To be presented at: Symposium on Interactive Visual Information Collections and Activity (IVICA2009), Austin, TX., June, 2009.

Time-Based Authoring Tools for Informal Information Management

Yoshinari Shirai[1] Yasuhiro Yamamoto[2] Kumiyo Nakakoji[2,3]

[1]NTT Communication Science Laboratories, NTT Corporation

[2]RCAST, University of Tokyo
4-6-1 Komaba, Meguro
Tokyo, 153-8904, Japan

[3]SRA Key Technology Laboratory, Inc.
3-12 Yotsuya, Shinjuku,
Tokyo 160-0004, Japan

way@cslab.kecl.ntt.co.jp {yxy, kumiyo}@kid.rcast.u-tokyo.ac.jp

ABSTRACT

When we generate writings and drawings in creative knowledge work, such as in producing a company report, writing a paper, making presentation slides, or in developing programs, we often go back to what we have previously authored. This paper presents our history-centric approach to support such types of authoring. The approach first saves all editing actions in an editor in its history database, and then provides the Time-based Slicing mechanism to interactively visualize the historical data in various levels of granularity. We instantiate the approach through the design and development of HeEditor (History-Enriched Editor) for writing, and HeSketch (History-Enriched Sketch) for drawing.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical User Interfaces, Interaction styles.*

General Terms

Design, Human Factors

Keywords

Time-based Slicing, History-enriched Tools, History-centered Authoring Support, HeEditor, HeSketch

1. INTRODUCTION

We as knowledge workers generate a large number of writings and drawings on computers using text editing and drawing tools. As such an authoring task proceeds, the authored content keeps growing but not necessarily monotonously increases. New text or strokes are added while some of the older text or strokes are deleted. Some text may need to be removed because of page limitations and space constraints. The task may ends within a few hours or in a few days, or may even span over weeks and months. We save the state of writings or drawings as a "file," which is a snapshot of the current cluster of writings or drawings. When saving them as files, we decide their names, and their locations (i.e., where in the file directory structure to put them). Some people keep producing versions of the content by using different file names, and others keep overwriting the content onto a single file. As the task proceeds, the authored content originally saved within a single file may be split into multiple files, or the content of multiple files may be merged into a single file. We may share the current writings or drawings via email with our colleagues by attaching files. When receiving files sent by others, we may rename the files and save them at some locations.

Desktop search tools help us manage what we have previously authored on computer systems. Common desktop search tools, such as Google Desktop [5], Apple Mac OS X Spotlight [8], and Microsoft Windows Search [9], for instance, help us quickly search for files on PCs by using file names as well as file content. In addition to such content-based search, more specialized tools for personal information management, such as Stuff I've Seen [2] and phlat [1], provides context-based search, where users refine search results by selecting filters, using dates, file types, or authors.

During the authoring process, we may sometimes go back to what we have previously authored. This type of authoring strategy is typical when we generate writings and drawings in creative knowledge work, such as in producing a company report, writing a paper, making presentation slides, or in developing programs.

We think that existing search tools and information management tools are not enough for supporting this type of authoring because:

(1) a file is not something a user looks for; the content of the file is. The file as a unit of information to retrieve does not corresponds with the user's unit of concern; and

(2) a user cannot search for the content that is not saved within a file thereby the user has to keep worrying about when to save as a file. Even so, the user may not have saved a certain stage of writing or drawing by thinking that it would not be worth saving, or merely by mistake.

This paper presents an approach that supports authoring through informal information management. Our approach is first to save all editing actions in an editor in its history database, and second to provide mechanisms to browse the historical data in various levels of granularity through the Time-based Slicing (TbS) technique. The TbS mechanism allows users to interactively change how to cluster a set of historical data elements by changing time intervals. We instantiate the approach through the design and development of HeEditor (History-Enriched Editor) for writing, and HeSketch (History-Enriched Sketch) for drawing.

2. HISTORY-CENTRIC AUTHORING SUPPORT

It has been a commonly acknowledge exercise that we re-appropriate what we have previously written or drawn for the current authoring task.

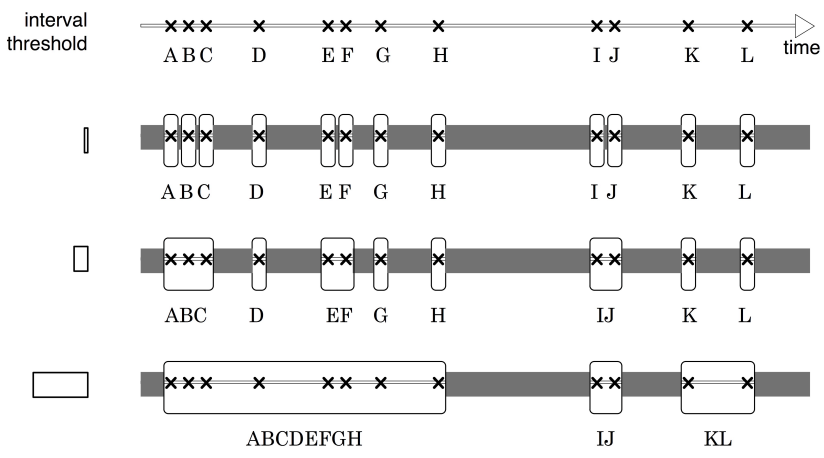


Figure 1: Time-based Slicing Mechanism

For instance, when we make slides in preparation for a paper presentation at a conference, we often go back to the file of the original paper. We may use a couple of diagrams used in the paper for the slides, and may copy a few phrases from the paper for the slides. When we write an extended journal paper based on the published paper, we may want to access unused sentences in the final version of the published paper that had been removed due to the page limitations. When we need to extend a computer program that we wrote some time ago, we review the existing source code to make sure what the implemented algorithms were. If we cannot remember the reason why we adopt some of the algorithms, we may confirm the process of modification by comparing the current version with the earlier versions of the code.

This paper proposes a history-centric approach to support such types of authoring where a user revisits what the user has previously authored and re-appropriate it for the current authoring task.

The approach consists of two elements: first, to collect all the editorial activities and store them, and second, to provide a browsing mechanism so that an author can peruse previous editorial stages according to the author's unit of concern.

Collecting and saving all the editorial actions, including key typing and stroke drawing, frees an author from worrying about whether and when to save the current writing and drawing as what files. Fischer [3] uses the notion of design-time and use-time to illustrate the difficulty of predicting a priori during the design-time how the artifact would be used in use-time. By saving the whole process of authoring, the user does not have to make a decision about how the authored document would be used during the design- (i.e., authoring-) time.

The browsing mechanism would be a key to take a full leverage of the existence of the entire authoring process. It is not the resulted writings and drawings but the experience of authoring the writings and drawings that the author wants to re-appropriate. The same individual differently interprets the same piece of information depending on his or her context [10]. The author should be able to interact with the historical stages of the authoring task at various levels of granularity during the re-appropriation process.

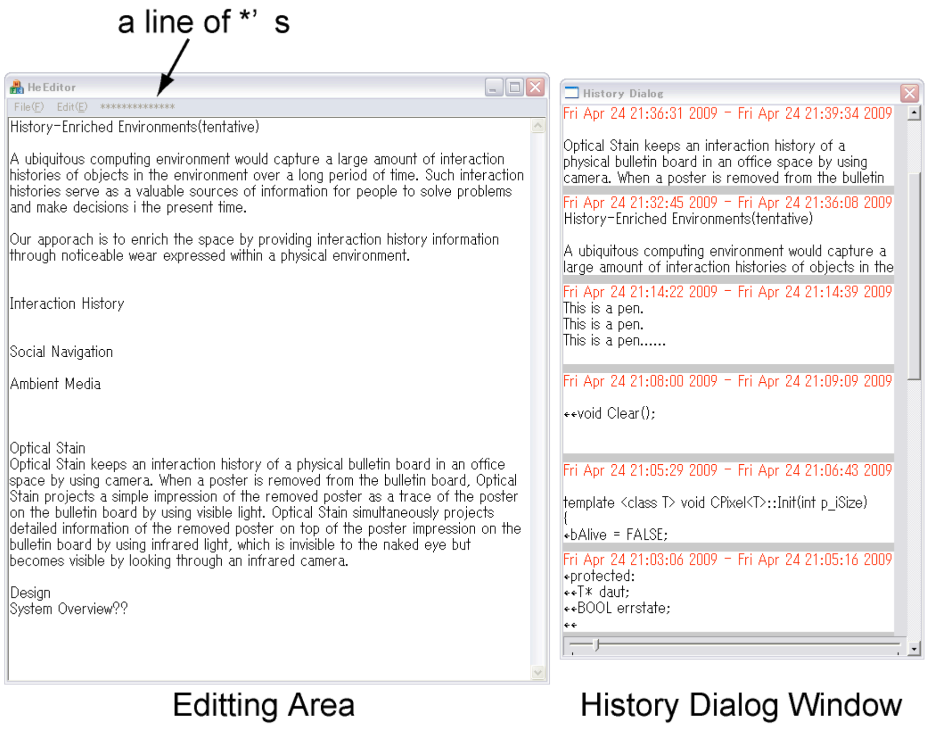
For a browsing mechanism to enable this process, we use the Time-based Slicing (TbS) technique. The following section describes the technique.

3. THE TIME-BASED SLICING TECHNIQUE

We have developed a Time-based Slicing (TbS) technique to interactively change the clustering of editorial history elements. The technique uses the notion of a history data, which is a sequence of time-stamped events. Each event is a history element. In text editing, each key action, such as inserting a letter or deleting a letter, is recorded as a history element. In drawing, each stroke operation is recorded as a history element.

The TbS technique uses the time interval of two temporally adjacent history elements as a way to visually slicing the history data. By visually slicing, we mean that a sliced group of history elements is visually presented at a time as one cluster. The basic idea is that a set of history elements that successively occurred within some range of time interval can be grouped as one cluster, and that changing the range of time interval would result in different sizes of clustering (see Figure 1). The idea comes from our daily experience. One usually types the letters of a word or a phrase in a successive manner with very short intervals. Typing stops a little before start typing a new word or a phrase. By taking larger intervals as a slice, we may get to identify a paragraph or a document. By looking at the time interval of the two successive editorial actions, we would be able to identify the position within the paragraph where one ponders in the midst of writing the paragraph.

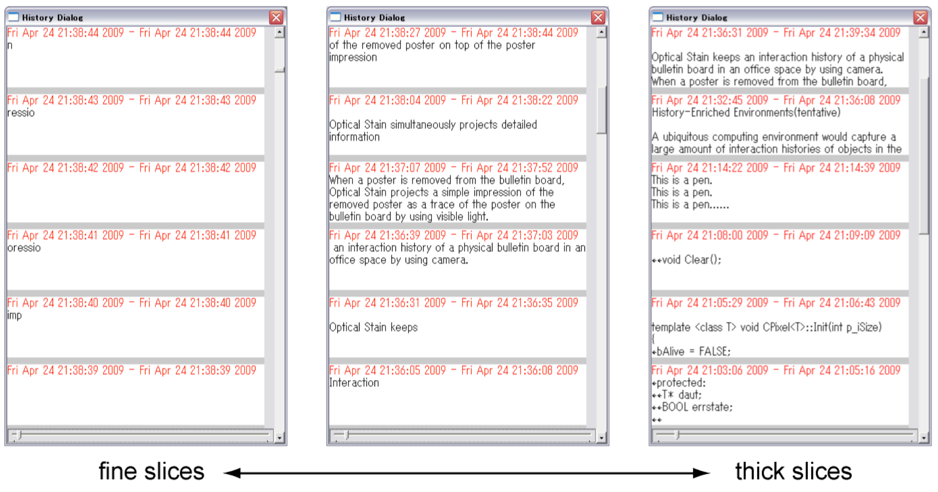
As illustrated in Figure 1, when specifying a small time interval as a threshold value, history elements are clustered into a large number of small groups. The minimum size of a cluster is composed of one history element. When specifying a larger time interval as a threshold value, history elements are clustered into a smaller number of larger groups. Note that this time interval ranges from milliseconds (time interval between typing two characters) to hours and days (time interval between the last editorial action in the previous authoring task and the first editorial action in the next authoring task; that is, the duration of no editorial actions) as our approach collects all the editorial



```
History-Enriched Environments(tentative)

A ubiquitous computing environment would capture a large amount of interaction
histories of objects in the environment over a long period of time. Such interaction
histories serve as a valuable sources of information for people to solve problems
and make decisions i the present time.

Our apporach is to enrich the space by providing interaction history information
through noticeable wear expressed within a physical environment.


Interaction History


Social Navigation

Ambient Media



Optical Stain
Optical Stain keeps an interaction history of a physical bulletin board in an office
space by using camera. When a poster is removed from the bulletin board, Optical
Stain projects a simple impression of the removed poster as a trace of the poster
on the bulletin board by using visible light. Optical Stain simultaneously projects
detailed information of the removed poster on top of the poster impression on the
bulletin board by using infrared light, which is invisible to the naked eye but
becomes visible by looking through an infrared camera.

Design
System Overview??
```

```
Fri Apr 24 21:36:31 2009 - Fri Apr 24 21:39:34 2009

Optical Stain keeps an interaction history of a
physical bulletin board in an office space by using
camera. When a poster is removed from the bulletin

Fri Apr 24 21:32:45 2009 - Fri Apr 24 21:36:08 2009
History-Enriched Environments(tentative)

A ubiquitous computing environment would capture a
large amount of interaction histories of objects in the

Fri Apr 24 21:14:22 2009 - Fri Apr 24 21:14:39 2009
This is a pen.
This is a pen.
This is a pen......

Fri Apr 24 21:08:00 2009 - Fri Apr 24 21:09:09 2009

↔void Clear();

Fri Apr 24 21:05:29 2009 - Fri Apr 24 21:06:43 2009

template <class T> void CPixel<T>::Init(int p_iSize)
{
↔bAlive = FALSE;

Fri Apr 24 21:03:06 2009 - Fri Apr 24 21:05:16 2009
↔protected:
↔↔T* daut;
↔↔BOOL errstate;
↔↔
```

Figure 2: HeEditor: Editing Area (left) and History Dialog Window (right)



```
Fri Apr 24 21:38:44 2009 - Fri Apr 24 21:38:44 2009
n

Fri Apr 24 21:38:43 2009 - Fri Apr 24 21:38:43 2009
ressio

Fri Apr 24 21:38:42 2009 - Fri Apr 24 21:38:42 2009

Fri Apr 24 21:38:41 2009 - Fri Apr 24 21:38:41 2009
oressio

Fri Apr 24 21:38:40 2009 - Fri Apr 24 21:38:40 2009
imp

Fri Apr 24 21:38:39 2009 - Fri Apr 24 21:38:39 2009
```

```
Fri Apr 24 21:38:27 2009 - Fri Apr 24 21:38:44 2009
of the removed poster on top of the poster
impression

Fri Apr 24 21:38:04 2009 - Fri Apr 24 21:38:22 2009

Optical Stain simultaneously projects detailed
information

Fri Apr 24 21:37:07 2009 - Fri Apr 24 21:37:52 2009
When a poster is removed from the bulletin board,
Optical Stain projects a simple impression of the
removed poster as a trace of the poster on the
bulletin board by using visible light.

Fri Apr 24 21:36:39 2009 - Fri Apr 24 21:37:03 2009
 an interaction history of a physical bulletin board in an
office space by using camera.

Fri Apr 24 21:36:31 2009 - Fri Apr 24 21:36:35 2009

Optical Stain keeps

Fri Apr 24 21:36:05 2009 - Fri Apr 24 21:36:08 2009
Interaction
```

```
Fri Apr 24 21:36:31 2009 - Fri Apr 24 21:39:34 2009

Optical Stain keeps an interaction history of a physical
bulletin board in an office space by using camera.
When a poster is removed from the bulletin board,

Fri Apr 24 21:32:45 2009 - Fri Apr 24 21:36:08 2009
History-Enriched Environments(tentative)

A ubiquitous computing environment would capture a
large amount of interaction histories of objects in the

Fri Apr 24 21:14:22 2009 - Fri Apr 24 21:14:39 2009
This is a pen.
This is a pen.
This is a pen......

Fri Apr 24 21:08:00 2009 - Fri Apr 24 21:09:09 2009

↔void Clear();

Fri Apr 24 21:05:29 2009 - Fri Apr 24 21:06:43 2009

template <class T> void CPixel<T>::Init(int p_iSize)
{
↔bAlive = FALSE;

Fri Apr 24 21:03:06 2009 - Fri Apr 24 21:05:16 2009
↔protected:
↔↔T* daut;
↔↔BOOL errstate;
↔↔
```

Figure 3: Time-based Slicing in HeEditor's History Dialog Window

actions over a long period of time. Even when the tool is quit, it continues recording editorial actions as "no acitons" till the tool is reopened.

4. TWO TOOLS: HeEditor AND HeSketch

We have implemented HeEditor and HeSketch based on the approach described in the previous sections.

4.1 HeEditor for Writing

HeEditor is a tool for authoring text. The tool captures and stores all the keystroke data in its history databases. Each time a user makes a keystroke within HeEditor, a history element is composed and added to the history database. Each history element consists of a tuple of the key operation type (i.e., insertion or deletion), its position (i.e., the number of characters from the head), its character code, and its timestamp.

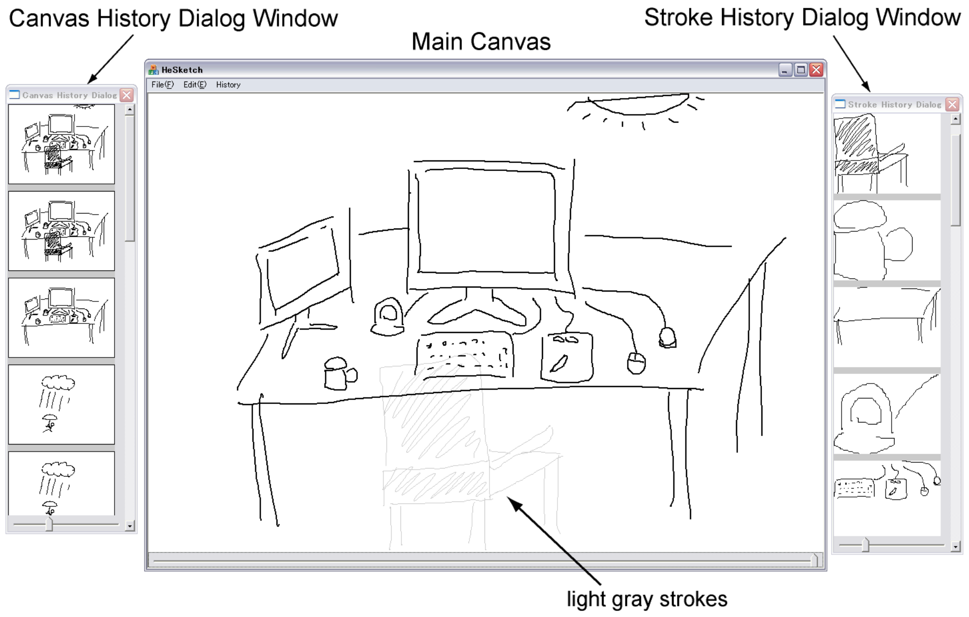


Figure 4: HeSketch (center), its Canvas History Dialog Window (left) and Stroke History Dialog Window (right)

HeEditor displays a line of *'s in the menu bar of the editing area. The number of *'s indicates the approximate amount of keystrokes made within HeEditor. An * is added each time a certain number of editing actions are stored in the history database.

Figure 2 shows HeEditor. The editing area (Figure 2 left) provides basic editing functionality, such as inserting and deleting characters, copying/cutting/pasting text, or clearing the editing area. Each of such operation is stored in the history database as a timestamped history element.

The History Dialog window (Figure 2 right) shows a list of clusters of history elements that are grouped together using the TbS mechanism. The list of clusters is ordered from the bottom to the top according to their timestamps. The user may scroll up or down to browse the list of historical states, and thus view how the authoring has proceeded.

When the user clicks on one of the clusters displayed in History Dialog, the current editing area is first stored in the history database then cleared, and finally the state of the editor displayed in the selected cluster is brought to the current editing area so that the user may examine the details of the cluster. The user may continue editing with the retrieved state of the edits, go back to the previous state by using History Dialog, or start writing from scratch by clearing the writing area.

Using the handle located in the bottom of the History Dialog window, the user specifies the time-interval threshold. If the user moves the handle toward right, the threshold becomes larger, resulting in thicker slices (i.e., the history events happened in a longer time period are grouped together) (see Figure 3). As the user moves the handle back and forth, the History Dialog window dynamically updates the visual representations of them, allowing the user to explore the desired clustering size for him or her to focus on.

4.2 HeSketch for Drawing

HeSketch is a tool for drawing sketches. The tool captures and stores both the stroke data and the states of the canvas (i.e., the drawing area) in its history databases. Each time a user draws a stroke in the canvas of HeSketch, a stroke history element is composed and stored in its database. Each stroke history element consists of a tuple of the stroke ID, a set of X-Y coordinates that compose the stroke, and the timestamps of the starting and finishing times of drawing the stroke. When a user adds a stroke either by drawing or copying strokes from the history database, or removes one or more strokes in the drawing area, a new canvas history element is composed and stored in its database. Each canvas history element consists of a tuple of a set of currently displayed stoked ID's in the canvas, and its timestamp.

The canvas allows the user to draw a stroke by moving a mouse cursor while holding its button. The canvas provides a stroke deletion operation with a specialized visual effect. When a user selects one or more strokes in the canvas by specifying the area by using a mouse, the system draws a bounding box with dotted lines. When the user clicks on the top right corner of the bounding box, the strokes are immediately become very light gray, less visible in the canvas. This is a removing operation. The color of the removed strokes keep fading out and they completely become invisible (i.e., white) when the certain amount of time passes after the user evokes the removal operation.

Figure 4 shows the main canvas window of HeSketch, and its two history-browsing components: the Canvas History Dialog window (Figure 4 left) and the Stroke History Dialog window (Figure 4 right). The main canvas area shows several light-gray strokes, which indicate that they have been recently deleted from the canvas. Note that those strokes are also displayed in each of the top of the Canvas History Dialog window and Stroke History Dialog window.

The Canvas History Dialog window displays a list of thumbnails of the canvas states stored in the canvas history database. The Stroke History Dialog window displays a list of clusters of strokes stored in the stroke history database (note that a cluster of strokes are enlarged to fit the thumbnail display area). In the same manner as HeEditor's History Dialog window as described above, such clusters are ordered from the bottom to the top according to their timestamps (i.e., the newer ones are toward the top).

When the user clicks on one of the thumbnails displayed in the Canvas History Dialog window, the current state of the canvas as well as the strokes are first stored in the two history databases, then the canvas is cleared, and finally the state of the corresponding historical state is brought in to the current canvas area. When the user clicks on one of the clusters displayed in the Stroke History Dialog window, the set of strokes consisting of the cluster are added to the current Canvas area.

By using the handle located in the bottom of the each Canvas History Dialog window and Stroke History Dialog window, the user specifies the time-interval threshold for slicing the clusters of the historical data in the same way as HeEditor's History Dialog window. As the user moves the handle and changes the threshold, the History Dialog window dynamically updates the visual representation of the clusters, as shown in Figures 5 and 6.

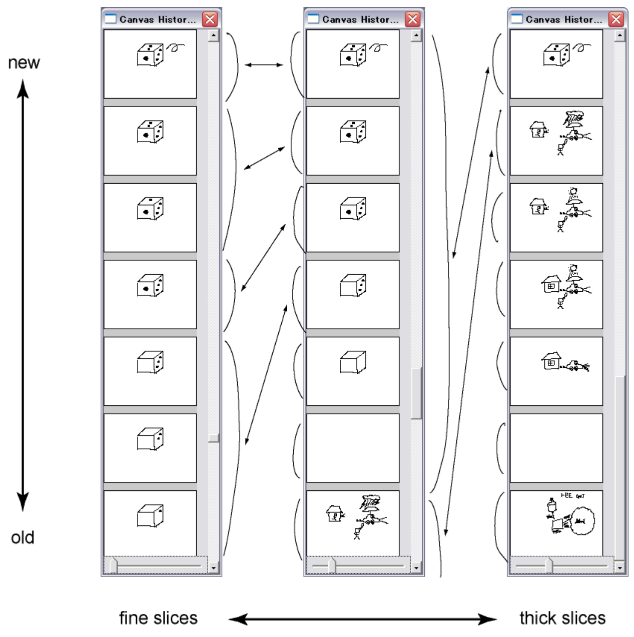


Figure 5: Time-based Slicing in HeSketch's Canvas History Dialog window

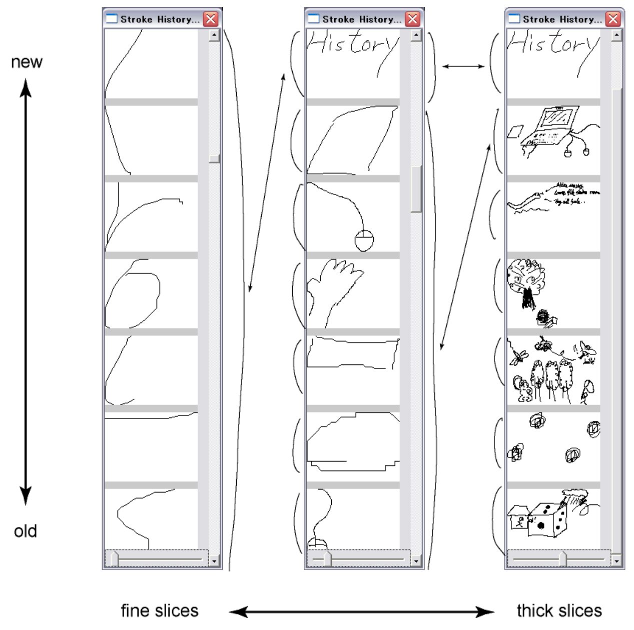


Figure 6: Time-based Slicing in HeSketch's Stroke History Dialog window

5. RELATED APPROACHES

This paper presents our history-centric approach for supporting authoring and managing authored content. From the viewpoint of using temporal information for managing authored material, a physical notebook is a good example.

People take notes in a notebook typically in the order of pages starting from the first page to the last page. Text notes, sketches, and sentences on a notebook are thus temporally ordered from the starting page toward the subsequent pages. Such content continue to remain on the notebook unless we explicitly erase it. That is, notebooks capture and store all of the writing and drawing histories and visualize them in the temporal order. The historical content is divided by "pages" and cannot be freely changed. Our TbS technique can be viewed as a way to interactively changes the "page-divisions" within a notebook.

The goal of our TbS technique is to interactively visualize the historical states of writing and drawing processes with various levels of granularity so that a user can explore the "right" size of a history cluster that corresponds to the user's unit of concern. Existing tools visualize a variety of information materials using such temporal information.

For instance, LifeStreams stores the entire documents in a time-ordered list and presents them as a stream of documents [4]. A concept of Time-Machine Computing allows a user to visit the past states of a computer system [11]. Its TimeScape component visualizes the history of all modifications of a desktop environment, such as producing and removing a file from the desktop [11]. Stuff I've Seen [2] and phlat [1] help users find files on PCs by using keywords and present the results lists, which are sorted by time. TimeSpace [7] and Milestones in Time [12] plots the files on a time-line diagram. The system by Viegas, et.al [14] visualizes the modification process of wiki. These examples primarily focus on file-based information and not the processes of how each file is authored.

Activity tracing tools are most similar to our approach. Edit wear and read wear capture and store the text editing history of documents and visualize them onto an editor window's scrollbar [6]. They use the editorial historical data to guide a user by suggesting which part of the document to pay attention to. TypeTrace is a software system that records and plays back typing on computers [13]. It helps users to reflect in how the writing has proceeded. ART019 is a sketchbook interface based on a time-based representation for hand-drawn strokes, where free-hand drawing is recorded as a sequence of time-stamped strokes [15]. ART019 allows a user to access strokes through the time-based representation, and go back to any point in the previously made snapshots of his or her drawing. Our TbS technique would add powerful interactivity with editorial historical data for these tools.

6. FUTURE DIRECTIONS

This paper presented two tools, HeEditor and HeSketch, which are designed based on the Time-based Slicing technique. Both tools are currently implemented as tools for viewing a single stream of editorial data. We have been working on designing a tool that allows a user to engage in an authoring task by looking at multiple streams of editorial data, including writing, sketching, programming, and email messaging.

The two tools are currently implemented as prototypes as a proof of concept. We plan to extend the tools so that they become feasible for practical usage. We would then conduct user experiments with real editorial data collected over a long period of time.

7. REFERENCES

[1] Cutrell, E., Robbins, D., Dumais, S., and Sarin, R. 2006. Fast, flexible filtering with phlat. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 261-270.

[2] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., and Robbins, D. C. 2003. Stuff I've seen: a system for personal information retrieval and re-use. SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. 72-79.

[3] Fischer, G., and Nakakoji, K. 1991. Making design objects relevant to the task at hand. In Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91). 67-73.

[4] Freeman, E., and Gelernter, D. 1996. Lifestreams: a storage model for personal data. SIGMOD Rec. 25, 1, 80-86.

[5] Google Desktop, http://desktop.google.com/

[6] Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T. 1992. Edit wear and read wear. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 3-9.

[7] Krishnan, A. and Jones, S. 2005. TimeSpace: activity-based temporal visualisation of personal information spaces. Personal Ubiquitous Comput. 9, 1, 46-65.

[8] MacOSX Spotlight, http://www.apple.com/macosx/features/300.html#spotlight

[9] Microsoft Windows Search, http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx

[10] Oda, T., Yamamoto, Y., and Nakakoji, K. 2006. Use-Centric Information Re-presentation for Creative Knowledge Work. Symposium on Interactive Visual Information Collections and Activity (IVICA2006).

[11] Rekimoto, J. 1999. Time-machine computing: a time-centric approach for the information environment. UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology. 45-54.

[12] Ringel, M., Cutrell, E., Dumais, S., and Horvitz, E. 2003.Milestones in Time: The Value of Landmarks in Retrieving Information from Personal Stores. INTERACT2003.

[13] TypeTrace, http://typetrace.jp/about/tt/

[14] Viegas, F. B., Wattenberg, M., and Dave, K. 2004. Studying cooperation and conflict between authors with history flow visualizations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Vienna, Austria, April 24 - 29, 2004). 575-582.

[15] Yamamoto, Y., Nakakoji, K., Nishinaka, Y., and Asada, M. 2006. ART019: A Time-Based Sketchbook Interface. Technical Report, KID Laboratory, RCAST, University of Tokyo.