The box-glue-penalty algebra of TEX and its use of \prevdepth

Frank Mittelbach

Contents

Abstract

This article discusses certain aspects of TEX's approach to line breaking and its consequences for automatically calculating the right amount of vertical space between lines in more complex layouts.

It starts with giving a short introduction to the box-glue-penalty algebra used by TEX to model material to typeset. We then look at how the program calculates the vertical glue between lines in which the parameter \prevdepth plays a crucial role. Next we examine different types of output routines and evaluate how and to what extent the TEX algorithms can accommodate their goals.

The final conclusion is that this is an area where we can pose problems that cannot be resolved using current TEX, ε-TEX, pdfTEX, or XETEX, unless you restrict the allowable input, as there is no way to obtain some of the information used by TEX's algorithms for later manipulation of the result.

Like the answer to many questions these days, the situation is (probably) different with LuaTEX — probably, because I haven't actually tried it, but given the additional possibilities offered by LuaTEX a solution should be feasible.

1 The box-glue-penalty algebra

TEX's typesetting is built around a model that is known as the box-glue-penalty algebra [1]. At the lowest level we have (character) boxes that have (as far as TEX is concerned) no inner structure. These (character) boxes intermixed with glue (representing spaces) and penalties (representing possible break points) are what TEX combines to form higher level objects and eventually build up pages.

The first level of construction is called a horizontal list and such a list can either form a new box of its own (a horizontal box also known as an \hbox) or it can be passed to the paragraph builder that then (using a large number of parameters) will break the list apart into sub-lists (possibly dropping some content at the break points) to form the individual lines (again \hboxes) of a paragraph.

It is also possible to build vertically oriented lists, again consisting of boxes, glue and penalties. Here the glue represents vertical spaces and any penalties guide splitting the list later on. Such a list can become a box of its own (a vertical box or \vbox) or it can simply form a "galley" from which, by some method, TEX once in a while chops off a certain amount to form the content for a page to be produced.

Boxes in vertical lists are different though: while all lists can contain explicitly or implicitly constructed boxes, only horizontal lists can contain character boxes. If, while constructing a vertical list, TEX encounters a character box it puts the current construction on hold and starts building a horizontal list. If it then encounters a \par command (or an empty line) it will pass the constructed horizontal list to the paragraph builder. That in turn chops it up into individual lines and returns those as horizontal boxes (paragraph lines) intermixed with glue and penalties. These are then added into the vertical list and construction of the vertical list continues.

2 Splitting lists

Splitting of horizontal lists can only be done by passing the list to the paragraph builder. The result in that case is a vertical list (or rather something that will become part of a vertical list) and it contains a varying number of horizontal boxes forming the lines of the paragraph. In other words it is not possible directly to take a horizontal list (or box) and split it into two horizontal lists.

In the case of vertical lists the situation is slightly different: on the so-called "main vertical list" on which the material for pages is gathered, TEX monitors the amount of material being gathered and at certain points, either directed by some explicit penalty or simply because it decided that there is now enough material, it will chop off the right amount of material for a single page and then fire up a subroutine called the output routine to process that material and build a final page from it.

In addition to that, TEX offers the possibility to split off a chunk of a specified size from a given vertical box and place it into a new box. Technically speaking this is more or less what the output routine process on the main vertical list does when it gets fired up. The only difference is that on boxes this is an explicit command that needs to be invoked in the

programming code and it operates only on explicit boxes formed earlier.

It would be interesting to have the same functionality on the program level for horizontal boxes but for some reason that never made it into the program.

In other respects horizontal and vertical splitting is a very similar operation (on the box-glue-penalty algebra level). Splits can only happen at explicit penalties or at the left of glue provided it is immediately preceded by a box.[1] So in case of two globs of glue directly next to each other, a split can only happen before the first glue. Consecutive penalties behave like a single penalty unless they both force a break and we are in a horizontal list.[2]

Once a break point is chosen TEX drops all glue and penalties following it until it comes to the next box. The rationale behind this is that something like white space between words should vanish if you have a line break, and so should white space between lines if you have a page break.

This rather simple model allows to define surprisingly complex behavior, simply by specifying cleverly constructed sequences of glue, penalties and (empty) boxes. Appendix D in *The TEXbook* [2] shows a number of examples.

In summary, the box-glue-penalty model proved to be an ingenious way to model typesetting requirements and although it is not fully orthogonal and perhaps misses one or the other feature that would be useful, it gets the job done in a concise manner, and it is fair to say that even after more than three decades nobody has come up with anything better.

3 Assembling a vertical box or galley

A box in TEX terms is described by three dimensions: its height, depth, and width. The rationale is that characters that form words are lined up horizontally, each having a certain height but some of them stick out below the imaginary line (known as the baseline) and thus have a depth. Consequently, the boxes have a reference point at its left side with the material above the reference point forming the height of the box, and the material below, the depth.

If a horizontal list is used to form a new box then the inner boxes are lined up on their reference points, glue between the boxes appears as spaces, and the height and depth of the resulting box will be the maximum height and depth of the inner ones.[3] The width of the new box is simply the sum of all widths including the amounts taken up by the glue items. The reference point of the newly created box is then again on its left side at the baseline.

The situation with vertical lists is similar: they too align the boxes inside on their reference points (only this time vertically) and glue between boxes becomes spaces in the vertical direction. The width of the newly formed box is then the maximum width of its inner boxes. The calculation of height and depth, however, is slightly more complicated. By default, the depth will be the depth of the last box inside but only if this box is not followed by glue (a penalty would be allowed) — in the latter case the depth would be zero. The height is then calculated as the sum of all the heights and depths of all boxes plus the spaces and minus the box depth that was calculated earlier. In short you will end up with a box that has a very large height and a (normally) small depth.

In fact the depth of the new box is further restricted by a parameter called `\boxmaxdepth`: if the calculated depth exceeds this value then the reference point of the constructed box is lowered until the depth is no longer in violation. By default the value of this parameter is the largest possible dimension so that the restriction doesn't apply for manually created boxes unless this is changed.

On the main vertical list `\maxdepth` is used for the same purpose. If TEX decides that material needs to be packaged into a box to be passed to the output routine it uses that parameter to determine the maximum depth and thus the height of that box. In contrast to `\boxmaxdepth` this parameter typically has a setting that allows only for small depths to ensure that material is not "falling off the page" if it has an unusually large depth.

As an alternative to the construction explained above TEX also supports building a vertical box whose reference point aligns with the first box inside (`\vtop`), i.e., its height will be the height of this first box and the depth will hold everything else. If the list starts out with glue then the height of such a box will be zero.

[1] In real life the situation is, of course, more complex. TEX, for example, understands how to hyphenate words and so during the paragraph breaking it might introduce additional break points (and possibly even additional characters or variations into a list of character boxes), but abstractly one can think of this as just another version of boxes, glue and penalties that have been present from the beginning.

[2] The value of a penalty describes the desirability to break at a certain point (the smaller the better). The anomaly that two forcing penalties in horizontal mode behave differently and produce two breaks is due to some implementation decisions.

[3] As there are some operations to lower or raise boxes with respect to their reference point, this is not quite accurate, but one could consider such a manipulation simply as an operation that forms a new box with new dimensions.

4 Calculation of vertical glue

In the previous sections we explained how lists in the box-glue-penalty algebra are turned into new boxes. We also mentioned that the paragraph builder splits a horizontal list into a sequence of box-glue-penalty items but so far we haven't explained how this precisely happens and what kind of glue items are constructed during this process.

The main goal in paragraph building is to form lines of text that are (normally) vertically positioned in a way that the distance from baseline to baseline is constant. TEX manages this in the following way: whenever it builds a vertical list it keeps track of the depth of the last box appended to the list in a parameter called `\prevdepth`. At the beginning of a list it has a sentinel value of `-1000pt` to indicate that no box has been added so far.

When the paragraph builder builds a line box this box will have a certain height and depth. TEX then calculates the glue to be placed before the box by using the parameter that holds the standard baseline to baseline distance (`\baselineskip`) and subtracts from it the height of this box and the current value of `\prevdepth`. The resulting value is then appended as a glue item unless it is smaller than `\lineskiplimit` (i.e., the box height or the previous depth was very large). In that case TEX simply appends a glue item with a fixed value defined by another parameter called `\lineskip`.[4] The `\prevdepth` is then updated to hold the depth of the box just appended so that the calculation for the next line will be correct.

When a box is manually added to a vertical list, e.g., via `\box\mybox`, the same happens i.e., baseline glue gets prepended to the box and `\prevdepth` gets updated to hold the depth value of the newly appended box and thus any following box added manually or by the paragraph builder would correctly calculate glue for baseline separation. There is only one exception to this process: when the material is added as part of an `\unvbox` operation then TEX will neither update `\prevdepth` nor prepend any glue.

It is possible to inspect and even change the current value of `\prevdepth` within macro code while TEX is in vertical mode, i.e., while it is building a vertical list. Thus this only works between paragraphs and not within a paragraph as the paragraph builder acts on a horizontal list after all macro code has been expanded. In other words, you can determine the depth of the last line in a paragraph and by changing `\prevdepth`, pretend that it has a different value and thereby influence the baseline calculation for the first line of a following paragraph. However, you can't do the same for lines within the paragraph.

It is also very important to understand that this parameter is special in that it is local to the current vertical list being built and that it doesn't obey the normal scoping rules. So if you change it within a group it will keep its value when the group ends. Instead it only (and always!) reverts to a previous value if the construction of a vertical list has come to an end and TEX resumes building an outer vertical list, e.g., if boxes are nested within each other or if we return to the main vertical list after building a box or return from an output routine.

[4] Again this is a bit of an oversimplification. There are some more parameters involved and in certain circumstances nothing is appended. Also, if we are at the beginning of the list, or more precisely when `\prevdepth` has this special value, no glue is appended. For precise details see [2], especially chapters 12 and 14.

5 Standard output routines

TEX's paragraph builder is a sophisticated piece of software that uses dynamic programming to optimize the line breaking over the paragraph as a whole (with the sometimes surprising result that a change near the end modifies line breaking decisions much earlier on).

In contrast, TEX's page breaking concepts are much simpler. In essence, TEX assembles material on the main vertical list until it is clear that there is more material than can possibly fit on the page. At that moment TEX stops assembling material for the main vertical list and instead looks through all material gathered and decides on a final break point for the page using a number of parameters to guide this process. The material prior to this break point (which is a vertical list) is then packaged into a box (`\box255`) and a special piece of code, "the current output routine", is fired up.

The normal purpose of this output routine is to repackage and possibly embellish the material stored in `\box255`, e.g., by adding a running header or a page number, and then shipping it out to the output file. When everything has been done, control is given back to the process that fills the main vertical list and processing continues there.

As the lines of a paragraph are always added in one go to the main vertical list, TEX has typically accumulated more than it actually can use in the output routine. So when it returns from processing the page material, the main vertical list is not empty but contains a few boxes (and glue) that TEX had seen but decided not to use.

Furthermore, the output routine code is allowed to put some material from `\box255` back (typically after splitting it into several pieces) and in fact it

can even generate new material to be put back into the main vertical list. To allow for this, TEX starts a new vertical list when the output routine starts and the output routine can then place box-glue-penalty items into this list while working. Once the output routine has ended, this vertical list (if it contains anything) is placed at the head of the main vertical list, followed by any material already on it but not chosen for the current page.

Now what happens with `\prevdepth` during that time? When the output routine starts, it holds the depth of the last box contributed to the main vertical list, which may or may not be the last box that shows up in `\box255`. As the output routine starts a new vertical list, this value is shelved away and this new list gets its own instance starting out with `-1000pt` as usual. So if the output routine does something fairly complicated that includes building paragraphs, these paragraph lines are vertically spaced out using the rules explained above. Once the output routine ends, the value for `\prevdepth` from the main vertical list is restored.

This is normally the correct decision: if some material was not being used for the current page, then this will form the end of the main vertical list after the output routine has ended and thus `\prevdepth` will correctly reflect the depth of the last box appended there.

If on the other hand all material got used for the last page, then the value of `\prevdepth` no longer reflects the real situation as it still contains the depth from the last box. However, as long as the main vertical list is effectively empty at this point, this doesn't matter as TEX throws away any glue item after a page break until it sees the first box. It then inserts new glue, based on the height of this box and the current value of a parameter, namely `\topskip`. So all that happens is that the paragraph builder may have calculated a glue to go in front of the first paragraph line based on wrong assumptions but as this glue is thrown away later it doesn't matter.

6 Special output routines

But there is one further case: everything from the main vertical list got used but the output routine itself put something back. In that case the last box on the main vertical list will be whatever the output routine has deposited there, but the value of `\prevdepth` still reflects the last box that was there before the output routine was called.

The standard output routines in LATEX and plain TEX do not have this issue as they do not put anything back. However, the situation is quite different if you look at special output routines. These output routines typically get explicitly invoked by setting some explicit penalty and thus there will be no leftovers on the vertical list that correspond to the `\prevdepth` value.

For example, the multicol package, on reaching the end of a `multicols` environment, invokes an output routine that takes the gathered material, splits it up into balanced columns and then pushes the result back as a single block for reprocessing.

In that case the value of `\prevdepth` on the main vertical list will be the depth of the last box in the last column, but after balancing the overall depth of the result may very well be quite different (as the last column may be the shortest one, so its depth isn't even taken into account). As a result the baseline calculation of a following paragraph line will be based on wrong assumptions.

In fact multicol tried to account for this and added a negative space and then set `\prevdepth` to zero. However this happened inside the output routine so that the negative space survived but the value of `\prevdepth` got reverted after the output routine returned! (And it doesn't help to use `\global` from within the output routine as `\prevdepth` simply doesn't care.) As the difference is typically less than 2pt and multicol additionally adds a space of `\multicolsep` this bug remained undetected for a long time.

The solution to this problem then is, of course, to carefully keep track of what the output routine intends to put back, measure the final depth within the output routine and store it away in a global variable. Then, once the output routine has ended, explicitly set `\prevdepth` to the saved value to make it reflect the true situation.

7 An unsolvable problem?

The previous section explained how special output routines can be written to correctly reflect the situation with `\prevdepth`. But this requires that the output routine is always explicitly triggered — a situation in which we know that there is no remainder material that could throw us off track. But what happens if the output routine puts material back but is invoked asynchronously by the standard TEX mechanism?

For starters we then have a problem in regaining control after the output routine has ended though that can be resolved with a few tricks involving `\aftergroup` and a nested set of output routines.

But even if we do this we don't really know to what value we should set `\prevdepth`. It would need to be the depth of the material we put back in case there was nothing left on the main vertical list, but

it needs to be left alone if this is not the case. And we can't arrange for the material returned to have the same `\prevdepth` that was current before the output routine since we don't have access to this value within the output routine, and as the output routine is triggered asynchronously we can't (easily?) obtain it beforehand or as part of the process.

So this is something to ponder.

8 Some answers

Donald Knuth already gave a partial answer to this problem in *The TEXbook* [2, p. 262ff] where he discussed an output routine that adds index headings in random places in the text. The restriction in his algorithm is that the `\prevdepth` is assumed to be sufficiently small (that is, smaller than `\maxdepth` in fact). In that case we can use the depth of `\box255` in the output routine as a measure for the `\prevdepth` calculations that will have taken place if there is any remainder in recent contributions, and use this to adjust the nominal depth of the material added to match that. And if there isn't any remainder then this doesn't really matter either.

However, this approach can't be used unmodified if this restriction isn't guaranteed to hold, i.e., if the material typeset may have arbitrary depth that is then masked by the page `\maxdepth` adjustment. Artificially enlarging `\maxdepth` is not an option either, as that would incorrectly alter the allowed page break positions.

A possible extension of the algorithm is to re-box the material inside the output routine to determine its natural depth, but unfortunately that turned out to be not enough to cover all cases.

So after a couple of false positives (i.e., pseudo-solutions that failed in one or another boundary case) my conclusion is that this problem cannot be solved within TEX or ε-TEX as long as the typesetting and line breaking is done by the engine. The main reason for this is the following:

- If TEX is doing the line breaking and automatically appends new material to the vertical lists it calculates the necessary glue based on the height of the newly appended box and the current `\prevdepth` value to achieve a baseline to baseline distance that corresponds to the value of `\baselineskip`.
- However, if that brings two boxes too close together it adds some extra glue (`\lineskip`).
- So if a "baseline skip" glue was added we can adjust its value based on the size of newly inserted material as we know the target size (i.e., `\baselineskip`) and the `\prevdepth` used (applying a variation of Knuth's algorithm outlined above).
- If, however, a `\lineskip` glue was added, our calculations are off base and there is no way within TEX to determine that we are in this branch of the typesetting algorithm short of disabling it and doing all box maneuvers manually.[5]

With LuaTEX the situation is different: With some moderate Lua programming effort it should be possible to traverse a node list, say the one stored in `\splitdiscards`, and determine if `\lineskip` was used. Depending on the scenario one could then either keep that node or delete it or replace it with an appropriate new value.

Addendum

Thanks to Petr Olsak for identifying a needed correction to the text in section "Calculation of vertical glue" on how TEX handles manually added boxes when building a vertical list. This online version of the article has been updated accordingly.

References

[1] Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. In *Digital Typography*. CSLI Publications, Stanford, CA, USA, 1999.

[2] Donald E. Knuth. *The TEXbook*. Addison-Wesley, Reading, MA, 1986.

◇ Frank Mittelbach
Mainz, Germany
`http://www.latex-project.org`

[5] I would be very much interested to be proven wrong here: If somebody finds a solution that covers the general case using just TEX or ε-TEX, please offer it as a *TUGboat* article.