NtEd a new and free musical score editor for Linux

Dr.-Ing. Jörg Anders
Chemnitz University of Technology, Fakulty of Computer Science
Strasse der Nationen 62, 09107 Chemnitz, Germany

ABSTRACT

NtEd[2] is a WYSIWYG musical score editor, which can support music teachers and music students to learn reading notes, to learn playing an instrument, and to compose music. Apart from all other WYSIWYG musical score editors it is cost free because it is licensed under GPL[1] and runs on Linux operating system, which is cost free, too. Thus, you can save money which is often a real obstacle for using such software in class rooms, because the usual WYSIWYG musical score editors are very expensive.

NtEd lets you insert notes to create a musical score, play the score, import MIDI files, and record the score from a MIDI keyboard attached to your sound adapter.

This article gives a short introduction into NtEd, explains the difficulties in writing such a software, and indicates how NtEd solves them.

Keywords: Musical Score Editor, WYSIWYG Editor, ALSA, Linux, Open Source

1. USING NtEd

You can get NtEd from [2] for free. You can get both, the ready packages and the C++ source. The main window [Fig. 1] lets you insert notes by means of the mouse or alternatively by PC keyboard. As you can see NtEd allows you also to place chord names and guitar chord diagrams. Furthermore, you can use drum notes. By means of the toolbox in the right bottom corner you can select the note length, the accidentals like sharp, flat, natural, double flat a.s.o., the articulations like staccato, tenuto, portato a.s.o., and the note style for drum notes.

By clicking pairs of notes you can connect them with a tie (legato). You can build chords and tuplets (2 - 13). You can place repeat signs and rests of different length.

NtEd places the material onto different systems and pages. The user can continuously change the size of the musical signs and he/she can choose the paper format.

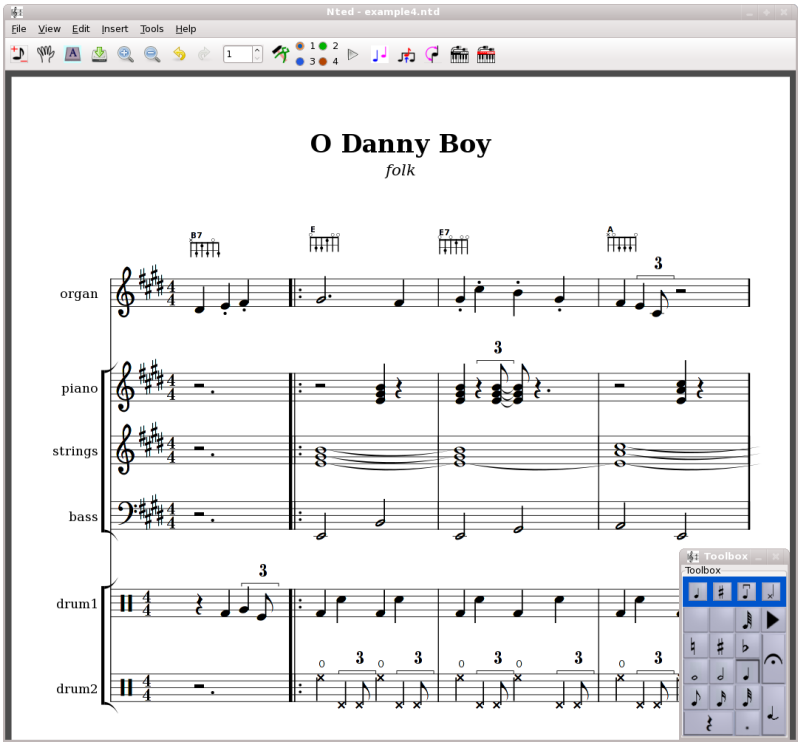

Fig 1: The NtEd main window

NtEd allows up to 4 voices per staff:

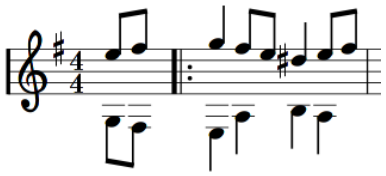

Fig 2: Two Voices in a staff

Furthermore, the user can attach an instrument to every staff. Thus, a whole orchestra is built. The play button starts replay via ALSA (Advanced Linux Sound Architecture)[3]. If some of the instruments are not in C tuning, NtEd allows you to add an offset to the pitch.

Other interesting features of NtEd are:

- **Transpose** You can transpose the score or parts of the score to a new key.
- **Mute** The teacher can mute some staves. This way the pupils can recognize how the single instruments contribute to the whole sound.
- **Copy/Paste** You can select groups of notes and copy them somewhere, also between different scores.
- **Extract staves** If the composition is ready it is necessary to create the notes for every single instrument[Fig. 3]

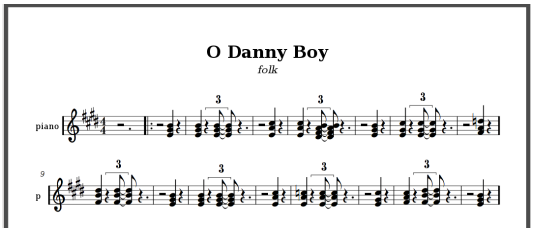

Fig 3: The Piano excerpt

To print the score there are 2 ways:

1. There is a *print* function on some menu which prints the score such that the printer output matches exactly what you see on display.
2. You can export the score to LilyPond[4]. LilyPond is a text based score editor, which is also cost-free (GPL). It creates the score from a text transcription of the music. For instance, the text:

```
\relative c' {\clef treble
\key a \major \time 2/4
 e'4  cis cis2 d4  b b2
 a4  b cis  d e  e e2
}
```

produces the following PDF output:

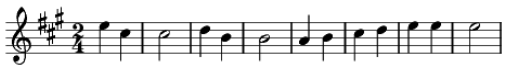

Fig 4: The LilyPond output

Thus, one can export the score to LilyPond, create the PDF by means of the LilyPond processor and then print the PDF. Why is it useful: The LilyPond output is better than the NtEd output, because LilyPond as an offline renderer takes as much time as needed to create the best results. So, you can regard LilyPond as a beautifier.

But there is a small drawback: Because LilyPond uses its own placing algorithms and its own music fonts the WYSIWYG principle is broken.

NtEd is able to create:

- PostScript
- PDF
- SVG
- PNG
- MIDI

It can import:

- MusicXML[6] *(with restrictions)*
- MIDI

Especially the latter is very helpful for pupils and music students. Most people who play an instrument are unable to play a piece of music without notes. They cannot hear what is played. But there are dozens of MIDI files in Internet. By reconstructing the musical score from these MIDI files, NtEd gives such musicians the possibility to play the music.

A further interesting feature is: Recording the music from MIDI keyboard. If a MIDI keyboard is attached to the sound adapter, the user can play on this keyboard and NtEd creates the appropriate notes. On one hand this eases the score creation. On the other hand a teacher can give his pupils an idea of how even live played music can be notated.

2. WHY IS IT SO DIFFICULT?

There are only a few WYSIWYG musical score editors. And the best ones are relatively expensive. One reason is: There are potentially only a few musicians which are willing to use a computer.

Another reason is: It is difficult. Sometimes more difficult than writing a text editor.

This chapter will give a brief overview of the main problems. Most of them come from timing restrictions. A WYSIWYG user expects a program reaction immediately after the mouse click. That is, you have at most 50 ms from mouse click till program reaction. All placing decisions have to take place during these 50 ms.

Vertical Alignment

In contrast to a text editor, the program must always keep a watch on the vertical alignment. In following example there are 2 tied quarter notes in first staff and one eighth in 2nd staff:

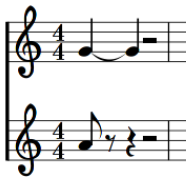

Fig 5: Before alignment

Now the user adds 4 32ths and a quarter into the 2nd staff:

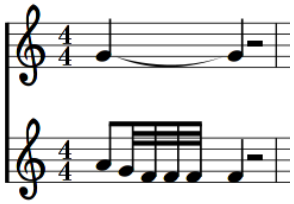

Fig 6: After alignment

The 2nd quarter in 1st staff must be replaced, because it sounds together with the quarter in 2nd staff. But even worse: The tie, a complicated filled Bezier curve pair, must be re-computed. And at end of line it can happen the line is too long and must be broken. This is still more complicated because you can move only whole measures, not parts of a measure. And think about the undo and redo functionality: All of this must be reversible, including the line break!

Horizontal Grouping

Horizontal grouping is almost unknown in text processors. Have a look at Fig 6: There are ties and beams which group notes together. Imagine the user moves up the 2nd note in 1st staff and inserts a 32th rest in 2nd staff:

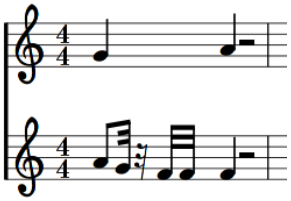

Fig 7: After modification

The tie must be broken because it can exist only between notes on the same line. But the beam must be broken, too. And bear in mind: All this must be reversible by undo button! This requires continuous recomputation of such groups.

3. NtEd SOLUTIONS

NtEd handles the vertical alignment by means of a complicated system of dynamically enlargeable arrays. If the user changes the score, all notes and rests are tagged with the time they should sound. This happens staff by staff. In a 2nd step all notes are placed into a position array, which can be regarded as a 2 dimensional grid. The example in Fig. 6 would result in:

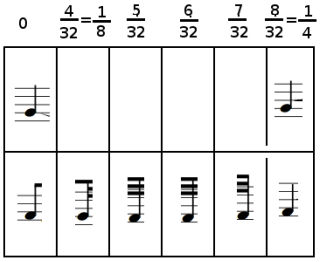

Fig 8: The position array

Elements which have the same time appear in the same column. Every symbol has its bounding box. The position algorithm computes the needed space for every column by determine the maximum bounding box width per column. From this information the position of each column is computed and the symbols in this column are tagged with this position.

This takes place every time the user modifies the score. The times are computed, the array is build, the positions are computed. And all this is fast enough to meet the 50 ms requirement. This holds even if the score is the first movement of Beethoven's 5th symphony with 12 staves and 500 measures! To save time NtEd tries to keep track of already positioned staves to avoid double positioning.

The handling of horizontal grouping is still more surprising: Before positioning there isn't any single information where to place beams. Because the beaming follows certain musical rules, the beams are computed after all elements are placed.

For ties the notes are internally connected by pointers. As you can imagine this is a source of many crashes especially because ties can be removed implicitly as shown in Fig. 7.

To draw the ties NtEd uses the Bezier curves offered by the Cairo library[5]. This is a very fantastic and very fast graphics library. It can draw lines, curves, paths, circles, and predefined fonts. And it can fill all these elements, independent of whether they are concave or convex. And it can antialias these elements. This is important, because it is impossible to give the signs a smooth appearance by using only black and white pixels. Intermediate colors must be computed as an enlargement shows:

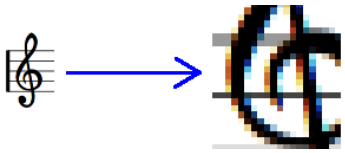

Fig 9: Antialias

Believe it or not: If the user moves upward a single note as happened with the tied quarter in Fig. 6 the positioning takes place, the whole page is cleared (filled with white color), and the score is redrawn with antialias. And all this is so fast, that the user has the impression to only move a single note upward. The only "trick" is the usage of a double buffer to avoid flickering.

Other Problems

There are of course much more problems. The beams and especially the beam slope must be computed. Regard Figure 10:

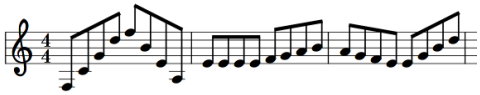

Fig 10: Different beams

As you can see the beam slope depends on the position of the beamed notes. NtEd offers 7 different slopes between -30° and 30°. Believe it or not: Although there are only 50 ms available, NtEd tests each of the 7 slopes and chooses that one, which causes the smallest change of the stem length compared with the un-beamed notes.

And there is another issue. The length of the beam groups depends on the time signature:

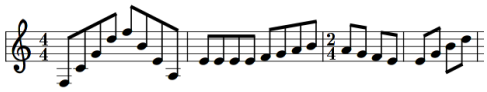

Fig 11: beam length

Here is nothing to compute! How many notes are beamed on which time signature is a result of a long history of note engraving. NtEd has an internal table which determines where to break a beam depending on the time signature.

Of course, the user can overwrite these rules in special cases. The same holds for the slope.

Some further problems are caused by polyphonic music. For certain instruments and choir music it is usual to place different voices on one staff. NtEd offers up to 4 voices per staff. Imagine the user places a first voice:

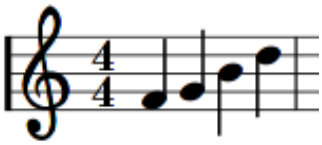

Fig 12: Single voice

Then he/she inserts some notes in the 2nd voice:

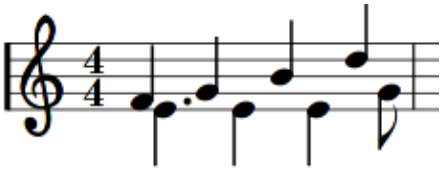

Fig 13: Polyphony

First of all a stem decision must be made. A musician expects the notes belonging to one voice to have the same stem direction. Note, the stem direction can change in next measure if the voices cross. Therefore, NtEd computes the stem direction on a measure-by-measure basis. To do this, it builds the average pitch and uses the stem up policy for the voice with higher average pitch.

If the voice count increases this problem becomes more and more difficult. In case of 4 voices it is often up to the user to overwrite the stem direction or to break some beams to avoid line crossing.

I regard it as impossible to solve all problems automatically. In general this is a task left to the user, even if a very professional *(and very expensive)* musical score editor is used.

Figure 13 discovers a further problem: Regard the first notes of every voice. They sound at the same time. But there is no place to put them at the same horizontal position. This is a known problem of music notation. So-called microshifts must be applied. To detect such conflicts and to compute the microshifts every column of every voice must be compared with the other voices. Thus, the computation complexity increases with the square of the number of voices. Bear in mind: All must be done during 50 ms! So, the limited computation time limits the number of voices.

4. MIDI IMPORT

As already stated, NtEd can import MIDI files. Users expect this feature as a matter of course. But reconstructing the score from MIDI file is a very difficult task. A MIDI file can be regarded as an endless paper with holes, like the paper roles used in mechanical pianos. Every line represents a pitch (D, G, and C in this case):

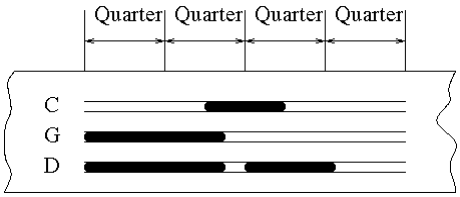

Fig 14: MIDI data

The horizontal direction represents the time. And the duration of a quarter note is know by a field in MIDI file header. The score looks like this:

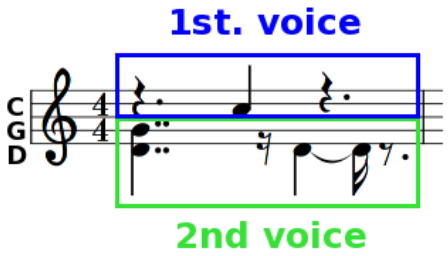

Fig 15: Score of MIDI data

As you can see, some very intelligent decisions must be made:

1. Because D and G end after C begins they belong to different voices.
2. There seems to be no conflict between the leading D and G and the 2nd G, i.e. they belong to the same voice.
3. There is a conflict between the leading D and G, but they start and stop at the same time, i.e. they build a chord.
4. The length of the chord is a quarter + an eighth + a 16th, which can be expressed by a double dotted quarter.
5. The length of the last D is a quarter + a 16th which can only be expressed by tieing 2 notes together.
6. The gaps on paper must be expressed by appropriate musical rest symbols.

As you can imagine it is not quite easy to perform the MIDI to score transformation. And Figure 14 shows a relatively simple example. The problem is still much more complicated if the MIDI file is produced by a musician playing a keyboard. Imagine the start of a C scale:

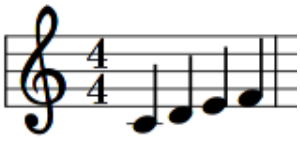

Fig 16: A C scale

The MIDI data should look like this:

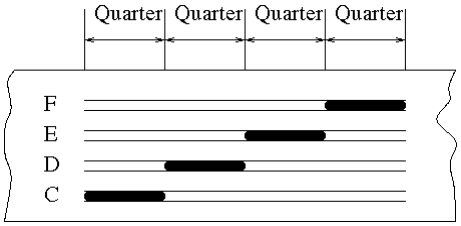

Fig 17: Ideal MIDI C scale

But no musician plays it this way. Most musicians release the keys a bit - say a 32th - too early *(the last 32th is filled by reverberation and sustain effects)*:

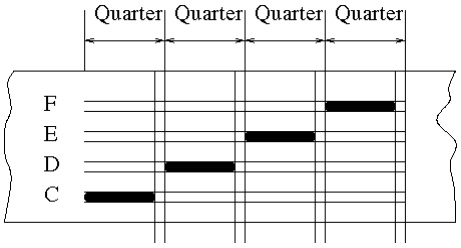

Fig 18: Bad MIDI C scale

An exact MIDI ⇒ score transformation would result in a system of double dotted eighths followed by a 32th rest:

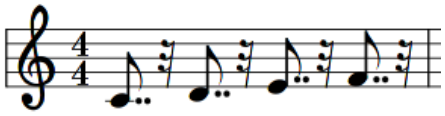

Fig 19: Bad MIDI C scale

Note, this is unplayable! NtEd has some heuristics which recognize the situation and reproduce the correct C scale as shown in Figure 16.

5. OUTLOOK

There are still some missing features. A lot of additional musical signs must be implemented. Fingering is important and surprisingly many users request special notation for lute or Gregorian notes.

But my dream is: OMR (Optical Music Recognition). It would help a lot if the user could scan a sheet of music and NtEd could read it. Many commercial musical score editors offer this feature, even though as very expensive additional module.

6. MOTIVATION

At last one may ask: Why does the author create such a program? Indeed, my field of activity is computer networks and distributed systems which has less to do with music.

I have to lead tutorials which require programming knowledge. In the recent times it became more and more difficult to be better than the students in writing programs. Therefore, some time ago decided to start to implement a sufficient complicated object oriented program. I decided for musical score editor because there exist only a few programs and especially on Linux there are only 2 WYSIWYG musical score editors, at all.

References

[1] The GNU public license:
`http://www.gnu.org/licenses/gpl-2.0.html`

[2] Jörg Anders: NtEd
`http://vsr.informatik.tu-chemnitz.de/staff/jan/nted/nted.xhtml`

[3] ALSA: Avanced Linux Sound Architecture
`http://www.alsa-project.org`

[4] LilyPond:
`http://www.lilypond.org`

[5] The Cairo library:
`http://cairographics.org`

[6] MusicXML:
`http://www.recordare.com`