LiquidText: A Flexible, Multitouch Environment to Support Active Reading

Craig S. Tashman
GVU Center, Georgia Institute of Technology
85 5th St., Atlanta, GA 30308 USA
craig@cc.gatech.edu

W. Keith Edwards
GVU Center, Georgia Institute of Technology
85 5th St., Atlanta, GA 30308 USA
keith@cc.gatech.edu

ABSTRACT
Active reading, involving acts such as highlighting, writing notes, etc., is an important part of knowledge workers' activities. Most computer-based active reading support seeks to replicate the affordances of paper, but paper has limitations, being in many ways inflexible. In this paper we introduce LiquidText, a computer-based active reading system that takes a fundamentally different approach, offering a flexible, fluid document representation built on multitouch input, with a range of interaction techniques designed to facilitate the activities of active reading. We report here on our design for LiquidText, its interactions and gesture vocabulary, and our design process, including formative user evaluations which helped shape the final system.

Author Keywords
Active reading, multitouch input, visualization.

ACM Classification Keywords
H5.2 Information Interfaces and Presentation: User Interfaces – Interaction Styles. H5.2 Information Interfaces and Presentation: Miscellaneous.

General Terms
Design, Human Factors.

INTRODUCTION

From magazines to novels, reading forms a critical part of our lives. And beyond the relatively passive reading of a blog or newspaper, many reading tasks involve a richer interaction with the text. This interaction, known as *active reading* (AR) includes processes such as highlighting and annotating, outlining and note-taking, comparing pages and searching [13].

AR is a regularly occurring component of knowledge work. For example, prior studies have shown that two specific AR-related activities, reading to answer questions and reading to integrate information [15], each constituted about 25% of the time knowledge workers spend reading [1].

Generally, AR demands more of the reading medium than simply advancing pages—active readers may need to create and find a variety of highlights and annotations, flip rapidly among multiple sections of a text, and so forth. Many of these activities are well supported by paper, but historically, computers have proven inadequate [1, 16, 17]. Recently, however, advances in hardware and software have enabled computers to achieve many of the qualities of paper, such as support for freeform annotation and navigation using the non-dominant hand. Thus, studies suggest that recent pen-based tablet PCs match or surpass paper as an AR medium [13]. In effect, computers are getting better at supporting AR by building upon an increasingly paper-like experience.

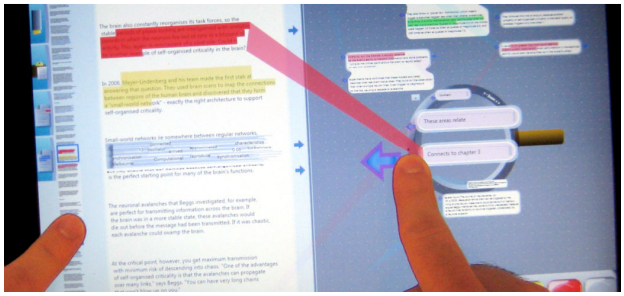

Figure 1. LiquidText running on tablet PC.

However, while current systems may have succeeded in mimicking key affordances of paper, paper itself is not the *non plus ultra* of active reading. O'Hara's description of the actions that occur in the AR process highlights the many weaknesses of paper; while paper supports some things well—such as freeform annotation and bimanual interaction—it lacks the flexibility for others [15]. For example, viewing disparate parts of a document in parallel, such as for comparison or synthesis, can be difficult in a paper document such as a book. Even annotation—traditionally seen as a strong point for paper—can be constraining, complicating the creation of large annotations, or marginalia that refer to disparate or large portions of text, or to multiple texts. And though the tangibility of paper does support some rapid forms of navigation, the linear nature of most paper texts gives the reader little flexibility for creating their own navigational structures.

All this is not to say paper is bad; however, purely mimicking the affordances of paper in a computer-based system may not address all of the opportunities for improvements that digital technology can potentially provide. Particularly, a more flexible fundamental approach to document representation may provide opportunities to better support AR.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

This paper presents *LiquidText*, an AR system we developed, which explores a substantially different approach to representing and interacting with documents. In contrast to the model embodied by paper, which offers a stable but rigid representation, we focus on giving the reader a highly flexible, malleable kind of document. Specifically, LiquidText provides rich, high degree-of-freedom ways to manipulate the presentation of content, control what content is displayed and where, create annotations and other structures on top of content, and navigate through content. To do this, LiquidText provides a multitouch gesture-based user interface running on modern multitouch displays.

The key contributions of this work are: 1) its collection of multitouch interactions, and specific gestures, for manipulating text documents; 2) the coherent integration of these interactions into an application where they can be refined, evaluated, and built upon; and 3) the underlying paradigm of more flexible document representations. We also present the design processes, varying ideas, and tradeoffs involved in developing LiquidText.

In the following sections, we consider areas of related work, our design process, and the LiquidText system, followed by discussion and reflections on the project.

RELATED WORK

In this section, we review prior AR-related research from three perspectives: 1) we consider investigations into the AR process itself, and how technology might support it; 2) we discuss past projects created specifically to support the AR process; and 3) we consider systems that, while not targeting AR *per se*, focus on some of the processes that occur in AR.

Active Reading Background

Broadly, earlier work has shown that AR involves four core processes: annotation, content extraction, navigation, and layout [15]. Here, we briefly discuss each process and note some of the requirements for supporting it. As this research informed our design goals for LiquidText, we structure our later presentation of the system in terms of these categories.

Annotation. Annotation involves embellishing the original text, as in highlighting, marginalia, etc. This process requires high efficiency, which can be lost when switching between tools used for different types of annotation [15, 16]. It also requires support for idiosyncratic markings, and effective ways to retrieve past annotations [16, 20].

Content extraction. Content extraction involves the copying of original content to a secondary surface, as when outlining or note taking. To maintain efficiency, this must be closely integrated with the reading process, as there is a significant potential for interference between the two. Users also must easily be able to organize, review, and trace their extracted content back to its source [15, 17].

Navigation. Navigation involves moving both within and between documents, as when searching for text, turning a page, or flipping between locations to compare parts of a text. This requires high efficiency, even for traversing large amounts of text, and benefits from metadata cues or landmarks to help the reader maintain orientation. It should also support reader-created bookmarks, and allow the reader to perform other tasks in parallel with navigating [2, 16, 22].

Layout. Layout is the process of visually, spatially arranging one's documents and materials, as when cross-referencing, comparing materials side by side, or gaining an overview of a document. It requires support for viewing different pieces of content in parallel, while maintaining the document's original linear, sequential structure [13, 15].

Prior Systems Supporting Active Reading

Of the systems designed to support AR as a whole, two of the most complete are XLibris [19, 21] and PapierCraft [11]. XLibris is a stylus-based system that augments an explicitly, extremely paper-like metaphor with digital functionality. PapierCraft goes further in offering a paper-like experience, actually letting users control digital documents through marks on physical paper. Building atop a paper-like experience has benefits, including familiarity [12], but can bring some of the limitations associated with paper as well. As examples, PapierCraft was still subject to a lack of margin space, and challenges creating annotations that spanned multiple documents; XLibris users noted an absence of flexibility that arose from the explicitly paper-like metaphor followed by the system, including the inability to alter the linear presentation of text, or to spatially alter pages to construct different, parallel representations of content. So while the system was comfortable to use, its advantages over paper were sometimes unclear, as one user stated explicitly [12].

By contrast, our goal with LiquidText was to remove some of the rigid constraints imposed by a too-literal adoption of a paper-like metaphor; in contrast with these earlier systems, we sought to explore the possibilities afforded by more flexible, decidedly *non-paper-like* representations.

Other Relevant Systems

In contrast to the above, other systems have addressed certain AR-related tasks without seeking to support the entire active reading process. Such work ranges from helping authors provide new types of navigational affordances for their readers [14] to dynamic layouts that provide detail on demand [3]. Still other systems have explored novel approaches to text visualization both for documents [9] and for source code [10]. While these systems are not specifically targeting AR, they do suggest various interactions and manipulations that may be relevant to AR nonetheless.

DESIGN APPROACH

Our goal with LiquidText was to create a design that overcomes some of the innate difficulties inherent in paper-like models. In particular, our specific goals were to support 1) extensive, flexible, direct manipulation control of the visual arrangement of content, including both original material as

well as annotations, and 2) flexibility in navigating through the content, with a rich array of navigational affordances that could be tailored and organized as required. To do this, we undertook an iterative design process, through which we created the final version of the system described here. But first, we describe our design process in more detail.

Initial design ideation. We began by creating scenarios and accompanying requirements for AR, grounded both in our own experiences as well as the prior literature. We also began exploring alternative metaphors for text representation—ones that would promote a user experience that felt fluid and flexible; not only allowing, but *inviting* users to step outside the bounds of predefined structure that computers often impose.

From a design perspective, to construct these metaphors, we found it helpful to explore forms and substances that exemplify malleability, such as putty or water. And while considering these ways of manipulating materials, we improvised interacting with imaginary systems inspired by such substances. Throughout this design process, we sought to brainstorm in the context of plausible scenarios, in order to lead our thinking toward designing a complete, integrated system, rather than just a collection of standalone interactions.

Two guiding criteria for this phase of the design process were 1) to seek to include in our designs features that would support all of the major aspects of AR, and 2), to explicitly focus on supporting those processes where paper-like representations fall short. Likewise, to maintain a manageable scope, we focused on supporting the core processes of AR: the interaction with source texts and the creation of notes, as opposed to a larger AR workflow including creating wholly new documents, sharing documents, and so on.

Prototyping. After completing an initial system design, we sought feedback from other designers using a simple medium-fidelity prototype containing a limited set of interactions (like fisheye text zooming, and creating and grouping notes). This led to several semi-formal critiques where other HCI professionals offered feedback on our designs and prototype. This provided us with feature suggestions and design criticism, and helped to inform the required feature set of the project.

In developing this initial prototype, we also chose to focus on single-document scenarios, as the e-reading appliances with which LiquidText would most likely be used are relatively small, tablet-sized units, and these simpler scenarios seemed a more tractable design goal. Adler et al. likewise show this is not uncommon; of time spent reading/writing with multiple documents, about 40% had only one document being read (i.e., the other document(s) were writing surfaces) [1].

Formative study. Through continued design and development, we completed a high-fidelity prototype system embodying most of the essential functionality we envisioned from the previous design stages, and which was sufficiently refined to be used in actual active reading tasks. We evaluated this functioning prototype with users, to get feedback on our approach and to provide direction for subsequent steps. This evaluation consisted of a multi-part, 18-person study. Our objectives were: 1) evaluate the prototype's particular interactions, 2) evaluate the prototype as an integrated AR environment, and 3) evaluate the acceptability of multitouch by users in a knowledge-work activity.

The study began with a weeklong journaling task, in which participants recorded each time they performed an AR task, answering questions about their goals, difficulties, etc. for each. This helped us obtain a baseline view of our participants' current AR practices. Participants were then interviewed individually about their general AR behavior, followed by fifteen minutes of training on using the prototype system, which ran on a 12.1" Dell XT2 multitouch tablet. Participants also received a help sheet showing how to perform seven of the main functions in the prototype.

Participants then performed a 25-minute AR task, where each read a 5-page popular science article using the prototype and wrote a detailed critique.

Afterward, we performed a semi-structured debriefing interview, asking participants about various aspects of their reactions to the system.

Finally, we concluded this phase of the design process with two separate, three hour participatory design workshops, where 14 participants (two of whom did not participate in the above parts of the study, and are not included in the 18) brainstormed and mocked-up future AR environments.

We recruited broadly for the study, seeking regular active readers in a large design and manufacturing firm. The 18 participants (12 women) included managers, designers, lawyers, students and strategists. Of these, 16 returned their journals and were found to perform an average of 6.6 AR tasks over the roughly weeklong journaling period. While the complete results of this study are beyond our present scope (see [23]), we describe below how our results informed the final design and functionality of the system.

Redesign. Using participants' reactions to the prototype, in conjunction with their thoughts and ideas about AR generally, we identified several potential refinements, which we integrated into our redesigned system. We focused on areas that were especially important to users, but also relevant to our larger design objectives of exploring more flexible representations of text documents. The section below describes the final LiquidText system and its features.

LIQUIDTEXT SYSTEM

Having discussed the design process we followed, in this section we consider the final form of the LiquidText system and the rationale behind many of its functions.

Multitouch Input Model

The use of multitouch in LiquidText stems from two requirements raised by our design approach: first, because we

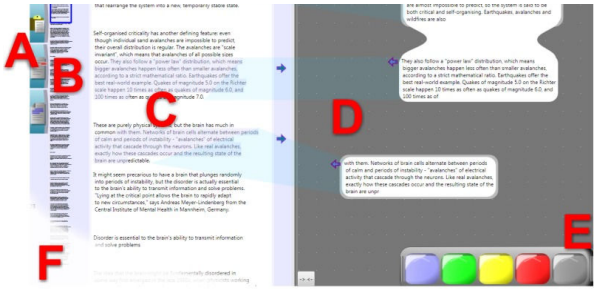

Figure 2. Overview of LiquidText screen: A) button bar, B) preview pane, C) main document, D) workspace, E) color palette, F) Dog-ear area.

were trying to let the user control visual/spatial properties, and do so using interactions based on physical/spatial metaphors, we needed an input device with a natural spatial mapping. Second, past literature reveals that readers rely on bimanual control while reading with paper, such as when arranging content, navigating, etc. [13, 16]. The interaction gestures that resulted from our design phase similarly reflected this style of rich, parallel input, and so lent themselves well to multitouch interaction.

Naturally, other configurations of devices could potentially satisfy these criteria—such as multiple mice. In comparison, multitouch has advantages and disadvantages. For example, multitouch is faster than multiple mice for many bimanual, and even some unimanual, tasks, such as selection [4]. But by contrast, multitouch also tends to incur far higher error rates than mice [4], although improved touch sensing and modeling may remediate this significantly in the future [7]. Additionally, a mouse reduces each hand's input to a roughly single point, reducing the richness of the interactions each hand can perform. A potential compromise then could come from multitouch-plus-pen interaction [6], allowing richer, multi-finger input from one hand and simpler but more precise input from the other [8]. But while we wanted both of the user's hands free to engage in multi-point interaction, we ultimately opted against pen-plus-touch because we were unaware of an adequate hardware platform that could process pen and multitouch input *simultaneously*—which would be needed for bimanual interaction. Therefore, we chose pure multitouch input for LiquidText.

Overview of the System

This section provides a high-level overview of the basic functionality of LiquidText; subsequent sections provide detailed explanations of the system and its features.

LiquidText starts by presenting the user with two panes (Figure 2): The left contains the main document the user has loaded for reading (C); a range of interactions (described shortly) allow the document to be panned, scaled, highlighted, partially elided, and so on. The large *workspace* area (D) provides space for any comments or document excerpts the user may have created; here, they can be grouped and arranged, and the workspace itself can be panned or zoomed. Users can also create multitouch fisheye lenses to zoom only certain areas. Finally, users may also create custom navigation links among all the various forms of content in the system, such as from the main document to comments or excerpts in the workspace.

Some of the basic interactions in LiquidText reuse a number of common gestures that appear in other touch applications (e.g., [24]). For example, the user can position objects, such as comments, excerpts, and documents, just by dragging with a fingertip. The user can rescale an object by using two fingers to *horizontally* pinch or stretch it, or she may rotate it by twisting with three fingers. The preview pane next to the document (Figure 2, B) provides a scaled-down view of the document, and the user may simply touch it to navigate the document to the corresponding point.

In the following sections we discuss the details and rationale for the central aspects of the design, organized according to the major AR processes: content layout and navigation, content extraction, and annotation.

Content Layout and Navigation

LiquidText provides a number of novel features intended to support the AR processes of content layout and navigation. These features allow users to access an existing text *linearly* in its original form (as shown in the main document view in Figure 2), yet leverage a range of interactions to selectively view multiple regions of text, and create custom navigational structures through the text. The major LiquidText features for content layout and navigation are Collapsing, Dog-Earing, and Fisheye Views in the workspace area.

Collapsing. Among the most important—yet problematic—aspects of layout in AR is the need for parallelism, such as viewing multiple pieces of a document at once [13, 15]. But as our intuition suggested, and our study participants told us, viewing disparate areas of a paper document in parallel is often difficult, requiring frequent flipping back and forth. To better support this in LiquidText, we were motivated by elastic substances which can be selectively compressed or expanded—suggesting a document that lets one compress or shrink some areas to bring text from disparate areas of a document together—resulting in a sort of 1-dimensional fisheye. The visualization is thus similar to [9], and the interaction similar to [25], but in a very different context.

The visual representation used in collapsing reduces selected rows of text to horizontal lines, thereby indicating the amount of text hidden. Additionally, multiple regions of text may be collapsed at once, letting the user choose precisely which portions of the document are visible in parallel. Interactions involving this effect are used throughout LiquidText.

Since this "collapsing" process is applied to a vertical, linear document (Figure 2), LiquidText uses a vertical pinch gesture to provide a natural mapping for this interaction (Figure 3). This gesture offers degrees of freedom for controlling the amount of text collapsed (through how far the user moves her fingers), and whether it elides text above and/or below the

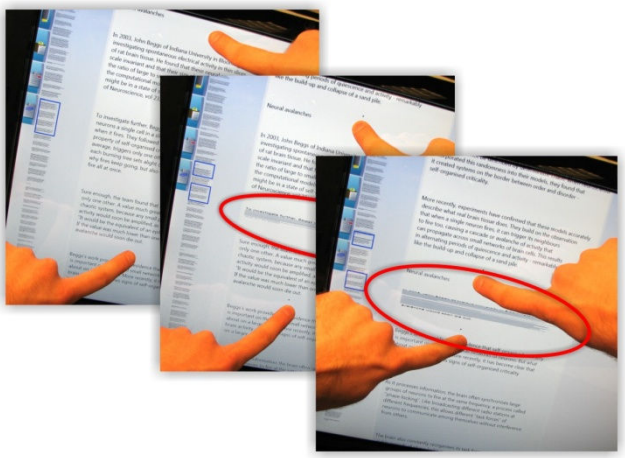

Figure 3. Three stages of an increasing amount of text being collapsed together.

user's fingers (through whether the user moves the top finger to the bottom, vice versa, or both).

This interaction, however, raises the design question of what should happen when the user scrolls a partially collapsed document. For example, if the user scrolls text above a collapsed area, either 1) the entire document could scroll, including the collapsed area. Or 2), only the text above the collapsed area could scroll, adding text to, or removing it from, the collapsed area. Under the first option, the collapse is *a part* of the document itself, and can be used to conceal irrelevant material. Under the second option, the reader effectively has two windows into the document, each of which can be scrolled independently. Both choices have advantages, but we selected the first option as it better fit our metaphor of pinching/stretching the document.

By contrast, the latter option is similar to systems like Adobe Reader, which show disparate regions of a text by allowing the user to create multiple, independent views into the document via a split pane. But that approach to multiple, independent views comes at a cost, offering little clue as to the relative order and distance between the document regions; this disruption to the document's underlying linearity may also interfere with a user's sense of orientation [15]. Collapsing, through either approach though, guarantees that visible regions are always in the order of the original document, and provides a visual cue as to how much text has been collapsed between two visible regions as well.

While this basic collapsing interaction provides a useful way to hide irrelevant text, and bring disparate document regions into proximity, manually pinching a long document together to view, say, the first and last lines of a book, is tedious. LiquidText thus provides two indirect ways to collapse a text: first, touching the preview pane with two or more fingers causes the document to automatically collapse to show all the indicated areas at once. Alternately, holding down a desired part of the document with a first finger—effectively holding it in place—while using a second to touch the preview pane,

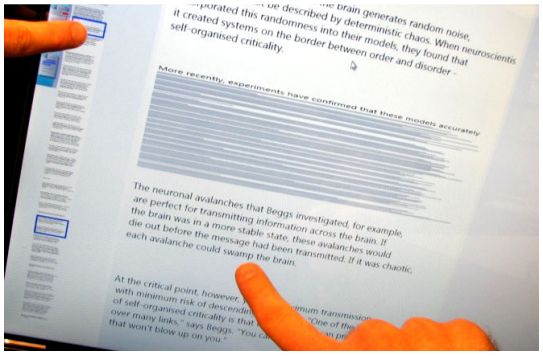

Figure 4. Holding one finger on document while scrolling another finger on preview pane collapses document to show both locations.

causes the document to collapse so as to keep the area under the first finger visible, while also showing the area indicated by the second finger (Figure 4). And in contrast to traditional document viewing software, in which users must create separate panes and scroll them individually, this functionality lets a user view two or more document areas with just one action, parallelizing an otherwise serial task.

We created the collapse interaction before our formative study, so we evaluated it in the study and found user feedback to generally be positive. Notably though, two participants used collapsing as a way to conceal irrelevant material, even though we only presented it as a way to bring disparate material together—suggesting our decision to make collapses part of the documents themselves was of value to users.

Dog-earing. Bookmarks meant for short-term, transient use are an important part of AR navigation [2]—as when holding a thumb at a particular page while browsing a book. Such "ephemeral" ways of book-marking are often used when rapidly flipping between regions of a text. In addition to offering several options for persistent book-marking (described below), LiquidText includes a type of transient bookmark we call dog-earing. This feature was focused on scenarios where the user would create and refer to a bookmark over a short period of time, and is intended to support near effortless creation and deletion of the bookmarks.

In LiquidText, this interaction is modeled on its paper counterpart: users associate a finger with a given document location, much like putting a thumb on a book page allows a user to rapidly flip between places in the text. To create one of these bookmarks, the user simply puts down a finger in the dog-ear region of the LiquidText application (see Figure 2, F), which creates a small orb under her finger corresponding to the current state of the document (including it is positioned, whether it is collapsed, etc.). The user can then navigate as she wishes using the other hand, while keeping her finger on the orb to "save her place." To return to the captured state, the user simply lifts the finger off the orb and LiquidText returns to that state while the orb fades out. Taking advantage of multitouch input, the user may do this with more than one finger at a time to capture and flip between

several layout states. To discard an orb without using it, the user drags it away from the dog-ear region before releasing it.

The choice of gesture here enforces transience, in effect: the bookmark will be kept for only as long as the user's finger is in place (just as with a page in a book). Bookmarks held in this way vanish as soon as the finger is lifted, meaning that the user is freed from having to explicitly delete the bookmark. (Other forms of book-marking, described shortly, can be used when a user wishes to create a more permanent placeholder in a text.)

Though we developed dog-earing before the formative study, we found our tablets could not reliably detect the gesture, and so we disabled it in the build given to users.

Fisheye workspace. As noted, the workspace area to the right of the document allows users to freely arrange comments and excerpts (described shortly). These can be rescaled to see them more closely, or to make more space as needed.

During the study, however, users commented on the dearth of workspace available to them on the 12.1" tablet which ran the prototype version of our system. Likewise, our participatory design workshops indicated that users required a large space in which to see all their documents at once—and to get an overview of any cross-document relationships. In light of this, we explored a set of new features for the final version of our system, designed to overcome the limitations of simply scaling individual objects.

We considered several alternatives for letting users work with larger numbers of objects effectively. Obviously, a physically larger display would be helpful, but this trades off against portability (something our participants strongly required as they reported often shifting between various public and private working areas). We also considered providing some form of organizational structures, such as allowing users to make hierarchical outlines of comments and excerpts in which tiers can be shown or hidden as needed. Outlines have certain downsides, however: they impose a strict organizational structure that may be inflexible, and they also privilege hierarchical over spatial relationships.

Ultimately, for the final system, we settled on the notion of a quasi-infinite, continuous workspace for comments and excerpts, extending beyond the display. This workspace can be panned and zoomed, thus supporting spatial overviews of comments and excerpts, and maintaining consistent spatial relationships among the objects within the space. Since simultaneous viewing of multiple pieces of content is important in AR, we considered a number of approaches to supporting this functionality in the workspace region. For example, one might allow regions of the workspace to be collapsed, to bring distant objects into proximity even when zoomed in. Although this idea is appealing—especially since the collapsing concept is already present in LiquidText—the technique was most suited to one-dimensional spaces. Collapsing in a large 2D space would hide a great deal of content unnecessarily and, if vertical and horizontal collapses were used at once, could easily become confusing.

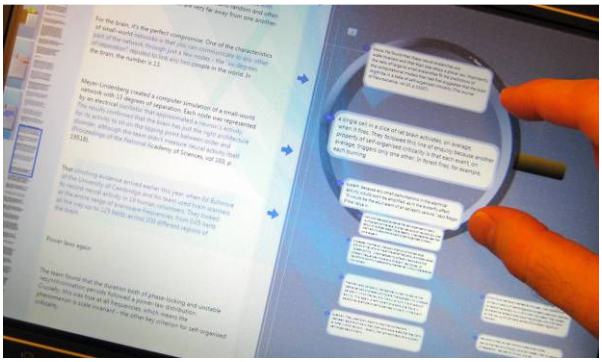

Figure 5. Multitouch fisheye lens function.

To avoid these downsides, we created a novel type of fisheye effect; users access this functionality through one or more fisheye "lenses" they create in their workspace (Figure 5). But in contrast to typical fisheye effects, all text in a given object is scaled to the same level in order to promote readability (as in Figure 5).

In the final version of our system, the workspace is panned or zoomed using typical drag/pinch gestures. Fisheye lenses are created by tapping with three fingers; once created, they can be moved by dragging and resized by pinching or stretching the lens border. The magnification level is controlled by rotating the lens—akin to the zoom lens of a camera. Thus, the user can control all the essential properties of the distortion in tandem and, using multiple hands, for two lenses at once. Thus, while fisheye distortions have been deeply explored, our approach combines two unique features: consistent scaling within each object, and the ability to create and manipulate the distortion's attributes in parallel through multitouch.

Content Extraction

As one of the central processes of AR, extracting textual excerpts from documents serves to support aggregation and integration of content from across one's documents. It lets the reader maintain peripheral awareness of key material, and explore alternate organizations of an author's content [12, 15, 17]. But, in most present approaches to AR, content extraction has been quite heavyweight: paper-based active reading may require copying, scanning, or rewriting portions of content in a notebook; even computer-based AR may require one to open a new document, copy and paste text into it, and save it. We sought to create a fast, tightly-integrated set of mechanisms for content extraction in LiquidText that would not distract from or interfere with the AR process itself [15].

To devise an intuitively appealing, flexible interaction to support content extraction we again sought to draw on physical metaphors. Imagining a document as puttylike, we conceived of extracting content as analogous to *pulling it off* of the document. The two major parts of this interaction are, first, selecting the text to be excerpted, and, second, actually transforming that selected text into an excerpt.

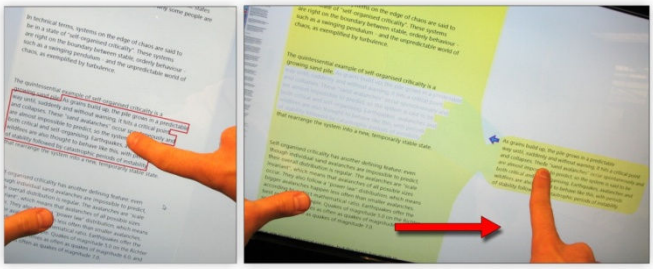

Figure 6. A) Holding document with thumb while putting finger on selection. B) Dragging selection as indicated by arrow to create excerpt.

Text selection. Creating an excerpt begins with selecting the desired text. In earlier versions of our system, we offered two ways to do this: 1) an easy but modal approach in which the user held down a soft button with one hand while dragging a finger over the desired text, and 2) a non-modal approach, where the user puts her thumb and forefinger together just below the start point of the selection, and moves her fingers to the endpoint. This way, the user did not have to shift her gaze away from the document to invoke a selection mode.

Our formative study revealed disadvantages to both approaches. Several users disliked the bimanual requirement of the modal approach, and the non-modal approach was difficult to perform reliably. By contrast, users indicated that they wanted to select text by simply dragging a finger—the same gesture they preferred to use to move objects. Thus, for the final system, we sought a gesture distinguishable from, but very similar to, simple dragging.

The result, replacing our initial non-modal interaction, was a simple tap-and-drag—also used in some iOS apps. That is, the user puts down a finger at the start of the intended selection, lifts her finger and then lowers it again, then drags her finger to the desired endpoint. To give the user prompt feedback, a selection cursor is displayed as soon as the finger touches the screen.

Excerpt creation. Once a span of text has been selected, users may excerpt it from the document. Following the putty metaphor, the user creates an excerpt simply by anchoring the document in place with one finger, while using another to drag the selection into the workspace until it snaps off of the document (Figure 6). The original content remains in the document, although it is tinted slightly to indicate that an excerpt has been made of it.

Excerpt manipulation. After creating excerpts, users must be able to organize and review them [15, 17]. To support this and our general design requirement of flexible content arrangement, excerpts can be freely laid out in the workspace area; they can also be attached to one another (or to documents) to form *groups*. This allows users to create whatever structure they may desire on their excerpts, rather than the system imposing explicit relationships that may not be necessary or may not fit with users' models of the structure.

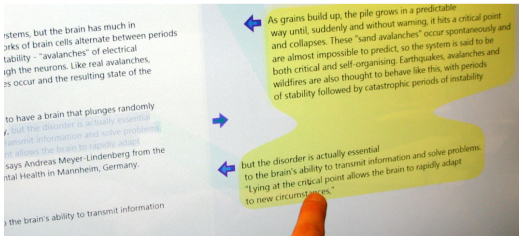

Figure 7. Attaching two excerpts to form a group.

In designing the ultimate appearance and behavior of these groups of excerpts, though, we faced a tension between the advantages of structure versus flexibility. In one approach we considered, grouped excerpts move into a container-object, where they are aligned and sequentially listed. This approach is visually simple and organized, especially for a small screen. Alternately, we considered letting grouped objects be positioned arbitrarily and freely within a group, and have visual links rendered between them to show they are grouped. This option gives users more means of expression, letting them indicate differences in group constituents by position/size/angle, but would likely be messier. We felt however, that the fisheye workspace would adequately accommodate any potential disorganization, and so chose this latter option, informing the interaction's behavior with the putty metaphor used in excerpt creation.

Excerpts are grouped simply by dragging them together—which creates a fluid-like border surrounding all objects in the group (Figure 7). Pulling them apart stretches the border until it eventually snaps, separating the group.

Excerpt context. The final aspect of excerpts we discuss here relates to using excerpts as a means to return to the excerpted text's original context in the document. While excerpts alone can be helpful, being able to trace them back to their source or context is known to be vital for AR [17]. We explored two alternatives to providing such context: the first was *in-place expansion*, where the original context of an excerpt was made to appear around the excerpt, showing the context without disturbing the original document. The second was *linking*, where users can interact with an excerpt to cause the source document to position itself so the excerpt's context can be seen.

Each has advantages: in-place expansion lets the user keep her gaze in the vicinity of the excerpt and avoids disturbing the position of the original document. But if the user needs to do more than just glance at the source context, she will likely need to navigate the original document to that point anyway—which linking does immediately. Likewise, an in-place expansion on a small screen may well cover much of the user's workspace, raising visibility problems, especially if one wanted to see the contexts of two excerpts at once.

For this and other reasons we ultimately chose linking, which we implemented bi-directionally, so excerpts could link to sources and vice versa. Arrow buttons (shown in Figure 7) appear near both excerpts in the workspace as well as areas

of the original source document that have been excerpted. By touching the arrow button near an excerpt, the source document immediately moves to a new position in which it can show the excerpted text in its original context; likewise, touching the arrow button near the source document will move the excerpt object in the workspace into view. This mechanism effectively provides a way for users' own excerpts to be repurposed as a way to quickly navigate through the original source document. Further, we again take advantage of multitouch to enable users to view multiple contexts at once: multiple arrow-buttons can be held down to scroll and collapse the document to show multiple source areas simultaneously.

User feedback on excerpts. As the excerpt interactions were designed before the study, we investigated how they were used by participants. We found that the idea of extracting content via touch was in-line with user expectations, as even before seeing our prototype system, some users expressed an interest in using touch to pull content out of a document, and several described the existing copy/paste process as being laborious. Perhaps as a result, of the various functions in the prototype, the ability to create excerpts received the most strongly positive feedback, with eight users noting it as a feature they liked. In the AR study task, participants typically used excerpts as a way to aggregate content or to provide peripheral awareness. They noted that they pulled out quotes and even whole paragraphs so they could refer back to them later. And partly due to the support for grouping, several users discussed its value in allowing them to reorganize the content of the document. One user described this as, "What [the prototype] was really helpful for was because I had kind of written my own version of the story on the side." Excerpting, therefore, seems to have been used in much the way it was intended, but its frequency of use allowed it to easily consume available display space. So the findings of the study did not lead to significant changes to the excerpts themselves, but did provide part of our motivation for adding the fisheye workspace (above).

Annotation

The final active reading process we focused on was annotation. LiquidText provides two features for annotations: comments and highlights.

Comments. One important shortcoming of paper is the constraint it places on textual annotations such as comments. Comments on paper must generally be fit to the space of a small margin, and are typically only able to refer to a single page of text at a time. While software such as Microsoft Word or Adobe Reader avoid some of these difficulties, they still largely follow paper's paradigm; their annotations are thus still limited to single referents, and control of the size or scale of annotations is very limited, so available space is easily consumed.

Following from our general design goals, we sought to provide a more flexible alternative in LiquidText. Text annotations therefore act as attachments to the document itself. Like excerpts, they can be pulled off, rearranged, grouped with other items (including excerpts), and maintain persistent links back to the content they refer to. This especially helps support separation, aggregation and retrieval of annotations, as suggested by [16, 20].

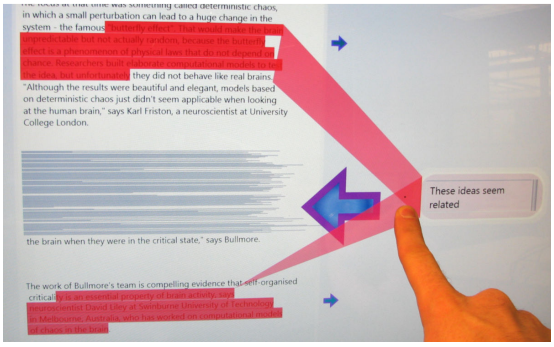

Figure 8. Comment's arrow-button collapses document to show both pieces of associated text.

LiquidText also breaks away from a one-to-one mapping between content and annotations. Rather, comment objects can refer to any number of pieces of content across one or more documents (Figure 8). And since they maintain two-way links to their referents, annotations can thereby act as persistent navigational affordances, letting readers freely create trails of links within and between documents.

To actually add a comment, the user simply selects the desired text and begins typing. This creates the comment attached to the document, and next to the selected text. The user can also create a comment by just pressing a button on the button bar (Figure 2, A), creating a comment that is not linked to other text. This allows comments to act as very simple text editors, letting the user begin to work on the content creation tasks which often come later in the AR workflow.

As the commenting functionality was included in our prototype, we received feedback on it in our user study. We found comments to be used frequently, employed by 10 of the 18 users. These annotations were used for summarization and commenting, such as identifying contradictions and noting relationships to other projects. For the particular task we used in our sessions, users did not appear to have a need to associate comments with more than one piece of content; nonetheless, one user spoke more generally of this being one of the most valuable features of the prototype,

"I liked the ability to annotate and connect different chunks, so the ability to say I want to highlight this piece, [and] I want to highlight this [second] piece, and make a comment about both of them for some later use."

Highlighting. The final annotation feature provided by LiquidText is the ability to highlight spans of text. While this is a common feature in many document processing systems (such as Adobe Reader), our assumption during our design process was that with LiquidText's extensive functions for content

extraction, commenting, and layout, highlighting would be of little value. Our user study showed this to be partly true—we found that while 9 of the 18 participants used highlighting, those users who tended to make more use of LiquidText's excerpt and commenting functions also seemed to make less use of the highlighting. One participant even described highlighting as possibly redundant with the system's other functions—especially excerpts.

Nonetheless, some users' preference for highlighting led us to further develop the feature beyond the humble implementation in our initial prototype. User feedback led to more color options, which we provided through a color palette (Figure 2, E). To improve efficiency, we also added an in-place highlight gesture: after selecting text, an upward swipe on the selection highlights it, with the swipe distance controlling the highlight intensity. Swiping downward dims and removes the highlight.

Participants also requested highlight aggregation functions. The final version of our system includes this by building on our existing collapse notion, and provides an auto-collapse function to show only the highlighted text, plus some variable amount of context.

Since the aggregation function was built on the concept of collapsing, we chose to control it in a similar way. While holding a soft-button accessed through the color palette, the user simply does the vertical pinch gesture to cause the document to collapse to show all the highlights. We chose this approach as a way to give the user precise, flexible control of the aggregation process, specifically: 1) the further the user pinches, the more of the space between the highlights is removed, letting the user select how much context is seen around highlights, and 2) the direction of the pinch (top finger brought to bottom or vice versa) controls whether the part of the document above or below the user's fingers is aggregated. This was necessary for scenarios where the user might wish to aggregate highlights in part of the document, while reading another part normally.

Implementation

The final LiquidText system described here is a fully functional application built on Windows Presentation Foundation and consisting of about 12,000 lines of C# and XAML code. LiquidText runs on Windows 7 and can work with any touchscreen using Windows' standard driver model.

DISCUSSION AND CONCLUSION

The LiquidText features—as well as the results of our design process, particularly exposure to users—are suggestive about the generalizability of the system to other domains.

Specific interactions. While LiquidText was intended to be used as a unit, several of its interactions may be applicable on their own in other contexts. Comments, for example, supporting multiple referents and two-way links allows them to act as simple navigational affordances, and might make sense in many document annotation situations. Collapsing likewise was generally well received and offers a natural way to distort the visualization of a document to hide content or bring material into proximity. Multitouch fisheye control was added only after our study, but may be viable in many navigation and layout situations as multitouch platforms become more popular.

General approach. Underlying the specific interaction techniques, however, is the notion of flexible, high degree-of-freedom representations, combined with comparably rich input devices to take advantage of them. We sought to apply this in the context of active reading, but the approach may be advantageous in other domains as well. Our formative study participants, for example, informed us that tasks like email, spreadsheet use, and manipulating presentations often intersect with AR, and share many of it challenges—like visualizing disparate pieces of content in parallel.

Our use of multitouch, specifically, turned out to be an essential component for realizing our design goals. Interactions involving many degrees of freedom, such as the fisheye lenses and collapsing, depended heavily on this type of input—they otherwise would likely have required many separate controls. Our study also supported this input model, with many users seeing touch as natural and very direct. However, they also described an interest in more multi-modal input, combining touch with gaze, a mouse, or a pen—which we may investigate in our future work.

But while the spatial flexibility of our representation is also essential to LiquidText, one potential pitfall of this approach is that a sense of position within the document can be sacrificed, impeding the spatial cues that help the reader maintain a "sense of a text" [5]. While a full assessment of this issue remains for future work, we did ask the participants about their sense of orientation while using LiquidText; of the 15 who addressed the issue, 13 claimed they maintained orientation when using the system.

Lessons Learned

The conception and iterative design of LiquidText offered a variety of lessons to us; we share several of these here.

Flexibility vs. structure. Throughout the design of LiquidText, we observed a tension between the flexibility we offered users in performing their task, versus the amount of predefined structure we provided to simplify their task. Even approaches to making structure optional sometimes seemed to impose structure. For example, we considered allowing users to name object groups, but to make the feature visible required a default title already in place; but any meaningless, default title (e.g., "Group 1") would be effectively begging the user to rename it, meaning that the choice of whether or not to name is no longer so optional. Given the goals of LiquidText, we generally opted for flexibility, but we leave for future work the question of how better to balance these competing ideals.

UI feel. It is difficult to disentangle the factors that contribute to users' impressions of a system, but over the course of our design process, people generally seemed to appreciate the "feel" that LiquidText provided. For example, p2's comments about the bubbliness and personality of the system, and the positive reactions from people to the putty-like connections between objects. Seeing that these things, at least in some cases, are not lost on users, hints to us that even in supporting mundane tasks like reading, design must encompass not just what the system does, but the message it conveys about itself and how it expects to be used.

Future Work and Conclusion

As we explore the potential value and uses of LiquidText, it will be important to conduct additional evaluation both to provide empirical assessment of our final designs, as well as understanding the types of AR tasks where it might be of the most use. One of these tasks, our formative study suggested, is multi-document scenarios—which LiquidText currently does not support. We plan to explore this in the future and believe some of LiquidText's interactions, such as the fish-eye zooming, will help provide the flexible workspace control that multi-document work requires [18].

In summary, LiquidText offers readers a substantially new way to interact with their documents. While most systems supporting AR replicate the experience of paper, paper is in many ways inflexible, and carries significant limitations. LiquidText takes a different approach, providing a flexible, malleable representation that gives the user fine grained control of the visual/spatial arrangement, navigational affordances, and annotations of their documents. To control this representation, LiquidText uses a vocabulary of multitouch gestures allowing efficient, often in-place, interaction.

ACKNOWLEDGEMENTS

We would like to thank Steelcase, Samsung, Dell and the NSF (award #IIS-0705569) for their support of this research.

REFERENCES

1. Adler, A., Gujar, A., Harrison, B.L., O'Hara, K. and Sellen, A. A diary study of work-related reading: design implications for digital reading devices *CHI*, ACM Press, 1998.
2. Askwall, S. Computer supported reading vs. reading text on paper: a comparison of two reading situations. *International Journal of Man Machine Studies*, *22*. 425-439.
3. Bouvin, N.O., Zellweger, P.T., Gronbaek, K. and Mackinlay, J.D. Fluid annotations through open hypermedia: using and extending emerging web standards *Proc. World Wide Web*, ACM, Honolulu, HI, USA, 2002.
4. Forlines, C., Wigdor, D., Shen, C. and Balakrishnan, R. Direct-touch vs. mouse input for tabletop displays *CHI 2007*, ACM, 2007.
5. Hansen, W.J. and Haas, C. Reading and writing with computers: a framework for explaining differences in performance. *Commun. ACM*, *31* (9). 1080-1089.
6. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H. and Buxton, B. Pen + touch = new tools *UIST 2010*, ACM.
7. Holz, C. and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints *CHI 2010*, ACM.
8. Holzinger, A., Holler, M., Schedlbauer, M. and Urlesberger, B. An investigation of finger versus stylus input in medical scenarios *ITI 2008*, IEEE, Dubrovnik, 2008.
9. Hornbaek, K. and Frokjaer, E. Reading patterns and usability in visualizations of electronic documents. *ACM Trans. Comput.-Hum. Interact.*, *10* (2). 119-149.
10. Jakobsen, M.R. and Hornbaek, K. Evaluating a fisheye view of source code *Proceedings CHI 2006*, ACM, 2006.
11. Liao, C., Guimbretiere, F., Hinckley, K. and Hollan, J. Papiercraft: A gesture-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.*, *14* (4). 1-27.
12. Marshall, C.C., Price, M.N., Golovchinsky, G. and Schilit, B.N. Introducing a digital library reading appliance into a reading group *Proc. of conf. Digital libraries*, ACM, Berkeley, CA, US, 1999.
13. Morris, M.R., Brush, A.J.B. and Meyers, B.R., Reading Revisited: Evaluating the Usability of Digital Display Surfaces for Active Reading Tasks. in *Tabletop*, (2007), 79-86.
14. Murray, T., Applying Text Comprehension and Active Reading Principles to Adaptive Hyperbooks. in *Cognitive Science*, (Boston, MA, 2003).
15. O'Hara, K. Towards a Typology of Reading Goals *RXRC Affordances of Paper Project*, Rank Xerox Research Center, Cambridge, UK, 1996.
16. O'Hara, K. and Sellen, A. A comparison of reading paper and on-line documents *CHI 1997*, ACM, 1997.
17. O'Hara, K., Smith, F., Newman, W. and Sellen, A. Student readers' use of library documents: implications for library technologies *CHI 1998*, ACM Press, 1998.
18. O'Hara, K., Taylor, A., Newman, W. and Sellen, A. Understanding the materiality of writing from multiple sources. *Int'l Journal of Human Computer Studies*, *56*. 269-305.
19. Price, M.N., Golovchinsky, G. and Schilit, B.N. Linking by inking: trailblazing in a paper-like hypertext *Proc. of Conf. Hypertext and hypermedia*, ACM, 1998.
20. Renear, A., DeRose, S., Mylonas, E. and Dam, A.v. An Outline for a Functional Taxonomy of Annotation, Presented at Microsoft Research, Redmond, WA, 1999.
21. Schilit, B.N., Golovchinsky, G. and Price, M.N. Beyond paper: supporting active reading with free form digital ink annotations *CHI 1998*, ACM, 1998.
22. Sellen, A. and Harper, R. Paper as an analytic resource for the design of new technologies *CHI 1997*, ACM, 1997.
23. Tashman, C. and Edwards, W.K., Active Reading and Its Discontents: The Situations, Problems and Ideas of Readers. in *CHI 2011*, (Vancouver, Canada, 2011), ACM.
24. Wu, M. and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays *UIST 2003*, ACM, 2003.
25. Zeleznik, R., Bragdon, A., Adeputra, F. and Ko, H.-S. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving *Proc of ACM UIST*, ACM, New York, NY, USA, 2010.