

Panel Beat: Layout and Timing of Comic Panels

William Bares

Millsaps College, Department of Computer Science,
Jackson MS 39210, USA

{bareswh}@millsaps.edu

<http://home.millsaps.edu/bareswh/>

Abstract. In comic book conventions, the size of a panel is generally proportional to the amount of narrative time that passes or the amount of time a reader spends viewing the panel. This poster paper describes work in progress to develop an automated system to assist in generating comic book panel layouts from a given sequence of panel artwork, the desired narrative duration of each panel, desired reading order, and a user-provided library of preferred page layout style guidelines.

1 Introduction

This poster paper describes work in progress to implement an automated interactive assistant for arranging and sizing comic book panels to follow conventions of narrative duration and reading order. In comic conventions, larger panels tend to hold the reader's attention for a longer time. Larger panels can also suggest a longer duration of elapsed narrative time or a slower pace of narrative action [1, 5]. To use this automated comic book layout assistant, an artist inputs a sequence of images for the artwork and balloons in each panel. Some artists, such as Rivka (alias), designate the pacing of a page by a *timing string* for example, "DBBAABB." The duration of each panel ranges from 'A' to 'E', with 'A' being the shortest and 'E' being the longest. The artist provides a library of pre-built page layout templates which represent desirable layouts and pacing patterns. The artist also specifies the reading order as left-to-right or right-to-left. For western readers, panels are read moving left-to-right then top-to-bottom. Japanese comics are read moving right-to-left then top-to-bottom. Given these inputs, the automated assistant sizes each panel to correspond to its respective duration and places the panels, whenever possible, to conform to the specified layout templates having similar timing strings. The artist can then flip through the pages of the comic to inspect the layout. The artist can modify the duration of one or more panels and invoke the assistant to re-compute the layout.

2 Related Work

Recent works in automated generation of comics have summarized digital video [7], online chat transcripts [3], and events in virtual worlds [6]. Lok and Feiner summarize constraint- and learning-based automated layout systems [4]. This algorithm utilizes

implied low-level constraints of panel size, fit within page, and non overlap of panels. Jacobs et al compute layouts of text and figures using dynamically adaptable grid layout templates to adapt to different display devices [2]. This work shares the idea of using flexible templates, but addresses the sub-problem of comics. Uchihashi et al. size keyframe panels by video duration and pack panels into a grid using an exhaustive row-wise search, but their work is not directed at producing layouts directed to specified style guides [7].

3 Panel Layout Library

The artist begins by using the layout editor to construct a library of page layout templates to provide examples of his or her style (Figure 1a). The set of templates is organized into an index structure that will be used to retrieve layout templates that most closely match the desired timing string of the input sequence of artwork panels.

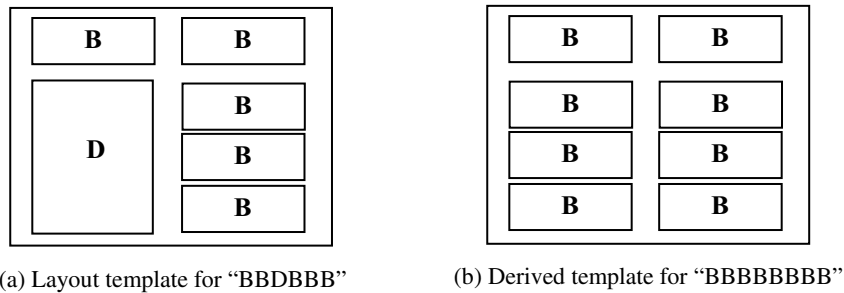


Fig. 1. Example of layout templates

The list of rectangular panels for a page layout template is automatically converted into a recursive structure called a `GuidePanelLayout (GPL)` that contains a list of elements. Each element is either a panel or another `GuidePanelLayout`. Each GPL specifies its orientation (vertical or horizontal), its bounding rectangle, and its *timing string*. Assuming a left-to-right reading order, the timing string for the panels shown in Figure 1a is "BBDBBB." The recursive GPL structure for the above panels is:

```
GPL-Vertical { GPL-Horizontal { B, B },
               GPL-Horizontal { GPL-Vertical { D }
                               GPL-Vertical { B, B, B } } }
```

The assistant automatically recognizes child elements, e.g., `GPL-Vertical { B, B, B }`, as first-class GPL's enabling it to derive additional templates for the library (Figure 1b).

4 Page Layout Algorithm

The automated layout assistant first tries to find a page layout template or a child element whose timing string exactly matches the timing pattern of the next unprocessed

timing substring of the given input panels. Whenever a match is found, the sequence of input panels adopts the layout specified by the page layout template or child element. Otherwise, the automated layout assistant applies a recursive backtracking constraint-satisfaction problem (CSP) search to find a workable layout. The constraint problem is formulated such that there is a variable to determine how each panel is arranged relative to its immediate predecessor in the specified reading order. For a left-to-right reading order, the next panel may be stationed ‘horizontally’ that is to the right of and as far up as possible (without hitting another panel or the edge of the page) from the previous panel. Or, the next panel may be positioned ‘vertically’ that is below and as far to the left as possible from the previous panel. Hence, the domain of each variable consists of the values ‘horizontal’ or ‘vertical’. Panel placements that would fall outside of the page are rejected. Backtrack if neither ‘horizontal’ nor ‘vertical’ placements are possible. If both placements are possible, then proceed with the one that leaves the least amount of empty space in the bounding rectangle of the panels placed so far. Once no more panels can be placed on a page, then output the set of panels for one page and repeat the process.

5 Implementation

The automated layout assistant and page layout template editor are coded in Java. In one test case, a sequence of 224 panels repeats a timing string of “DBBAABB.” The layout for these 224 panels was formatted in 32 pages and was computed in 26 milliseconds (10.4 seconds over 400 repetitions) on a 2 GHz Intel Core2 Duo iMac running Mac OS X 10.4.1. A screen capture of the panel layout for one of these pages is shown in Figure 2.

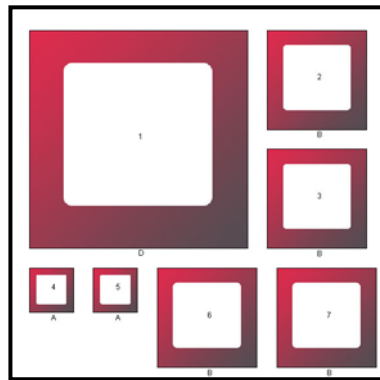


Fig. 2. Computed panel layout for one of the 32 pages in the test case

6 Conclusions and Future Work

The assistant can compute multi-page layouts for hundreds of panels at interactive rates. More work is needed to automatically generate GPL’s in the style of a small

number of artist-specified samples. The algorithm too often defaults to backtracking when a given timing string does not match a template. For example, variations in panel aspect ratio make it difficult to find matches. We plan more work to better align panel edges over the whole page, adjust panel sizes to utilize empty page space, and apply stylistic variations such as partially overlapping panels. We plan to solicit hands-on feedback from comic layout artists to improve the usability of the interactive layout assistant.

Acknowledgements

Thanks to Millsaps Computer Science majors Franklin Landry and Tyler Moss. Franklin designed a set of layout templates inspired by manga and is completing a comic editor application. Tyler contributed Java code to implement backtracking CSP search.

References

1. Eisner, W.: *Comics and Sequential Art*. Poorhouse Press, Paramus (1985)
2. Jacobs, C., Li, W., Schrier, E., Barger, D., Salesin, D.: Adaptive Grid-Based Document Layout. *ACM Transactions on Graphics* 22(3); *Proceedings of ACM SIGGRAPH 2003*, July 2003, pp. 838–847 (2003)
3. Kurlander, D., Skelly, T., Salesin, D.: Comic Chat. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 1996*, New Orleans, LA, August 4-9, 1996, pp. 225–236 (1996)
4. Lok, S., Feiner, S.: A Survey of Automated Layout Techniques for Information Presentations. In: *Proceedings of Smart Graphics 2001*, Hawthorne NY, pp. 61–68 (2001)
5. Rivkah.: Paneling, Pacing, and Layout in Comics and Manga, <http://lilrivkah.livejournal.com/168859.html>
6. Shamir, A., Rubinstein, M., Levinboim, T.: Generating Comics from 3D Interactive Computer Graphics. *IEEE Computer Graphics & Applications* 26(3), 53–61 (2006)
7. Uchihashi, S., Foote, J., Girgensohn, A., Boreczky, J.: Video Manga: Generating Semantically Meaningful Video Summaries. In: *Proceedings of ACM Multimedia*, pp. 383–392 (1999)