Adaptive Layout for Interactive Documents

Kamran Ali,[1] Knut Hartmann,[2] Georg Fuchs,[1] and Heidrun Schumann[1]

[1] Institute of Computer Graphics
Department of Computer Science, University of Rostock
Albert Einstein Str. 21, D-18059 Rostock, Germany
{kamran.ali,georg.fuchs,schumann}@informatik.uni-rostock.de

[2] University of Applied Sciences Flensburg
Kanzleistr. 91–93, D-24943 Flensburg, Germany
knut.hartmann@fh-flensburg.de

Abstract. In many application domains there is a strong need to produce content both for traditional print media and for interactive media. In order to fully benefit from digital devices, online documents must provide mechanisms to support interactivity and for the personalization of content. Thus, powerful authoring tools as well as flexible layout techniques are needed to display dynamic information effectively. In this paper we present a novel approach to template-based design of complex non-grid layouts and propose a unique combination of constraint-based and force-based layout mechanisms. Our authoring tool enables designer to specify designs pattern with overlapping content elements that are transformed into layout constraints and force systems by the layout engine. The smooth integration of interactive illustrations into complex layouts also induces constraints to achieve temporal coherent layout. We propose algorithms to achieve graceful layouts even when the space consumption of content elements significantly differs from the defaults specified in layout templates.

1 Introduction

Rapid growth of internet and advanced web technologies has seen large volumes of information being delivered over web. In digital networked world content can be frequently updated and individuals may request information in different forms at different devices and have their own viewing preferences. Therefore dynamic requirements of the digital media restrict that the documents cannot be formatted in advance. Hence, the traditional processes of content creation, editing, and quality assurance have to be completely revised in digital environments.

The reuse of printed material (e. g., text books, encyclopedia, or technical documentation) in interactive applications is very promising due to the high quality of their content. But layout techniques in current web browsers are limited in scope and quality. Besides, more often that not information recipients are going to be many individual users who are anonymous and connected to different devices. Variations in display resolution and computing power of electronic devices may require that the document is presented differently for each reader. Under this scenario, static design of a document might become unsatisfactory when the content or device capabilities change.

The situation becomes even harder for layouts with many illustrations. While web browsers guarantee the readability of text on all devices and offer at least some room to interact with text, illustrations are inflexible elements in the layout. *Interactive illustrations* on the other side would offer new possibilities for online documents. 2D and 3D visualizations can be overlayed with secondary elements (e. g., textual annotations, anchor points, connecting lines, scales, legends etc.). In addition, an unequal scaling of imagery and secondary elements can guarantee the readability of textual annotations and updation of content within textual annotations.

In this paper, we propose tools that (i) enable authors / designers to create blueprints for complex layouts that integrate interactive illustrations and (ii) describe several layout techniques to render documents with dynamic content. This paper is organized as follows: Sec. 2 reviews the related work. We present insights into our novel approach to adaptive document layout in Sec. 3. Sec. 4 gives a case study. Finally, Sec. 5 summarizes the initial results of our work and discusses directions of future work.

2 Related Work

Earlier research on document layout mainly focused on line-breaking and pagination. Commercial desktop publishing systems also provide a set of pre-designed layout templates. But these systems aim at static documents and do not provide any mechanism to author or layout documents with dynamic content. In contrast, the render engines of current web browsers were designed to re-layout the document's content dynamically. But there are still few mechanisms for authors to adjust the design of the layout to the capabilities of different devices.

Current research in the field of automated document layout has mainly explored graphical constraints and typographic grids.

Constraints: Weitzmann's system [16] and Graf's LayLab [10] use grammars to specify constraints of two types: abstract constraints (e. g., description-of, author-of etc.) and spatial constraints (e. g., right-of, top-aligned etc.). Spatial constraints can be directly used as input to the constraint solver. Abstract constraints must be converted to spatial constraints before the layout resolution might take place.

As powerful and expressive as these grammars are, it becomes extremely difficult to describe a set of all possible high-level relationships between the items of a presentation. But the major problem of this technique is the treatment of contradicting constraints that prevents any solution.

Grids: Display area of the presentation is split into an array of upright rectangular cells separated by margin space [14]. Each layout element has to span an integral of cells in width and height. Hence, the underlying typographic grid works as a proportional regulator for content items. Feiner's approach [6] imposes spatial constraints on the layout items via a typographic grid to generate presentation. The underlying grid is generated automatically from document's content and viewing restrictions.

By encoding the desired properties of a layout in a set of layout templates, Jacobs et. al. [11] were able to adjust the layout of online documents to the capabilities of different devices. Each layout template specifies abstract and spatial constraints on the components of the presentation. Several techniques are applied to select and

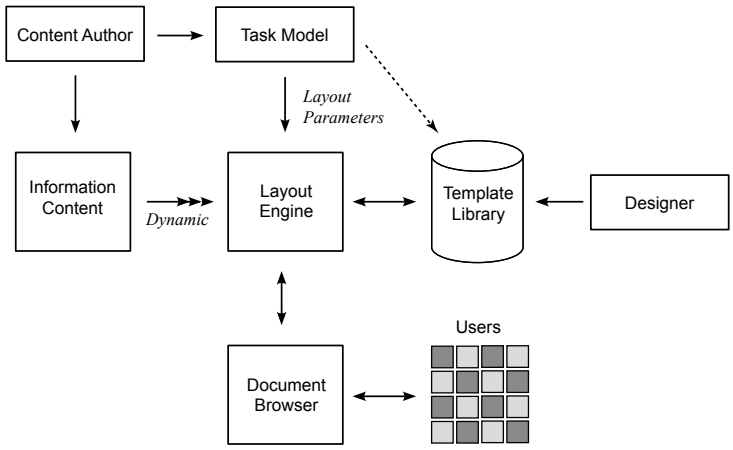

Fig. 1. Adaptive document system design.

adapt layout templates from the set of available templates. Their system also performs pagination to split the document's content on multiple pages that in turn are formatted separately. Even though this approach achieves impressive results for static content, it was not designed for interactive media elements. Moreover, the strict-grid based design may not be suitable for documents containing large number of *cut-out* illustrations.

3 Adaptive Document Layout

3.1 Architecture Overview

Figure 1 shows the architecture of our adaptive document system. We provide a template authoring tool that the designers can use to design document templates for various viewing scenarios (e. g., different screen aspect ratios or device classes). Designers can define layout elements by drawing rectangular boxes or, for complex non-grid layouts, arbitrary shaped convex polygons on the screen. The element placement is controlled using linear arithmetic equalities and inequalities constraints. For each template, designers also dictate when the design is appropriate by specifying which content must be present and under which viewing conditions the template can be used. The templates are stored in a *template library*.

Content authors prepare the document contents by tagging individual content items. These include text, static illustrations, animations, and interactive 3D representations. Items can be tagged with their importance values. The authors may also define item priorities which are considered to resolve space requirements on screen.

During the layout process, the *layout engine* determines the appropriate set of templates in the template library by matching content tags to the template item tags. For the selected templates, it adjusts the size and position for each layout element using

constraint solving methods. The constraint-based layout is then further refined in a post-processing phase. We discuss this most important part of our approach in Section 3.2.

3.2 Layout Engine

Document layout phase is divided into two stages. In the first stage an initial layout is derived from templates using the given constraints. The second stage is post-processing stage; forced-directed methods are applied on selective regions to modify the initial layout. During the latter phase, a check-mechanism is enabled so that layout adjustments are made only when they don't conflict with template constraints.

Initial Layout: Constraint-based document layout technique [3] are used to compute the initial layout configuration. Layout engine loads templates from the template library and evaluates them. It matches the template items to the given content and computes how well this content fits within the template. For the selected template, size and position variables of each element in the template are set by the layout engine. These values are fed into the constraint solver for resolving a given set of constraints. In our system we use CASSOWARY TOOLKIT [2] which is an incremental linear arithmetic constraint solver. Since the constraint-based layouts are a well explored area, in this paper we focus our attention on the forced-directed layout methods.

Post-processing: The post-processing phase is used to modify the initial configuration set by constraint-based layout. It is especially helpful to create complex non-grid layouts for varying amount of illustrative material which cannot be laid out using the given template. We explore force-directed layout techniques to provide graceful layout mechanism even when the content type and size is significantly different from what is suitable for the given template.

We formulate document layout as a *graph-layout* problem. A graph $G = (V,E)$ consists of vertices V and a set of edges E where each edge connects a pair of vertices. Fruchterman and Reingold [7] proposed spring-embedding approach for drawing graphs [4]. According to this, an attractive force enables the vertices to attract each other if they are edge-connected, thus keeping them close together. A repulsive force causes the vertices to repel each other, thus avoiding their overlapping.

Let a graph G represent a document. Vertices denote content items, typographic guidelines, and area boundary lines. Edges are used to model relationships between vertices. For example an edge can model dependency between two content items so that they are placed closer to each other. An edge between a typographic grid line and a content item can be used to indicate the alignment of item. Similarly an item can be attached to its reference position using an edge. In contrast, a repulsive force would ensure that the items mind a certain distance between them.

Types of forces: In our force-directed approach, several forces are applied on objects:

- An attractive force between related elements; *(derived from item dependency list)*

- attractive force between the current and reference placement of an object, *(derived using the template)*
- repulsive force between elements; *(derived from margin and gutter)*
- repulsive force at the page/screen boundary; *(derived from page margin)*
- attractive force between previous and current positions of an object in successive layouts.

If f_a and f_r are the attractive and repulsive forces, respectively, with d distance between the objects, then $f_a(d) = d^2/k$ and $f_r(d) = -k^2/d$. Here k is the ideal distance between two objects where attractive and repulsive forces cancel each other. The magnitude of these forces is dependent on the distance between the objects. Minimum distance between the two objects is determined by anti-podal pair between the convex-shaped polygons. The layout engine determines attractive and repulsive forces on each object i in the layout. These forces are summed together to calculate the magnitude and direction of displacement for i. At run-time content elements are attracted toward their old positions in the previous layout configuration. This allows us to achieve temporal coherence over the successive layouts.

A pressure-directed part controls different object sizes to objects. According to Boyle's law, volume of the gas increases when the pressure decreases at constant temperature and vice versa, i. e., the volume decreases when the pressure is increased. This relationship between pressure P and volume V is given as $P \times V = constant$.

To control the resizing of elements in layout, we incorporate *Inner* and *outer* pressure forces in our system with a modification to the approach mentioned in [12].

- The more relevant a content object, the higher the inner pressure, and vice versa;
- The more free space on the screen, the lower the outer pressure, and vice versa;

More pressure inside an element than the outer pressure leads to an enlargement of the element and decrease in inner pressure and to rise of outer pressure. Conversely more outer pressure than the inner pressure leads to the reduction of object. With this mechanism free space left by missing content items can be utilized by the other items. Items can be added/removed and resized freely on the screen as the pressure-directed method manages space requirements. This paves the way to integrate adaptive illustrations into complex layouts which impose and obey global layout requirements.

Our force-directed layout approach is given in Algorithm 1. It extends force-based algorithms in [7] used for graph drawing and [12] intended for placement of rectangular windows. In contrast, we apply force-directed algorithms for document layout. Our approach analyzes template constraints to infer the system of forces. Elements can be arbitrary shaped convex polygons overlapping each other. Moreover large amounts of content can be split into several pages that are laid out individually.

Using template authoring tool designers can define the size of items and specify if resizing is allowed by the pressure-directed forces. They can also declare a set of constraints that the post-processing must obey in the final layout. Initial layout sets the system of pressure forces in balance which are then manipulated at run-time. Since the pressure-directed resizing scheme goes hand-in-hand with force-directed displacement scheme, layout conflicts are resolved dynamically. Layout modifications in the post-processing step are applied only when they don't lead to an invalid layout regarding the constraints.

Algorithm 1 Force Directed Layout

```
{determine initial placement using templates}
for i = 1 to iterations do
    {force-directed part}
    for v ∈ V do
        {calculate repulsive force f_r and attractive forces f_a upon each element v}
        {move element v in the direction of f_r + f_a}
        {update forces}
    end for
    {pressure-directed part}
    for v ∈ V do
        if v.pressure_inner > v.pressure_outer then
            enlarge element v
        else
            reduce element v in size
        end if
        update pressure values
    end for
end for
```

Once the page layout is computed, content can be flowed into each determined region in the layout. The formatted document is presented via document browser to the users who can interact with it. For larger amount of content, we need a *paginator* to split the content into a set of discrete pages, subject to various constraint. Optimal pagination is a complex issue and begs more attention. Due to limited space in this paper we refer the readers to study pagination task in [15].

Our layout engine also incorporates an integrated illustration system to support dynamic illustrations in documents. Efficient dynamic labeling algorithms [1] and [8] compute placement of textual annotations on images. The labeled illustrations are automatically adapted to fit within 2D regions allocated to them by the layout engine.

3.3 Task-Driven Document Layout

Up to now, we only considered aspects regarding the layout of individual dynamic documents. However, their suitability as a means to effectively and concisely communicate complex structure make dynamic documents a primary choice for online documentation such as technical manuals for mobile maintenance support ("e-Manuals") [8,9]. Tasks associated with repair and maintenance work typically follow laid down procedures, which lend themselves to explicit notation in the form of a *task model* [13]. Generally, a task model forms a hierarchy of compound tasks with subtasks and conditional/temporal relationships between them.

This task model can be used to store additional information, to guide and optimize the initial layout of dynamic documents with respect to the current task at hand (cf. Fig. 1). To this end, the content author specifies, for each task, which template from the library is preferable for the content associated with that task. Moreover, she can specify different priorities per subtask for content items that will affect how the content

is laid out (cf. Sect. 3.2). This enables the author to specify which aspect(s) of the document are most important for the current working step. Note that after the task-driven generation of *initial* document views, interaction by the user still allows for dynamic layout changes as described in Section 3.2.

4 Case Study

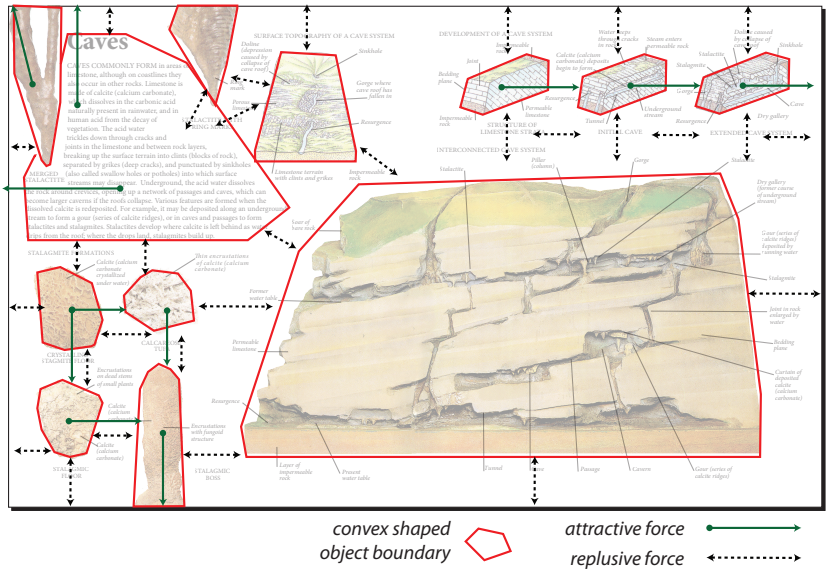

Fig. 2. Forced based layout.

Figure 2 shows the results of our forced-directed layout approach overlayed with the forces used to compute the layout. The content has been laid out in the style of a VISUAL DICTIONARY book [5]. In order to match the content of the descriptive text and illustrations, all figures are richly annotated with text labels.

Note also, that the bounding rectangles for the primary visual elements and their associated secondary elements overlap. The layout was achieved by (i) placing all primary elements at reference positions in template and deciding for appropriate scaling factor for the individual images or image groups and (ii) placing all the remaining secondary elements. Thus, secondary elements of all depictions share the unused background as a common resource. This layout procedure is based on semantic considerations (relevance of layout elements), from structural constraints (common properties for all elements of a group) and the difficulty to reposition or to adjust an layout element. Textual annotations, for example, *can* easily float and the line-break of the descriptive text can be adjusted in order to float around the thumbnails images.

5 Conclusion and Future Work

This paper proposes novel strategies to adapt the layout of multimodal documents under different viewing conditions according to the changes in content and user interaction. There is still room for many improvements. First, our algorithm should exploit more parameters of the layout: font sizes, line width, column size, or the grid size. Moreover, we believe that this freedom to adjust all parameters that determine the overall layout is the only way to convert aesthetics and readability criteria into metrics, forces, and constraints. Another interesting aspect is the automatic determination of layout parameters or layout templates from scanned documents. This would ease the time-consuming and error-prone trial&error process of fiddling around with layout parameter to providing a set of good examples. Of course, proper evaluations are needed both for the authoring tool as well as for the dynamic layout algorithms.

References

1. K. Ali, K. Hartmann, and T. Strothotte. Label Layout for Interactive 3D Illustrations. *Journal Of The WSCG*, 13:1–8, 2005.
2. G. J. Badros, A. Borning, and P. J. Stuckey. The Cassowary Linear Arithmetic Constraint Solving Algorithm. *ACM Trans. Comput.-Hum. Interact.*, 8(4):267–306, 2001.
3. A. Borning, R. Lin, and K. Marriott. Constraint-Based Document Layout for the Web. *Multimedia Systems*, 8(3):177–189, 2000.
4. G. Di Battista, P. Eades, R. Tamassia, , and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
5. P. Docherty, editor. *DK Ultimate Visual Dictionary*. Dorling Kindersley Publishing, 2002.
6. S. K. Feiner. A Grid-Based Approach to Automating Display Layout. In *Graphics Interface '88*, pages 192–197, 1988.
7. T. M. J. Fruchterman and E. M. Reingold. Graph Drawing by Force-Directed Placement. *Software — Practice and Experience*, 21(11):1129–1164, 1991.
8. G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, H. Schumann, and T. Strothotte. Adaptive Labeling for Interactive Mobile Information Systems. In *10th Int. Conf. on Information Visualisation*, 2006.
9. G. Fuchs, D. Reichart, H. Schumann, and F. P. Maintenance Support — Case Study for a Multimodal Mobile User Interface. In *Multimedia on Mobile Devices II*, volume 6074 of *SPIE*, 2006.
10. W. H. Graf. Constraint-Based Graphical Layout of Multimodal Presentations. In *Int. WS on Advaned Visual Interfaces*, pages 365–385, 1992.
11. C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin. Adaptive Grid-Based Document Layout. *ACM Transactions on Graphics*, 22(3):838–847, 2003.
12. P. Lüders, R. Ernst, and S. Stille. An Approach to Automatic Display Layout Using Combinatorial Optimization Algorithms. *Soft. — Prac. & Exp.*, 25(11):1183–1202, 1995.
13. G. Mori, F. Paterno, and C. Santoro. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Trans. Soft. Eng.*, 28(8):797–813, 2002.
14. J. Müler-Brockman. *Grid Systems in Graphic Design*. Verlag Arthur Niggli, 1996.
15. M. F. Plass. *Optimal Pagination Techniques for Automatic Typesetting Systems*. PhD thesis, Department of Computer Science, Stanford University, 1981.
16. L. Weitzmann and K. Wittenburg. Grammar-Based Articulation for Multimedia Document Design. *Multimedia Systems*, 4(3):99–111, 1996.