the gesture while ignoring it or tapping outside fades the gesture stroke back to normal ink. The top image in figure 2 illustrates this concept. The circular stroke has been recognized as a potential lasso gesture and is highlighted in red. The pop-up menu in the lower right corner informs the user that their gesture has two possible different outcomes. We are planning to incorporate techniques, like the one presented in GestureBar [1], in future versions of our prototype to enhance discoverability of our gesture set.

Ink Segmentation

Ink on the canvas is only interpreted and analyzed if the user chooses to transform it into an active object. Tableur offers two types of such objects: tables and *Freeform Cells*. A lasso gesture around existing ink (or a rectangular gestures for empty objects) is used to create such objects (figure 2 top). The system runs a segmentation algorithm that decides how to break the ink into a row and column structure. The algorithm looks at the distribution of ink among the X and Y axis and outputs a list of bounding boxes that represent individual cells. All ink is then assigned to its corresponding cell and the system invokes handwriting recognition on the content of each cell (figure 2 center). Note that such tables can easily be edited and extended by either writing more ink onto them (figure 2 bottom) or by a set of gestures that allow to correct the automatic segmentation (splitting or merging of cells and columns).

Smart Fill

Standard spreadsheet applications usually offer functionality that helps users avoid doing repetitive and tedious tasks such as propagating formulas to different cells or expanding

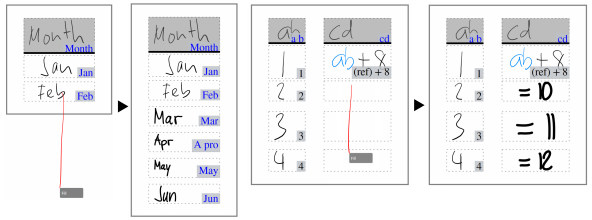

Figure 3: Two examples of *Smart Fill*. Left: Propagation of a date pattern. Right: Propagation of formulas with corresponding updates to row references.

number or date patterns across columns or rows. Tableur incorporates similar functionality. A horizontal or vertical gesture across multiple cells activates this smart filling. By analyzing the cells under the gesture-stroke that contain content the system tries to extrapolate how to fill the remaining cells where the length of the gesture-stroke determines which existing cells need to be filled or how many additional cells need to be added. We currently support simple patterns such as dates (figure 3 left) or ascending and descending numbers as well as propagation of formulas where reference to rows or columns are updated accordingly (figure 3 right). Note that the added synthesized content is just like any other regular ink: it can be manipulate or deleted with the same commands. We sample letters or digits from a prerecorded alphabet that can be customized to match a user's handwriting.

Formulas

Formulas in Tableur live within cells of tables or *Freeform Cells* and consist of handwritten math, references to other cells or combinations thereof. The content of all cells is analyzed automatically as soon as it is inputed by the user. This step involves invoking two separate recognizers: a

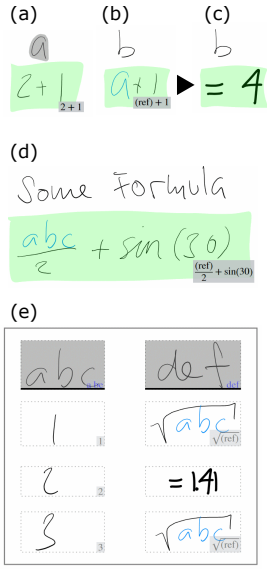

Figure 4: Screenshots from the application showing different examples of formulas. a) Labeled Formula in *Freeform Cell*. b) a) Formula including a reference in *Freeform Cell*. c) a) Formula in *Freeform Cell* showing its result view. d) a) Formula in *Freeform Cell* with more advanced handwritten math notation. e) Formulas in a table.

standard handwriting recognizer to detect textual input and a custom math recognizer. Tableur's math recognition engine is based on prior work developed by colleagues at Brown University [11, 10, 5]. It supports most common math handwriting notations (see figure 4 d for some examples) and offers a variety of customization settings to accommodate for user preferences and different handwriting styles. If the system determines that a cell contains a formula, either with just constants (figure 4 a) or with references to other cells (figure 4 b), the user can toggle between formula view (figure 4 b) and result view (figure 4 c) by tapping on it. Tableur's evaluation engine computes results of formulas by either directly interpreting ink or by following references over possibly multiple hops. Note that in case of cycles in formulas, illegal operations (e.g., division by 0) or invalid inputs (e.g., a reference to text) the system will display appropriate error messages.

Unlike traditional spreadsheet systems that use an abstract notation for cell references in formulas we implemented a label-based approach in Tableur. Any content that a user wants to reference in a formula needs to have a user-defined handwritten label. Users can label *Freeform Cells* through gestures or convert row and column headers to label-handles. These handles (figure 4: ink with gray background) can then be dragged onto other cells to create references. The underlying evaluation engine interprets these references based on the origin of the label and the location of the formula. In tables and when created through label-handles of row or column headers the appropriate cells of the table are referenced (figure 4e: reference to column "abc") while the same reference in a *Freeform cell* gets evaluated as the sum of all values of the corresponding row or column (figure 4d: reference to column "abc"). All formulas are live. Changes to referenced values are directly reflected in the result view of a formula.

Freeform Cell

For many cases creating a full tabular structure feels too heavyweight. Examples include writing down a constant value that will get referenced in other formulas (e.g., π, "Tax Rate") or doing a simple single calculation (e.g., $38 * 140$). For these scenarios Tableur offers *Freeform Cells*, which are created through a simple rectangular gesture, can be placed anywhere on the 2D canvas and contain any kind of formula or value referenced by a label. Their freeform nature can be used to break out of the rigid tabular structure of standard spreadsheets and promote more free flowing working styles where the result of a tabular calculation can be summarized in a single *Freeform Cell* and then reused in other calculations downstream. Figure 1 c shows an example of this where multiple *Freeform Cells* are used to first capture an important result of a table (the sum of the "Total" column in the cell labeled "Profit 2016") and then calculate derivatives of it ("Tax Owed" is based on "Profit 2016" and "Tax Rate").

Reverse Editing

The idea of *Reverse Editing* stems from the notion that it is sometimes easier to sketch the shape of data rather than expressing it through other means. In the realm of data this could include statements like "I want my data to be linearly interpolated between a and b" or "Our income will increase in months with higher temperatures". These statements can be easily illustrated by drawing a straight line between a and b or by sketching a curve that starts low, increases constantly over the summer months and then plateaus again. We support this notion by allowing users to edit tabular data directly by drawing on top of line charts to express how they want their data to look like. Figure 5 shows an example of this technique. In Tableur dragging and dropping row or column label headers onto the 2D canvas creates simple low fidelity line charts of the corresponding data (figure 5 top).

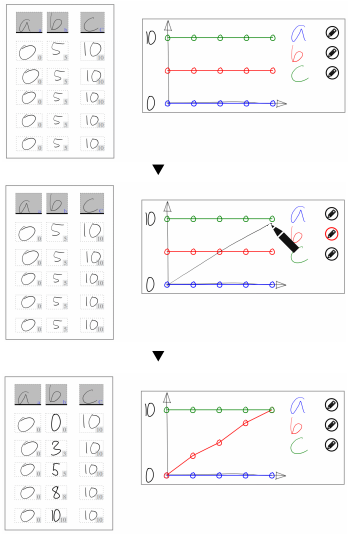

Figure 5: Screenshots from the application showing *Reverse Editing*. Top: Table with corresponding line chart. Center: The user wants the values of "b" to be linearly increasing from 0 to 10. She draws a line directly on top of the line chart that indicates this. Bottom: The system has sampled values along the user-drawn line and replaced the values of the "b" column correspondingly.

By tapping the edit button of a line series users enter the *reverse editing* mode in which lines drawn on top of the chart are interpreted as data-sketches. In the example (figure 5 middle) the user indicates that she wants the values of "b" to be linearly interpolated between 0 and 10. The system samples the ink the users drew at regular intervals and updates the values in column "b" of the underlying table. This sampling strategy supports any arbitrary curve to edit entire series as well as smaller scribbles or circles to move individual datapoints. While such sketches are relatively imprecise they still allow for rapid exploration of different what-if scenarios such as "What happens to our profit if our expenses are slightly higher or lower than expected?".

Implementation Details

The current prototype of Tableur supports all the functionality described in this paper, is implemented in C# and WPF (Windows Presentation Framework) and runs on pen- & touch-enabled Windows 8 or 10 devices.

Discussion and Future Work

We presented Tableur a gestural spreadsheet-like system that is optimized for pen & touch devices. Our system is based on digital ink where users input and edit data through handwriting and organize and annotate ink through a set of gestures. We use handwriting segmentation and recognition to transform digital ink into actionable and computable objects such as tables, constants and formulas. Furthermore we rethink how to fit existing spreadsheet functionality like *Smart Filling* and cell references for formulas into this new environment and describe novel functionality like *Reverse Editing* that exploits the benefits of a gestural and sketch-based application. Limited initial user testing hints at the benefits our approach might have over traditional WIMP interfaces especially on tablets and interactive whiteboards. However, thorough quantitative user studies are needed to analyze the effects in detail. We are currently planning a study where we will compare our prototype to a traditional spreadsheet system on tablet devices through user performance measures for adhoc back-of-the-envelope calculation tasks. Aside from evaluation work, we also intend to extend our system by including more *Smart Filling* patterns, offer more flexibility when creating formulas by supporting functions and labels to individual cells in tables and porting *Reverse Editing* to different chart types.

References

[1] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J LaViola Jr. 2009. GestureBar: improving the approachability of gesture-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2269–2278.

[2] Jeffrey Browne, Bongshin Lee, Sheelagh Carpendale, Nathalie Riche, and Timothy Sherwood. 2011. Data analysis on interactive whiteboards through sketch-based interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, 154–157.

[3] Steven M Drucker, Danyel Fisher, Ramik Sadana, Jessica Herron, and others. 2013. TouchViz: a case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2301–2310.

[4] Niklas Elmqvist, Andrew Vande Moere, Hans-Christian Jetter, Daniel Cernea, Harald Reiterer, and TJ Jankun-Kelly. 2011. Fluid interaction for information visualization. *Information Visualization* (2011), 1473871611413180.

[5] Joseph J LaViola Jr and Robert C Zeleznik. 2007. MathPad 2: a system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2007*

courses. ACM, 46.

[6] Bongshin Lee, Petra Isenberg, Nathalie Henry Riche, and Sheelagh Carpendale. 2012. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *Visualization and Computer Graphics, IEEE Transactions on* 18, 12 (2012), 2689–2698.

[7] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. 2013. SketchStory: Telling more engaging stories with data through freeform sketching. *Visualization and Computer Graphics, IEEE Transactions on* 19, 12 (2013), 2416–2425.

[8] Bongshin Lee, Greg Smith, Nathalie Henry Riche, Amy Karlson, and Sheelagh Carpendale. 2015. SketchInsight: Natural data exploration on interactive whiteboards leveraging pen and touch interaction. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific*. IEEE, 199–206.

[9] Arnab Nandi, Lilong Jiang, and Michael Mandel. 2013. Gestural query specification. *Proceedings of the VLDB Endowment* 7, 4 (2013), 289–300.

[10] Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, and Hsu-Sheng Ko. 2010. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 17–26.

[11] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J Laviola Jr. 2008b. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Smart Graphics*. Springer, 20–32.

[12] Robert C Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. 2008a. Lineogrammer: creating diagrams by drawing. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 161–170.

[13] Emanuel Zgraggen, Robert Zeleznik, and Steven M Drucker. 2014. PanoramicData: Data Analysis through Pen & Touch. *Visualization and Computer Graphics, IEEE Transactions on* 20, 12 (2014), 2112–2121.