LayoutExtensions

Layout problems

Empty areas and vertical justification

In order to satisfy keep condition and avoid the creation of widows and orphans, page breaking algorithms often create partially-filled pages. The height of the empty area can vary from less than one line-height to several line-heights.

`http://people.apache.org/~lfurini/wiki/widow.png`
(empty area created to avoid a widow)

`http://people.apache.org/~lfurini/wiki/keep.png`
(empty area created to satisfy a keep between titles and the first paragraph)

At the moment, the existing properties does not allow the user to control these empty areas.

Setting display-align for the body-region, it is possible to decide if the content areas should be stacked at the top, at the bottom, or in the middle of the page: this property has the effect of placing the empy area at the bottom and / or at the top of the page, but it does not affect its height.

The property display-align affects the vertical positioning of areas inside pages just as text-align affects the horizontal positioning of areas inside lines: the existing values for display-align "before", "center" and "after" correspond respectively to "start", "center" and "end" alignment.

And what about justified alignment? Horizontal justification involves the adjustment of word spaces and letter spaces; vertical justification could similarly be achieved adjusting spaces between paragraphs and between lines: the empty space would be distributed between the areas placed in a page. But there are other parameters that could be modified in order to create exactly-filled pages: for example, it is possible to layout paragraphs using fewer or more lines by reducing or augmenting text spacing, thus avoiding the creation of both widows / orphans and empty areas.

Aligned lines

A document is usually composed of "normal" lines of text and some other objects (titles, images, ...) with different heights; a common request for book-style documents is the alignment of normal lines between facing pages.

At the moment there isn't a specific property which could be use to espress this additional request: the only way to achieve this alignment is using space-before and space-after so that the overall height of each object is a multiple of the normal line height. This can be quite tricky, because the needed spaces depends on the object height (which could be unknown) and its position in the page.

`http://people.apache.org/~lfurini/wiki/bpunit.png`

Proposed extensions

Here is a set of extensions specifically designed in order to solve these layout problems.

Two more values for display-align

- "distribute": the areas placed in the page must be vertically distributed, so that the before-edge of the allocation-rectangle of the first child area is placed coincident with the before-edge of the content-rectangle of the reference-area, and the after-edge of the allocation-rectangle of the last child area is placed coincident with the after-edge of the content-rectangle of the reference-area
- "fill": the areas placed in the page must be modified so that they exactly fill the page; another property is needed to list the properties that can be modified

A new property called fill-by-modifying: it contains the list of properties that can be modified in order to fill the page.

A new property called block-progression-unit: it defines the unit of measure for the height of each area generated by the formatting object; in other words, if a block has block-progression-unit = "15pt" the height of each area generated by that block must be a multiple of 15pt.

Modifiable properties

Some properties can have .minimum, .optimum and .maximum components, such defining a range in which the actual value can be chosen. These properties can affect the block progression dimension of areas, either directly (for example space-before) or indirectly (for example word-spacing).

Other properties cannot have a range value at the moment, but they could be quite easily extended.

The page breaking algorithm could choose appropriate actual values inside these ranges in order to fill the pages.

Spaces

The properties space-before and space-after can have a range value; spaces before and after the blocks can be easily adjusted in order to fill the page.

The default value for these properties is 0, so this adjustment cannot be done unless the user explicitly allows it using range values instead of fixed ones. Moreover, space adjustment is incompatible with aligned lines.

Text spacing

Using reduced or expanded word spacing and / or letter spacing, a paragraph could generate fewer or more lines.

The properties word-spacing and letter-spacing can have a range value, and their default value allows an application dependent adjustment. If the document is well designed and the region-body height is a multiple of a "normal" line height, spacing adjustment can be used to fill the pages without losing the alignment of lines between facing pages.

Margins

Using reduced or expanded margins, each paragraph in a page could generate fewer or more lines.

This kind of adjustment could be used to achieve two different goals:

- optimize a page width according to the lines it contains: if each line in the page would need large word spaces, larger margins allow to use smaller spaces;
- find alternative layouts for paragraphs: sometimes spacing adjustment does not effect the quantity of lines generated by a paragraph, while a small change in margins could.

At the moment, margin-left and margin-right cannot have a range value.

Other modifiable properties

Other properties that could be modified:

- padding-before and padding-after (similar to spaces)
- padding-start and padding-end, start-indent and end-indent (similar to margins, but involving fewer blocks)
- font-stretch (similar to spacing)
- ...

Implementation details

Block level formatting objects, where block-progression-unit and / or other modifiable properties are set, must be represented using Knuth's formalism, in terms of boxes, penalties and glues. This way the page breaking algorithm will be able to automatically decide which properties must be adjusted, and how.

Elastic paragraphs

First-fit and best-fit line breaking algorithms build one line at a time, taking decisions with no lookahead (or with a small one).

Knuth's total-fit algorithm gets a whole paragraph and decides all line breaks at once; if there are several options (more than one node in the active list) it chooses the "best" one from the whole paragraph's point of view.

If there is more than one node, and they are all taken into account, it is possible to delay the choice, and to choose the best layout after the page breaking, having a wider view of the document.

For example: let's pretend that the active nodes show that a paragraph could generate 4 or 5 lines, and the 5-lines layout has fewest demerits.

If the line breaking algorithm chooses this layout and ignores the other one, when performing page breaking we could find that the first page in which this paragraph must be placed can contain only 4 lines; as we cannot put a single line at the beginning of the following page (because of the widows property), the page breaking algorithm will move 2 lines (the 4th and the 5th ones) to the new page, leaving an empty area at the bottom of the previous page.

Whereas, if both layout are taken into account, the page breaking algorithm could choose to create only 4 lines for that paragraph, such filling the page. The layout with 4 lines becomes the best one as soon as we have a wider perspective.

Layout selection rules

... coming soon ...

Sequence creation algorithm

See MultiLayoutSequence.

Adjustment negotiation

... coming soon ...

Using a different line width

... coming soon ...

Implementation of block-progression-unit

... coming soon ...

Examples

Here is an example of the improvements in page layout that can be achieved using these extensions.

This is a long xml file: it contains 19 chapters made of a title and many paragraphs of text.

http://people.apache.org/~lfurini/wiki/book.xml

This is a simple stylesheet using only standard properties: each paragraph creates a `fo:block` with `orphans="2" widows="2"`; there are no `keep-*` condition but the one between a title and the first paragraph; the `region-body` can contain 32 normal lines of text. The `region-body` has `background-color="lightgrey"`, while blocks have `background-color="white"`, so empty areas are coloured.

http://people.apache.org/~lfurini/wiki/book2fo.standard.xsl

The pdf output is this:

http://people.apache.org/~lfurini/wiki/book.standard.pdf

There are 185 pages. If we don't count the 11 pages empty because of break conditions, and the last page of each chapter (that could rightly be partially empty), the remaining 154 pages should be filled with lines: 61 (40%) have an empty area.

This is the same stylesheet plus the extensions `display-align="fill" fill-by-modifying="margins; spacing"`.

http://people.apache.org/~lfurini/wiki/book2fo.extensions.xsl

And this is the pdf output:

http://people.apache.org/~lfurini/wiki/book.extensions.pdf

There are 183 page, and 10 are empty because of break conditions; there is only 1 page out of 153 with an empty area (because of a bug, I suspect).