

# Annotating 3D Electronic Books

Lichan Hong, Ed H. Chi, and Stuart K. Card

Palo Alto Research Center

3333 Coyote Hill Road, Palo Alto, CA 94304

{hong, echi, card}@parc.com

## ABSTRACT

The importance of annotations, as a by-product of the reading activity, cannot be overstated. Annotations help users in the process of analyzing, re-reading, and recalling detailed facts such as prior analyses and relations to other works. As electronic reading become pervasive, digital annotations will become part of the essential records of the reading activity. But creating and rendering annotations on a 3D book and other objects in a 3D workspace is non-trivial. In this paper, we present our exploration of how to use 3D graphics techniques to create realistic annotations with acceptable frame rates. We discuss the pros and cons of several techniques and detail our hybrid solution.

## Author Keywords

Annotation; 3D Book; Digital Library; User Interface.

## ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical User Interfaces; I.7.4 [Document and Text Processing]: Electronic Publishing; H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia—User Issues; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques. **General Terms:** Algorithms, Human Factors

## INTRODUCTION

Annotation is as old as the practice of reading itself. Annotation is an integral part of reading, as observed by Marshall's study of annotations made by university students in their textbooks [8]. Similar observations were reported by O'Hara and Sellen, comparing reading on paper to reading on computer [9]. Indeed, in the brave new world of electronic reading, most 2D-based electronic readers provide means to annotate pages (e.g., Acrobat Reader, MS Word, and XLibris [11]). XLibris allows users to scribble notes, draw figures, and annotate text directly on the page using pens and highlighters of different colors.

More recently, several research groups and commercial products have started to explore the display of digital documents in a 3D form, simulating the appearance of a physical book [1,5,6,13], although many of these are actually 2D implementations with simulated 3D effects. Zinio

Copyright is held by the author/owner(s).

CHI 2005, April 2-7, 2005. Portland, Oregon, USA.

ACM 1-59593-002-7/05/0004.

Reader [13], for example, simulates the experience of reading a magazine in 3D, while FlipViewer [6] simulates the experience of browsing through a paper product catalog. Chu et al. [5] devoted their efforts to the 3D modeling of the pages and the book. Learning from our past experiences with the 3D WebBook [2], last year we reported a new system called 3Book [1], which is able to represent books of unlimited length and offers supports for reading of large format books. We envision 3Book to “serve as the basis for further experiments on digital libraries and systems to aid sensemaking in information-intensive tasks” [1]. For example, 3Book provides semantically enhanced indexes [3], enabling users to efficiently find information of interest. 3Book also enables semantic highlights to be automatically computed for more productive reading [4]. Beyond books, 3D workspace user interfaces such as Task Gallery [10] and Project Looking Glass [12] involve 3D documents and other objects that would benefit from annotation.

The goal of this work is to investigate suitable graphical and user interface techniques for supporting interactive annotation in 3Book. Surprisingly, although annotation has been repeatedly identified as one of the most essential features to the success of electronic readers [8,9], no prior work to our knowledge has been devoted to enabling annotation of 3D documents. In our exploration, user interface design, computer graphics techniques, and system response time all turn out to strongly interact with one another. In this paper, we present three annotation visualization techniques, and discuss their limitations for 3D interactive annotation. We then disclose details of our final hybrid solution.

## ANNOTATING 3D PAGES

In this paper we consider two types of annotation: *highlight* and *scribble*. A *highlight* is defined as an area on the page, which can be used to highlight a figure, a paragraph, or even a previously added annotation in the margin. On the other hand, a *scribble* is defined as an annotation made with pen strokes. Examples of scribbles include underlines of words and sentences, brief notes written between lines, and marginal symbols like arrows and brackets. In our design, we break down a scribble further into a set of strokes (i.e., polylines). As an example, depending on how the user draws an arrow, it may consist of two strokes “^” and “|”.

In 3Book, the geometry of a visible page is represented with a *polygonal mesh* and the content of the page is repre-

sented with a *texture* painted on the geometry. As the user sketches out an annotation with a pen on a tablet, we transform the pen trajectory from the screen coordinate system to the coordinate system of the page. That is, we *project* the stroke of the annotation onto the 3D surface of the page [7]. As a result, we know where on the page the pen first touched, where on the page the pen next moved to, and where on the page the pen finally detached from. We construct an abstract representation of the annotation from the trajectory data and store it in a data structure correlated with the page. The abstract representation of an annotation includes its location on the page, its width and height, its color, its transparency, and its time of creation, etc.

For rendering an annotation, we have experimented with three different techniques:

- The *transparent geometry* technique places transparent geometry (e.g., polygons or lines) representing an annotation over the page.
- The *vertex coloring* technique shows an annotation by modifying the color values at the vertices of the polygonal mesh representing the page.
- The *texture coloring* technique updates the pixels of the page texture to achieve an annotating effect.

### Transparent Geometry

One way to render an annotation is to create a transparent 3D object and superimpose this object over the page area to be annotated. For a highlight, this object will be a 3D polygon, and, for a scribble, a set of 3D polylines. The color of the object reflects the color of the pen, and the opacity of the object corresponds to the ink density of the pen. Figure 1 shows a yellow highlight of a figure on the left page of a book.

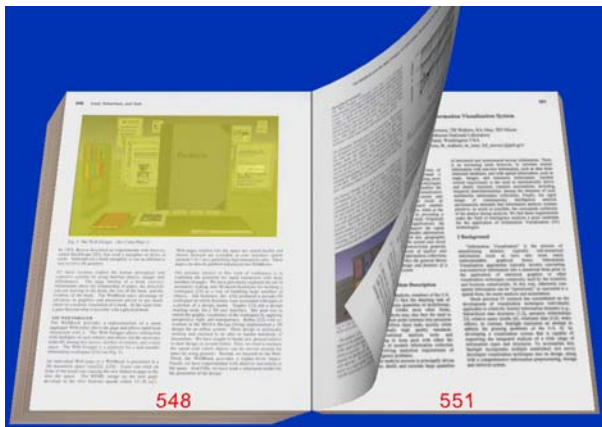


Figure 1: A yellow, transparent 3D rectangle is placed on top of the left page. A transparent geometry would have to be deformed along with the page area that it annotates.

Unfortunately, Z-fighting, a classical 3D graphics problem, arises when two overlapping, co-planar polygons P1 and P2 are rendered. To avoid this problem, the yellow rectangle needs to be slightly elevated from the left page towards the camera by a certain distance. The size of this distance is dependent on the graphics hardware. When multiple ge-

ometries overlap a common area, they need to be offset from each other. Consequently, as more and more annotations are added and deleted, it may become non-trivial to figure out how to offset a newly created geometry.

A more severe drawback of transparent geometry emerges during page turning, as illustrated in Figure 1. Since the turning page is modeled as a curved surface and dynamically changes its shape during page turning [1,5], any transparent geometry associated with the turning page needs to be deformed along with the page area that it annotates. This deformation is complicated to implement, especially for complex annotation shapes.

### Vertex Coloring

Another way to render an annotation is to color those polygonal vertices within the page area which is to be annotated. Since the page is represented as a polygonal mesh and a highlight covers an area on the page, it is fairly straightforward to identify those vertices lying inside the highlighted area, as illustrated in Figure 2. The idea is to re-evaluate the colors of those vertices as a function of the pen color and ink density. Subsequently in the rendering, the colors of those vertices will be automatically blended with the page texture, as shown in Figure 3.

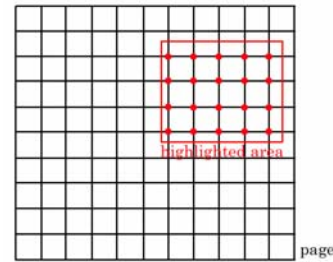


Figure 2: Vertices within a highlighted area are colored based on the pen color and ink density.

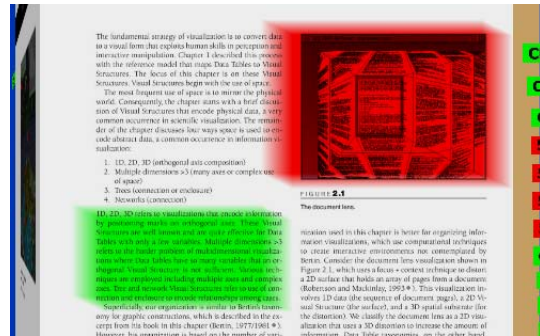


Figure 3: Two page areas are highlighted in red and green, respectively.

Unlike the transparent geometry technique, the vertex coloring technique can easily accommodate deformation of the page surfaces. However, as can be seen from Figure 3, the vertex coloring technique produces noticeable artifacts, due to the bilinear interpolation of vertex colors. Although using a finer polygonal mesh will, to a certain degree, ameliorate the highlight boundary, it comes with the cost of processing more polygons. Additionally, the mesh will never be

fine enough to support scribbles, which require pixel-size resolutions.

### Texture Coloring

Yet another way to visualize an annotation is to modify the texture of the page. For each annotation, we need to figure out where it projects onto the page, and then compute the projection onto the texture. Given an annotation like a highlight or a scribble, for each texture pixel that needs to be modified, we re-evaluate its color as follows:

$$C_t = (1.0 - density) * C_i + density * C_p,$$

where  $C_t$  is the pixel color,  $C_p$  is the pen color, and  $density$  is the pen ink density normalized to be in the range of 0.0 to 1.0. This blending operation results in a new texture as shown in Figure 4. Applying the new texture in the rendering gives us an image as shown in Figure 5.



Figure 4: Texture pixels within a highlighted area are blended with a red color.

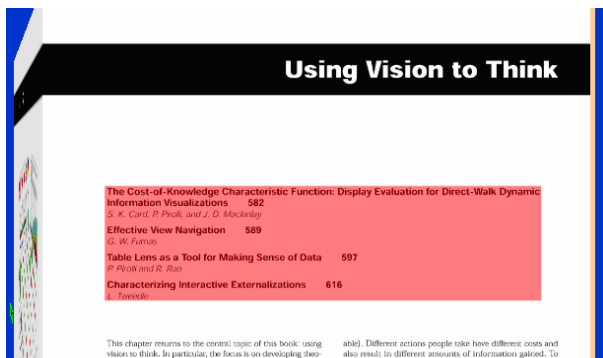


Figure 5: Rendering result after applying the new texture of Figure 4 to the page geometry.

In contrast to the vertex coloring technique, the texture coloring technique produces a well-defined boundary for the highlight. This is because, to preserve text quality, the page texture has a much higher resolution than the polygonal mesh. Consequently, the result of bi-linear interpolation of textures coordinates is visually more appealing than that of bi-linear interpolation of vertex colors.

Unfortunately, the computation of the blending operation above is costly. Given the number of pixels that we have to process, the blending operation prevents us from delivering frame rates for scribbles to work adequately. The low frame rates mean that a detailed sample of the pen stroke cannot be captured. Figure 6, which was captured on a high-end desktop (Dell Precision 340 equipped with an ATI

9700PRO graphics card), demonstrates the problem that we ran into using the texture coloring technique. In this case, our goal was to scribble “PARC TAP” in the top margin of page 127. The reason why the curves on the letters “P” and “C” were not captured adequately was due to the low frame rates. In other words, the latency between two consecutive frames was so significant that the system received very few pen events for each of the curves.

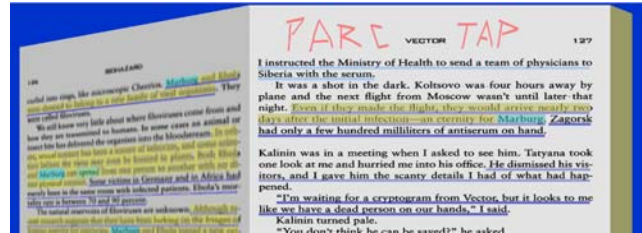


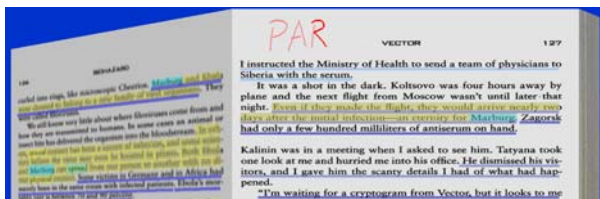
Figure 6: Using the texture coloring technique, “PARC TAP” was scribbled in the top margin of page 127. Note how system response time limits quality of annotation with this technique.

### Hybrid Approach

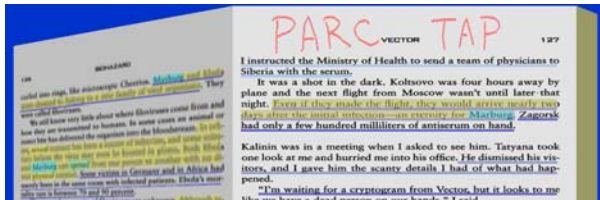
We have adopted a hybrid approach aiming to improve the rendering speed during annotation creation. The idea is that while the user is sketching out an annotation, instead of updating the page texture to visualize the annotation, we use 3D *transparent geometry* to represent the annotation. We then map the annotation onto the texture of the page after the sketch. In the case of a stroke, the steps are:

1. We first compute the pen trajectory in the coordinate system of the page.
2. We then bring the polyline representing the stroke towards the camera and place it slightly behind the near clipping plane of the camera frustum. This ensures that the stroke will always appear on top of the page. Since the polyline can be efficiently rendered and the page texture stays unchanged, our technique is independent of the texture resolution, enabling us to significantly increase the frame rate during pen stroke capturing.
3. When the user lifts her pen off the screen and moves on to write the next stroke, we take advantage of the brief time elapse between the current pen-up action and the next pen-down action, removing the polyline from the 3D scene and updating the page texture to visualize the newly added stroke with the texture coloring technique.

Figure 7a shows a snapshot taken when we scribbled “PARC TAP” in the top margin. The letters “P” and “A”, as well as the first stroke of the letter “R”, were visualized with the texture coloring technique. The second stroke of the letter “R”, which was being sketched out, was represented as a 3D polyline. Figure 7b shows a scribble of “PARC TAP” that we captured on the same desktop as that of Figure 6. Due to the relatively high frame rate achieved by the hybrid technique, we did not have to deliberately slow down our pen movement when writing the scribble. It can be seen that in comparison with Figure 6, the scribble in Figure 7b naturally captured all the curves of the letters.



(a)



(b)

**Figure 7: Two frames taken while using the hybrid technique to scribble “PARC TAP” in the top margin. Note improved quality of annotation with this technique over Figure 6.**

Obviously, the hybrid approach can be used to support the creation of a highlight as well. As the user is sketching out the highlight, we first compute the pen trajectory in the coordinate system of the page. We then construct a transparent 3D rectangle from the abstract representation of the highlight. We place the rectangle behind the near clipping plane as before, ensuring the rectangle will always be on top, covering the highlight area. The size of the rectangle changes interactively as the user moves her pen. When the user lifts her pen, indicating that she is done with the highlighting, we remove the rectangle from the 3D scene and update the texture to visualize the highlight.

## IMPLEMENTATION

The hybrid technique is implemented and fully integrated with other features of our 3Book system. We store the annotation data separately from the page texture and the page geometry, because the page data is public data which may be shared by multiple users, and annotation data is private data created and used by a single user. The separation provides flexibility to access un-annotated pages, or the annotations alone.

To mitigate the cost of the blending operation when the user re-visits an annotated page during page turning, we adopt a multi-resolution strategy to represent the page texture. Specifically, in preprocessing we acquire the page texture in three resolutions 256x256, 512x512, and 1024x1024. When a page becomes newly visible due to page turning, we load its low-resolution texture from disk and use the texture coloring technique to add its annotations. This significantly reduces the time interval for the first frame of the page turning animation to appear, creating a sense of rapid user responsiveness. At the same time, we launch a separate thread to load its high-resolution texture from disk and add its annotations accordingly. At the end of the page turning animation, the low-resolution texture is replaced with the high-resolution texture to enhance image quality.

For removing an annotation, the user draws a scratching gesture. We transform this gesture from the screen coordinate system to the coordinate system of the page. Any annotation whose visual representation intersects with the gesture will then be deleted.

## CONCLUSION

As shown in this brief paper, implementing a simple feature to support annotations of 3D books is non-trivial. We are sharing our experience in this design space so that other researchers can understand the pitfalls and issues in the implementations.

3D workspaces, such as Task Gallery [9] and Project Looking Glass [11], will eventually involve objects such as 3D books, shelves of books, and digital libraries that are more complex than just application windows. A technique for annotating such objects is an important building block for the design space of such 3D user interfaces.

## ACKNOWLEDGMENTS

We thank Jock Mackinlay for his contributions to the design of the 3Book system.

## REFERENCES

1. Card, S. K., Hong, L., Mackinlay, J., and Chi, E. H. 3Book: A Scalable 3D Virtual Book. *Proc. CHI'04 Conference Companion*, 1095-1098.
2. Card, S. K., Robertson, G. G., and York, W. The WebBook and the Web Forager: An Information Workspace for the World-Wide Web. *Proc. CHI'96*, 111-117.
3. Chi, E. H., Hong, L., Heiser, J., and Card, S. K. eBooks with Indexes that Reorganize Conceptually. *Proc. CHI'04 Conference Companion*, 1223-1226.
4. Chi, E. H., Hong, L., Gumbrecht, M., and Card, S. K. ScenTHighlights: Highlighting Conceptually-Related Sentences During Reading. *Proc. 10th International Conference on Intelligent User Interfaces*, 272-274.
5. Chu, Y.-C., Bainbridge, D., Jones, M., and Witten, I. H. Realistic Books: A Bizarre Homage to An Obsolete Medium? *Proc. 2004 Joint ACM/IEEE Conference on Digital Libraries*, 78-86.
6. FlipViewer. <http://www.flipviewer.com>, 2005.
7. Hanrahan, P. and Haeberli, P. Direct WYSIWYG Painting and Texturing on 3D Shapes. *Proc. SIGGRAPH'90*, 215-223.
8. Marshall, C. Annotation: From Paper Books to Digital Library. *Proc. 1997 ACM Digital Library*, 131-140.
9. O'Hara, K. and Sellen, A. A Comparison of Reading Paper and On-Line Documents. *Proc. CHI'97*, 335-342.
10. Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinkley, K., Ridsen, K., Thiel, D., and Gorokhovskiy, V. The Task Gallery: A 3D Window Manager. *Proc. CHI'00*, 494-501.
11. Schilit, B., Golovchinsky, G., and Price, M. Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations. *Proc. CHI'98*, 249-256.
12. Sun Microsystems. Project Looking Glass. [http://www.sun.com/software/looking\\_glass](http://www.sun.com/software/looking_glass), 2005.
13. Zinio Reader. <http://www.zinio.com>, 2005.