Buoys: Implicitly Anchored Sketches in Flowing Text

Adam Kumpf
Luxembourg
kumpf@alum.mit.edu

ABSTRACT

This paper provides a generalized approach for dynamically anchoring sketches within flowing digital text (i.e. line-wrapped typed characters), called buoys. By constraining certain parameters of the writing environment, users can freely move between typing notes and drawing sketches in situ without the need for explicit anchor points to maintain relative positioning.

Keywords

dynamic text anchors; free space annotation; implicit object positioning; constraint-based interaction; virtual notepad; creative tools;

INTRODUCTION

Text editors, email clients, and note-taking applications have become a daily touchpoint for most computer users; each relies heavily on the line-wrapped, or flowing, text paradigm for data entry and digital communication.

Over the past decade, advances in touchscreen and stylus inputs have brought with them a growing set of applications that allow users to annotate (or completely replace) their typed text with freeform drawings [8]. Combining typed text and hand-drawn sketches is a natural progression – the two forms of input complement one another to provide a means of communication that can be fast, precise, and expressive.

CURRENT PRACTICES

Unfortunately, the reality of combining flowing digital text with sketches in a single interface is often tedious. The problem lies in the relative versus absolute positioning of the two paradigms.

This tension forces developers to prioritize one format over the other, or alternatively, break the WYSIWYG interface altogether to include explicit meta-character anchor points to bridge the worlds of relative and absolute positioning [7, 5].

These three common approaches to combining text and drawings can be characterized thusly:

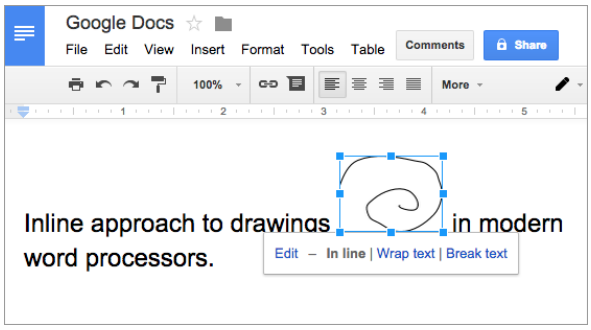

Figure 1. Example of *Inline* placement in Google Docs [3].

1. **Inline**: Force all text and images to flow based on line-wrapping rules. Freeform drawings are often abstracted as image objects or blocks for placement and selection (see Figure 1).

2. **Absolute**: Allow all images and text to be positioned absolutely anywhere on the page. Text may wrap, but often within explicitly sized and positioned text boxes.

3. **Anchored**: Force all text to flow, but allow images to be positioned anywhere via anchored meta-characters (non-printing reference points) that flow with the text.

Each approach contains within it a design decision that balances the priorities of the end user.

If text entry is the dominant mode, the designer is likely to choose an *Inline* approach. If free-form drawing is most common (for example, image editing applications), an *Absolute* approach is appropriate. And in the case where users may frequently switch between text and drawing, an *Anchored* approach can be used; making a wide range of layouts possible, but with the added cognitive overhead of managing meta-characters and positioning as nearby text is changed.

Additionally, other approaches have been examined (such as HybridPointing [2]), but these tend to address very specific use cases where the user is locked into a dedicated mode and thus not frequently switching between drawing and text entry.

BUOYS: ANCHORS WITH INDIRECTION

We propose a new **Buoyed** approach to combining text and drawings that uses dynamic positioning of implicit anchors and explicit line-wrapping characteristics to remove the burden of anchor management.

Instead of anchoring to an absolute location in the text, buoys provide a level of indirection to which an anchor can be attached. Let us break down the *Buoyed* approach into its components.

1. "explicit line-wrapping characteristics": By specifying exactly how lines of text will wrap within a given document (for example, font family, line height, size, kerning, etc.), rendering of each line in a given document must remain consistent across all platforms and can thus be precisely calculated.
2. "dynamic positioning of implicit anchors": When a drawing is added, it is given a reference point implicit to its form (for example, the drawing's center). That reference point can then be placed dynamically to float alongside the text as a function of each line's deterministic nature (explicit line-wrapping).

Buoys allow both flowing text and arbitrarily placed drawings to coexist; spatial relationships are maintained by line without the need for explicit anchors.

Buoy Algorithm

To implement buoys, a fixed-width line-wrapped text area is assumed. This ensures that the corresponding buoy location (which may be a line after it has wrapped) will not change when rendered on various displays.

For each new drawing d added to the text area:

1. Determine the drawing's weighted center c. In the simplest case, this is just the center of the drawing's bounding box.
2. Calculate the corresponding line segment l that appears closest to c.
3. Create a new buoy location b, where:

$$b_x = c_x$$

$$b_y \propto l$$

That is, b_x is an absolute offset, and b_y is relative to the line and will track when preceding lines are shifted up/down.

Because the buoy position b is implicit in what was drawn and where it was placed, there is no need to burden the user by displaying buoy locations or providing buoy management utilities.

Sketchwrite

To demonstrate and explore the real-world possibilities of buoys, we created an online notepad called Sketchwrite [6].

Sketchwrite is a minimalistic sketching and drawing application that lets users freely move between line-wrapped text entry and drawing (with a set of 6 basic drawing line types). There's also basic undo/redo functionality, as well as an eraser to allow the user to remove specific lines by dragging over them.

At first glance, users are not likely to realize that any kind of anchoring is taking place. Without the need to explicitly show drag handles and anchor points, the user interface feels

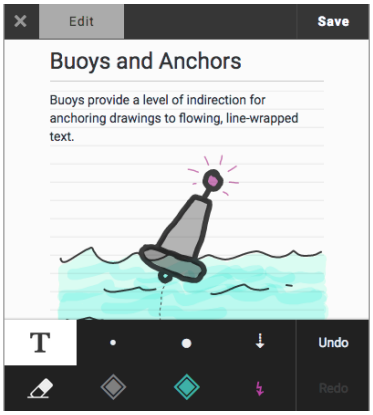

Figure 2. Sketchwrite uses buoys for drawing and text alignment.

natural and the added buoy functionality is unnoticeable. This is particularly true if a user is writing and drawing in a linear process, from the top to the bottom of the page.

But when the user decides to make a change above where he or she previously was working, the impact of the buoys becomes clear: the content below the change is shifted as expected, even though it was originally placed at a particular location on the page with possible relationships to the other text and drawings around it (see Figure 3).

Post-launch feedback about Sketchwrite is ongoing, but preliminary observations suggest that the buoyed approach provides a consistent and intuitive interaction for the majority of users (without the need to explicitly show or manage buoys).

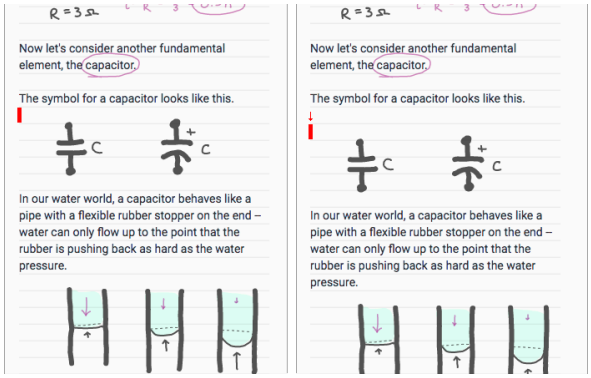

Figure 3. Adding a new line (at cursor shown in red) appropriately shifts sketches and writing downward without explicit anchor points.

Relationships and Callouts

Flowing text and free-form drawings often remain separated on the page, but there are also times when a user may want to circle a particular word or draw lines and arrows between elements to emphasize their relationships to one another (see Figure 4).

When text is inserted on lines above or below the drawing, the circled word will remain properly synchronized and attached as originally drawn. Similarly, if lines above or below the drawing are removed, the text and drawing will also remain aligned.

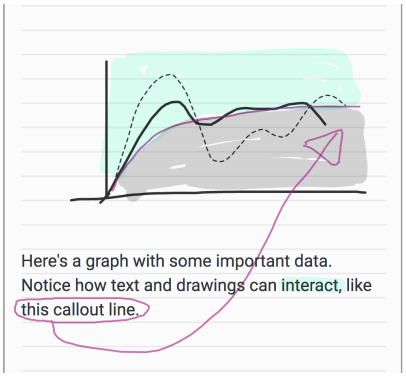

Figure 4. Interaction between flowing text and free-form drawing. Modifying lines above or below the overlapping region does not affect their relative alignment.

Managing Horizontal Misalignment

In the edge case where a user may have circled a word, and then gone back to add or remove characters preceding the drawing on the same line, the two may be misaligned; this ambiguity arises because text may shift horizontally within a line without altering the overlaid drawing.

While explicit anchors could allow a user to realign a word-shifted drawing, the cognitive load of managing the secondary representation would likely outweigh the alternative of simply erasing and re-drawing a circle around the word [9].

Furthermore, in the case where a line connects a larger drawing to a particular word, shifting the entire drawing to match the placement of the word is unlikely to be the desired behavior. Instead, simply re-drawing the connecting line appears to be the most straightforward solution to this rare occurrence.

NEXT STEPS

Beyond note-taking and word processing, the benefits of combining text with drawings could have even greater impact in domains where expert knowledge is documented and transferred. One such domain is software, where large codebases are shared among many developers over a long period of time.

Programmers frequently encounter complex segments of code that rely on ASCII diagrams (text-based drawings) or external links to explain how a process or algorithm functions. We believe that a buoyed approach could enable integrated ways of combing ad-hoc images with any plain text document; embedding related visual diagrams, graphical notes, or any other media alongside code and comments.

By allowing code and diagrams to speak the same language, expert knowledge could be shared more quickly and accurately between users over time, while allowing for the underlying, machine-parseable code to remain unchanged. Similar applications could also be possible in fields such as chemistry (documenting lab work), medicine (visually depicting symptoms), music (abstract chord patterns), literature (diagramming story relationships), and cooking (illustrating preparation methods) [1, 10, 4].

CONCLUSION

We have shown that adding *buoys*, an implicit and indirect form of anchors, to line-wrapped text documents can provide an intuitive and simple way of managing the relative placement of sketches and annotations. Through the development and use of Sketchwrite, we have gathered preliminary feedback and observations that indicate the effectiveness of buoys and their ability to be implemented. We have also described a potential shortcoming when horizontal relationships within a line are modified. This limitation, as well as adding buoys to other application domains, are areas we suggest for further research and development.

ACKNOWLEDGMENTS

Many thanks to Matt Wolfe for his input concerning text editors and embedded media positioning. This work was made possible by the generosity of Maura Atwater for her continued support of applied research in the creative arts.

AUTHORS

Adam Kumpf has two Masters degrees from MIT in Electrical Engineering & Computer Science (compliant robotics) and Media Arts & Sciences (tangible user interfaces). Adam has worked at LEES, Motorola, CSAIL, MIT Media Lab, Teague, The Chaos Collective, and Fiddlewax.
`http://projects.kumpf.cc`

REFERENCES

1. Bergstrom, T., Karahalios, K., and Hart, J. C. Isochords: visualizing structure in music. In *Proceedings of Graphics Interface 2007*, ACM (2007), 297–304.
2. Forlines, C., Vogel, D., and Balakrishnan, R. Hybridpointing: Fluid switching between absolute and relative pointing with a direct input device. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, ACM (New York, NY, USA, 2006), 211–220.
3. Google Inc. Google Docs, 2015. `https://docs.google.com`.
4. Horng, J.-S., and Hu, M.-L. The mystery in the kitchen: Culinary creativity. *Creativity Research Journal 20*, 2 (2008), 221–230.
5. Javed, A., and Schwenk, J. Systematically breaking online wysiwyg editors. In *Information Security Applications*. Springer, 2014, 122–133.
6. Kumpf, A. Sketchwrite, 2015. `https://sketchwrite.com`.
7. Myers, B. A Brief History of Human Computer Interaction Technology. *ACM interactions 5*, 2 (March 1998), 44–54.
8. Shneidemnan, B. Touch screens now offer compelling uses. *IEEE Software 9*, 3.
9. Sweller, J. Cognitive load theory. *The psychology of learning and motivation: Cognition in education 55* (2011), 37–76.
10. Whitin, P. *Sketching Stories, Stretching Minds: Responding Visually to Literature.* ERIC, 1996.