

The sequel to `firstpass.tex`

Federico Garcia

Started Aug. 30th, 2004

Contents

1	Ordering the elements	2
1.1	Anatomy of an element	3
2	The handling of instruments by T_EX	3
3	Space factors	4
4	Beaming	4
4.1	A qualitative algorithm	4
4.2	Chords in beams	5
4.3	Step 7	5
4.4	A breakthrough	6
4.5	Beaming refined	6
4.6	To do...	7
5	Stems	7
5.1	The stem-point	7

The file `firstpass.tex` is a travel-book, written in Spanish, of the first ‘epoch’ of T_EX^{muse}’s programming. This epoch has come to an end, because of several things, but mainly because I just finished the 2004-version of my *TUGboat* article, so it’s like a symbolic point.

I am renaming some files, so that the first epoch is still documented.

- The code of T_EX^{muse} used to be in `inven.tex`. I extracted it as `txmssmpl.tex` for the article, and now I’m calling it `texmuse.tex`. It is the file with the macros, no input, no output.
- The METAFONT programs are `texmuse.mf` (which used to be `txmsnew.mf`), and `txmfont.mf` (containing the actual pictures—used to be `tmsfont.mf`).

There is a new file, `invention2.tex`, which is where tests happen. I’m basing this second epoch on Bach’s second invention.

1 Ordering the elements

The essential change of approach for this second epoch is that the elements of the characters are now going to be ordered before they are actually drawn. This might have an effect on programmability, but in any case the old system had already an incipient (and inconsistent) ordering idea: the whole ‘`to_do`’ business.

So all I’m doing now is to expand this procedure. The elements—so far—fall into one of 6 ‘levels,’ thus (I’m going to append other elements as they are implemented—punctuation reveals additions):

1. Note-heads and rests.
2. Stems and flags.
3. Accidentals.¹
4. Barlines.
5. Clefs.²
6. Dependant on `note_axis`: Beams, ties, and slurs.

Very important questions: are there any things that *both depend and affect* `note_axis`? In any case, item 6 is things that won’t change `note_axis` in any way.

METAFONT has two variable, ‘`next_level`’ and ‘`curr_level`.’ For each new character, they are both initialized to 1. Elements then start to be met. An element of level n does:

- Command `instrument` is to be included in all successive versions of `to_do`.
- If $n = \text{curr_level}$:
 - `note_axis` is fixed if $n = 6$.
 - The element is drawn.
- If $n > \text{curr_level}$ (elements of $n = 1$ assume this, there’s no test):
 - If `next_level` is 0, initialize `to_do` to be the element.
 - Otherwise, append the element to `to_do`.
 - Set `next_level:=n`.

The basic convention is that `next_level` is 0 when there is nothing at a higher level than `curr_level`.

To be noted is that METAFONT never encounters an element of level lower than `curr_level`.

¹Not key-signatures.

²Not line-opening clefs.

Then, `curr_level` is set to 0, `doing_` is set to `to_do`, and `to_do` is set to `"fix_axis; finalize_char"`.

`note_axis` is fixed, in any case, before `finalize_char`, either because there is an element of level 6, or because there is none and `find_extremes` is executed right before `finalize_char` by the 'default' `to_do`.

This is going to require a complete re-programming of `texmuse.mf...` (so I'm saving the old version as `texmuseold.mf`).

1.1 Anatomy of an element

Elements of level 2–5 have follow this general outline:

If their level is greater than the current level, they defer: they add the `curr_instr` and themselves to the list `to_do`. Then they set `curr_instr` to nothing (so that other deferred elements in the same instrument don't add it too). They take care of either replacing, or appending `to_do`, depending on whether there were deferred elements before (at this level).

Otherwise, they actually draw the element.

For example:

```
def regular_stem =
  if curr_level<2:                                % Deferring...
    to_do := if next_level>0 : to_do & fi curr_instr & "regular_stem;";
    curr_instr := "";
    next_level:=2;
  else:                                           % Doing...
    make_stem(stem_coef*stem_length);
    if abs(aggr_note[stem_coef]+instr_clef[instr_])+stem_coef > stem_length:
      stem_extreme:=1;
    draw stem_point---(xpart(stem_point),interline) fi;
  fi;
enddef;
```

Elements with arguments cannot be implemented in the usual way of macros with arguments. Rather, a text variable is defined and contains the argument. This way, the deferring can append that to `to_do` without actually scanning it.

2 The handling of instruments by T_EX

So far, T_EX has been issuing a `instrument` command to tell METAFONT it's going to build the note for the next instrument. That has a problem, but in addition is not the most efficient (suppose there are many instruments with nothing, there will be a stream of consecutive `instruments`).

The new way is: METAFONT commands are defined for the instruments, and T_EX issues the one corresponding to the instrument it is going to typeset (if it is).

3 Space factors

It seems better to use `new_char` than `finalize_char` to give METAFONT the space factor of each character. The whole note classification through the argument of `new_char` is then shut down, but it doesn't seem to be necessary.

4 Beaming

It's proven that the inclination of the beam can be achieved by means of a 'transform' variable. It's the vertical analogous of `slanted`, so I've called it `yslanted`. It has the form $(0,0,1,0,s,1)$. METAFONT can easily figure the value of s out from the images of three points (say, the two extreme note-heads, and a point up from one of the extremes).

But beaming has to be handled with care, because the inclination is not defined by the note-heads; rather, it has to be 'rounded' so that the extremes of the beam fall in one of three possible positions respect to the staff: holding from, sitting on, or stranding a line. (Inner beams can also start or end 'on the air,' suspended between two lines).

The *direction* is determined by the two notes that are closest to the beam, and by what the left- and right-most notes are.

The *degree of inclination* would seem to be determined by the distance (both horizontal and vertical) between the two notes that are closest to the beam. But a better option is to think of it as that inclination for which the total sum of the lengths of all the stems is minimum.

The *position* (what line and what position respect to it) is determined by the note that is closest to the beam, and by the position that it would imply for the beam as a whole.

4.1 A qualitative algorithm

So the `beam` function can proceed thus:

1. Find the direction of the stems, the total length of the beam, and its 'general direction,' that depends on the relationship between the first and the last notes.
2. Reduce to 'relevant notes,' namely those individual notes from each aggregate that are closest to the beam.
3. Find the closest note-head. (Its index is enough to specify the note completely.)
4. The closest note-head—or more exactly the point from which its stem stems—is put at $(0,0)$, and then the rest of the note-heads—*idem*—are put at their corresponding points, perhaps scaling.

5. For a number of possibilities (in terms of quarters of interline-space, the possible ‘heights’ of the beam), the total sum of all the stem lengths is calculated.³ Instead of ‘rotating’ the beam, the notes are moved up or down according to each hypothetical case. If any note goes above the horizontal axis, the inclination is immediately discarded (its stem would be shorter than the minimum length allowed). 1/4 of interline space is given a bonus, to ensure that a second is beamed with a 1/4interline-height beam.
6. That calculation defines, implicitly, the angle of inclination. It’s explicit in the form of a transform variable (`yslanted`).
7. The position of the beam is decided taking into account the inclination and the inner beams. Upward-beams should not depart from sitting positions, downward-beams should not depart from holding.
8. `yslanted` is applied to everything except the point from which the stem departs (toward the beam).

4.2 Chords in beams

The form of the argument to `beam` so far is good only for 1-note-per-moment beams. In order to provide for chords, an extra number is needed for each note: the number of note-heads it has. After that, the note-heads themselves, separated by commas: ‘2,a5,c5’ would be an example of a valid construction. In general, the argument repeats this:

(index),(no. of beams),(no. of note-heads),(note,note,...)

4.3 Step 7

`draw_beam` will originally start the beam (from the closest note) at a height of `closest_note` plus or minus `stem_length`:

`beam_lift = closest_note + beam_coef*stem_length.`

But now we have to adjust this lift to make sure that the beam starts in the right vertical position for the first note.

This adjusting the position of the beam will only be necessary when in “careful mode” (4.5). This means that the inclination will always be either 0 or 1/4. In principle (only having into account 1 beam), we will then know that either the first or the last note (depending on `beam_dir`) has a stranding beam. So it’s simply a matter of rounding off the `yslanted` position of this note to the nearest integer.⁴

From deciding how carefully the beam is to be traced, METAFONT will know the number of beams that it needs to be careful about. When it’s one, it

³The ‘number’ of possibilities seems to be 3 in Finale: 0, 1/4, 1/2.

⁴Throughout this section the ‘correction’ of 1/4 that takes place in the actual drawing of the first beam is to be noted.

proceeds as explained above. This works also when there are two (the inner one will strand at the beginning). And when there is three beams involved in the adjustment, there is no way to avoid staff-line crossing, so the procedure might as well be the same. It seems, however, that it is preferable that the middle one of the three ends right between staff lines. This is also true for four, and even five, beams involved.

So the key is to find the outer beam *that is relevant* to the position, that is, the first one that enters into the staff.

4.4 A breakthrough

In the early implementations of the beaming algorithm there was an endemic problem. Inclination was calculated from the closest note, put to (0,0). This is very elegant. But then when the beam was drawn, (0,0) was not the closest note, but the last one. Adding a **shifted** part to **yslanted** affects stemming (the extreme of stems is calculated by using **yslanted**) in undesirable ways.

The solution is simple: the beam will be appended to the closest note's character, not to the last one's. The only possible problem is the **undrawing** that takes place, but it should be possible to manage it well. A side effect of this procedure is that, supposing that the closest note falls somewhere in the middle of the beam, the length of the beam can be larger and METAFONT will still be able to handle it. Maybe this is a step toward implementing too-long beams.

This solution (come to my mind on Sept. 24th 2004) has a more general dimension: the characters are effectively drawn only by **make_line**, so that it is possible to add things to any of them at all points during the creation of the line.

4.5 Beaming refined

In all truth, beams should behave differently when they happen 'on' staff-lines from when they don't. Finale, for example, makes all beams at least 1/4 tall when inside the staff, although it might go to 1/2 when outside. This is true of every rhythmic value.

This affects **find_inclination** and **find_position**. The first thing to be done is finding out whether the beam falls inside the staff: if **closest_note** plus **stem_length** are within the limits of the staff (-1 and 3) for a regular staff, then the beam is going to be drawn 'carefully:' maximum height is 1/4, and it always starts in a fixed way (for both **beam_dir** and **beam_coef** positive, for example, always holding).

The staff is defined by a text variable **staff[i]** (**i** is the instrument) with numbers separated by commas. Therefore, after finding the regular position of the beam from **closest_note**, it can be compared to **max(staff[i])** and **min(staff[i])**. It works as the rest of the beam's attributes: it's found out by **prepare_beam** *only* if it is **unknown**: the user can have specified it through different commands.

4.6 To do...

This is written April 4, 2005. I implemented the ‘careful mode,’ but it is not that simple. There are two ‘levels of care:’ if the beam is near the staff (within 1 interline of it), inclination of the beams should be at least 1/4. But the position doesn’t have to be adjusted but when the beams actually go *into* the staff. And this is having into account the number of beams of the extreme notes.

That remains to be done. The other thing is to fully generalize `find_position`, that is working properly (?) for beams with the stems going up, but apparently not when they go down.

5 Stems

There are variables `stem_axis`[] [] (offset of the stems respect to `note_axis`, and later respect to `note_axis+note_offset`, when this new variable allows for individually offset notes), and `stem_dir`[] [] indicating direction (up or down). This latter is to be set by the elements that really need them (`regular_stem`, `beam`), if the user has not specified it. In terms of it, `stem_axis` is decided (according to the type of note-head).

Eventually, `stem_point` will substitute `stem_axis`, in order to provide for strange note-heads.

Now, there are some note-heads that are not symmetrical, and furthermore many are wider (or narrower) than the regular note-head, so that if there are regular note-heads in a note, the strange one has to be placed according to where the stems lies. That’s a problem, because sometimes the position of the stem will not be known (for instance, under beams).

The solution is this: each character appends these note-heads to a text variable `note_heads`[] []. Before starting a new character, METAFONT scans the information contained in it: if there is any pending note-head for which `stem_dir` is known, it adds it to the picture of the note.

5.1 The stem-point

Each note-head picture has to define its ‘stem-point’ for both sides (i.e., for both `stem_dir` 1 and -1). This is independent from the actual drawing of the note-head: whenever `stem_dir` is found, `stem_point` is defined. So, for example, beams will always be drawn when `stem_point` is already known.

`stem_point` is a pair variable: its x part is what used to be `stem_axis`; its y part is the vertical displacement from the center of the note-head. Thus, it is independent from the actual note (pitch).