Requirements for a Music Engraving Program: a Composer's Point of View

Jean-Michel Hufflen

Abstract

It is well-known that some typesetting systems are *interactive*, that is, WYSIWYG, whereas some—WYSIWYM—work like *compilers* and process input files written using an *input language*. Interactive systems provide interesting features, whereas other qualities are implemented by WYSIWYM systems. We can observe the same points about music engraving programs. In this article, we summarise the properties of interest during the music composition process and review some music engraving programs from this point of view.

Keywords music engraving, scores, music typography, music composition, versioning, managing musical instruments, building MIDI files, Finale, LilyPond, MusiXTEX, NoteEdit, MuseScore.

Sommario

È ben noto che alcuni programmi di elaborazione di testo sono *interattivi*, cioè WYSIWYG, mentre alcuni — WYSIWYM — funzionano come *compilatori* ed elaborano i file di ingresso usando un particolare *linguaggio*. I programmi interattivi presentano funzionalità interessanti, mentre altre funzionalità sono fornite dai sistemi WYSIWYM. Possiamo osservare le stesse caratteristiche nei programmi per scrivere musica. In questo articolo presentiamo le funzionalità che interessano nel caso della composizione tipografica della musica e commentiamo alcuni programmi da questo punto di vista.

Parole chiave Incisione della musica, spartiti, tipografia musicale, composizione musicale, controllo delle versioni, gestione degli strumenti musicali, creazione di file MIDI, Finale, LilyPond, MusiXTEX, NoteEdit MuseScore.

1 Introduction

Among typesetting systems, the clear distinction between WYSIWYG[1] and WYSIWYM[2] systems is well-known. In the first case, such programs—an example being Adobe InDesign—are *interactive*, that is, the formatted text is directly displayed on screen, and updated as soon as end-users enter new characters or activate some menu operations. In the second case, a source file written using an *input language* is processed—this step may be viewed as a *compilation*—and result in the complete formatted text. Of course, LaTeX and other programs built out of TeX belong to this second category.

The same distinction exists for music engraving software, that is, programs drawing music scores. Some programs—e.g., Finale, MuseScore, NoteEdit—are interactive: a score is built step by step by means of graphical interface, even though notes and other musical signs are progressively written by hand on a sheet of paper. Some—e.g., LilyPond, MusiXTEX (Taupin *et al.*, 2002)—are clearly related to a WYSIWYM approach.

The advantages and drawbacks of these two approaches have already been described in many articles about word processing. However, writing music is a different task than writing texts, and some arguments relevant about written documents do not apply to music scores. Symmetrically, a music composer does not have the same requirements than a book writer. Since I personally compose music during my spare time, I judge interesting to give my point of view. I experienced some music software, not all of them, so my study is not exhaustive. However, I tried to express what I like or dislike in some programs, as precisely as possible.

In the first section I explain what a composer expects from a music program. I begin this section by summarising the requirements for a typesetting system in order to show how different is musical composition. Then I briefly describe the programs I practised in Section 3. I report my experience about music engraving programs in Section 4. Readers of this article are only required basic knowledge of music. Readers interested in precise definitions of musical terms can consult Jacobs (1988).

2 Requirements

In the following, I consider word processor basic tasks, and do not deal with specialised features such as tables, mathematical formulas, pictures of chemical molecules, etc. A word processor should allow its users to get high-quality print outputs. It should be able to implement basic graphical effects, that is, the use of boldface types, italicised characters, etc. It should reflect a document *structure*, from a graphical point of view, by means of hierarchical headings using different character

1. What You See Is What You Get.
2. What You See Is What You Mean.

sizes. It should also implement *cross references* among some subparts of such a structure. Last but not least, a good word processor should allow different parts of a large document to be written by several end-users, and the integration of these parts should be easy[3].

As mentioned above, many articles about the advantages and drawbacks of WYSIWYG and WYSIWYM systems already exist, so I just sum up some important points. Current interactive systems have been improved in comparison with programs used 40 years ago, but they are still limited by their interactivity: when an end-user types new characters, such a program must respond quickly and display a reformatted version of the current paragraph. On the contrary, a WYSIWYM system can examine many solutions before choosing the best way to split a paragraph into successive lines. So does LaTeX: it explores some solutions with respect to criteria summarised into a *badness* measurement, the chosen solution being minimal badness. An analogous *modus operandi* is applied for splitting a chapter into successive pages. An interactive system cannot explore many possible solutions and reach such efficiency.

More generally, WYSIWYM systems allow *style* and *content* to be clearly separated, so users can mainly focus on content when they are writing a document; considerations about style can be examined separately. Interactive systems may be preferred for short documents, e.g., administrative letters, but for large documents, consisting of many parts, such a separation makes it easier to merge these parts or to produce a new version according to another style. A very simple example: building a two-column version of a document previously typeset using a one-column format is easier with a WYSIWYM system than a WYSIWYG one.

Now let us go to music composition (first importanto point). Roughly speaking, there are two ways to get a score for a new musical piece: putting down all the notes and musical signs composing it, or deriving it from piano keyboard or from synthesised music like a MIDI[4] file. The second way outputs scores not ready to use. Such scores must be reworked in order to simplify them: they actually reflect *one* performance of a piece, *too much* exactly. In order words, it does not give the canonical way to express how to play it. A simple example: all of the notes of a *glissando* are explicitly put down onto the resulting score, whereas the standard specification of this effect consists of giving only the starting and ending notes, joined by a line. Like WYSIWYG word processors, music programs deriving scores from synthesised music have been improved compared to the first versions, but some limitations remain. As a consequence, a composer must always deal with scores, either because these scores are written from scratch, or because they are derived but should be reworked.

As a second important point, I think that getting a satisfactory version after improving or changing intermediate versions is a process longer for music scores than written documents. That is true for our personal production, and other composers I asked for this question confirmed that. The goal of such a process is to get the best version but, concerning music, there are some additional points. First, some notes may be misplaced, as if they are mistakes within a musical dictation. If you are able to compose music, you do only a few of such mistakes, but 'a few' does not mean 'nothing'; in fact, they are comparable to typing mistakes within written documents. In addition, as I explained in Hufflen (2017), the use of *accidentals* (♭, ♯, ♮, ...) is error-prone[5]. A synthesised version of a score can allow a composer to detect such mistakes which will have to be fixed. A second point is related to musical instruments: even though a composer knows how to use them, it is difficult—if not impossible—to master *all* of the technical features of *all* of the instruments[6]. In other words, some extracts may be impossible to play by the instrument planned for that[7]. Either such an extract may be suitably arranged[8], or given to another instrument. Similarly, if some instruments are unavailable when a musical piece is to be played, a solution can be to replace them by other instruments; this can yield changes in scores.

3 Some music software

First, let us consider the music programs built out of TeX, briefly described in Goossens *et al.* (2009, Ch. 9), with some examples. In the late 1980s, an early attempt for typesetting musical scores by TeX extensions was MuTeX (Schofer e Steinbach, 1987), which influenced MusicTeX (Taupin, 1992), definitively replaced by MusiXTeX in 1995. This program (Taupin *et al.*, 2002) has been maintained since the accidental death of its main creator, Daniel Taupin, in August 2003, but it seems that only minor development has been done. This robust program, still in use, is usable with *Plain TeX* or as a LaTeX 2ε

3. This point is important for industrially produced documents, but marginal about music pieces.

4. Musical Instrument Digital Interface.

5. Besides, musicologists often have doubts about their interpretation.

6. Many features are described in good orchestration manuals, e.g. Forsyth (1982) or Rimsky-Korsakov (1964). But they cannot be exhaustive. Besides, old manuals did not incorporate recent progresses and new features.

7. For example, the trombone can perform *glissandi* between many pairs of notes, but this effect is not always possible for *every* pair of notes.

8. ... or playable with a special trick: as an example, Stravinsky (1921, 2 bars after Mark 92, p. 79) uses a note too low for a clarinet, but makes it precise that a piece is to be inserted into the bell.

```
\version "2.18.0"

\score {
 \new Staff {
  \clef "treble" \time 4/4
  \accidentalStyle Score.dodecaphonic
  r8 d'8 bes'4~ bes'8[ d'8 bes'8 g'8] |
  c''4. a'8 a'8[ ees'8] ees'4~ |
  ees'16 ges'16 e'16 f'16 b'2 r4 |
 }
 \layout {}
}
```

Figure 1: Example using LilyPond.

package (musixtex). It has been designed to get high-quality print outputs concerning the layout of musical scores, as TEX does for texts. It does not aim to play music. It does not aim to be the input of a program playing music, either. Moreover, it has been reported as difficult to handle. Besides, MusiXTEX's manual says (Taupin *et al.*, 2002, § 2.2.1):

> *[. . .] If this sounds complicated, remember that TEX was designed to typeset text and not music.*

MusiXTEX's input language is very low-level; much placement is up to end-users, e.g., for the notes and corresponding accidentals of a chord. That is why some *preprocessors* have been developed in order to generate source files for MusiXTEX. They are text-based applications, using higher-level input languages and describing the contents of a score without reference to its layout. Historically, the first preprocessor is MPP[9] (Goossens *et al.*, 1997, § 7.4); this project being abandoned for several years. The second is PMX[10]. Scores are specified in a concise way, close to the horizontal and vertical arrangement of an 'actual' score, without reference to formatting; see Goossens *et al.* (2009, § 9.5) for an introduction and Simons (2004) for a complete description. The third is *M-Tx*[11]; see Goossens *et al.* (2009, § 9.6) or Laurie (2005). *M-Tx* language adds a layer of convenience to the PMX language: for example, instrument parts are input as they are printed—that is, from top to bottom—whereas they are entered last line first—that is, from bottom to top—with PMX. As another music program built out of TEX, let us mention TEX*muse* (Garcia, 2012).

Another WYSIWYM program is given by the GNU[12] LilyPond[13]. This project was started by

9. MusiXTEX PreProcessor.
10. Preprocessor for MusiXTEX.
11. Music From Text.
12. Recursive acronym: Gnu's Not Unix.
13. This name is a joke related to Rosegarden, an interactive music software.

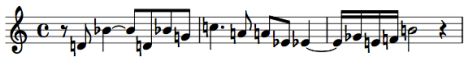

Figure 2: Output generated for Fig. 1.

MPP's authors. LilyPond's syntax evokes TEX since its commands are prefixed by the '\' character. As an example, Fig. 1 gives a specification of a tema of *Lulu-Suite* (Berg, 1935, *Rondo*, Mark 243, p. 3). The result is pictured in Fig. 2. After some global definitions, each note is defined by its pitch—or '`r`' for a rest—followed by a number specifying its rhythm[14]. This software can output music scores as PDF[15] files and generate MIDI files. Other formats are possible, too[16]. An introduction in Italian to this software is Gordini e Liessi (2014).

Interactive music engraving programs allow users to define *systems*, that is, set of staves, and place notes on staves interactively. To do that, a note's pitch and its rhythm have to be selected. Listening to the result is allowed. When a piece is saved, it is analysed and a warning message is emitted if some bars are rhythmically incorrect. Within this category, NoteEdit may also be viewed as a preprocessor since it allows to build input files for MusiXTEX or LilyPond. However, let us mention that files generated for MusiXTEX have to be reworked manually in order to get nice outputs (Hufflen, 2011). I personally experienced Finale and MuseScore. The latter is free, not the former. The services provided are comparable, Finale's ergonomy being better. Another difference is related to *importing* scores written using other conventions, including MIDI files. In both cases, there is no way to *share* common parts. For example, if the first and second violins play the same score, the only way to do that is to use the copy/paste buttons of your software menus.

4 Reporting our experience

Roughly speaking, two reasons may motivate the use of a music engraving program: giving a digital version of existing music pieces, or creating new pieces. In the first case, software built out of TEX can be used, because of the high quality of outputs. As a very good example, a rich collection of Polish traditional songs has been carried out by Odyniec (2016). He used *M-Tx* to ease the specification of lyrics and to get MIDI files to check results. He got very nice scores, but this is not a composers way of working: these songs were composed during a long lapse of time. I have the same

14. Let us remark that LilyPond's `dodecaphonic` accidental style allows us to specify an accidental before each note, as did by A. Berg in his manuscript.
15. Portable Document Format.
16. LilyPond's older versions generated .tex files, but this backend is no longer supported in recent versions.

feeling by looking into examples accompanying MusiXTEX's documentation: they are either classical examples or very simple pieces or arrangements that have not been reworked and reworked.

As mentioned in § 2, it is *essential* to be able to listen to the result of specifying notes. Of course, listening to synthesised music—e.g., MIDI files—does not replace a performance by real instrumentists but provides some feedback and allows some possible mistakes to be fixed. I personally regret that NoteEdit is no longer developed: it allowed a score to be entered nicely, checked, and reworked if needed and, when the score reached some maturity, NoteEdit output files for MusiXTEX or LilyPond allowed high-quality print, even in case the score had to be reworked.

Putting a new score into action with LilyPond—or a MusiXTEX preprocessor—is possible. Moreover, LilyPond allows variables to be used, in particular, they can share some common parts without information redundancy. Advanced features such as *transposition*[17] are directly available. However, such a way is difficult in practice since input languages are quite far from the standard visual representation of music. This drawback exists about mathematical formulas written in LaTeX: it may be difficult to intuit the look of a complicated formula just by looking at the source text used to produce it. On the contrary, we can often guess the look of fragments in text mode even though they are marked up by commands. Let us go back to input texts for LilyPond: they are difficultly practicable for a musician who would not be also a computer scientist. For that reason, I personally use LilyPond for musical texts that reached an (almost) stable state, but prefer MuseScore for scores developed from scratch.

If we consider the specification styles for output scores, LilyPond and Finale are the best, but the other programs seem to me to be satisfactory. Let us remark those parameters related to *spacing*: horizontal distance between adjacent notes, vertical distance about staves and systems.

5 Conclusion

Musical typography is a fascinating domain and we can admire the results produced by music engraving programs in most cases, concerning classical and popular music, or modern music using 'classical' effects[18]. But that raises difficult problems: reading a score is not a linear process, contrarily to reading a book, as I show in Hufflen (2015). Some historical background can influence the look of a score (Hufflen, 2013). The most part of musical symbols have been included, incompletely (Hufflen, 2014, 2017), into Unicode.

Let us go back to music composition. I tried to give a synthetic point of view of this activity. More technical details related to particular cases can be found in Hufflen (2011, 2012). I think that WYSIWYM systems are interesting, as shown by LilyPond success, but with a WYSIWYG interface. In other words, the *process* of getting scores as PDF files or synthesised music as MIDI files does not need to be interactive, but entering data should, unless a nicer input language is proposed.

17. The specification of the same musical fragment, but at a different pitch. See Jacobs (1988) for more details.

18. I recommend the reading of Vogt *et al.* (1982) to people interested in graphical effects used within contemporary music's scores. There is still a lot to do… unless considering only graphical formats.

Acknowledgements

I thank Claudio Beccari, who has proof-read this article and for his Italian translations of the abstract and keywords.

References

Berg, A. (1935). *Lulu-Suite*, volume 12 674. Universal Edition.

Finale (2017). *Music Notation Software*. http://www.finalemusic.com.

Forsyth, C. (1982). *Orchestration*. Dover Publications, Inc., New-York.

Garcia, F. (2012). «TEX and music: an update on TEX*muse*». *TUGboat*, **33** (2), pp. 158–164.

Goossens, M., Rahtz, S. e Mittelbach, F. (1997). *The LaTeX Graphics Companion. Illustrating Documents with TEX and PostScript*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Goossens, M., Mittelbach, F., Rahtz, S., Roegel, D. B. e Voss, H. (2009). *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts, 2ª edizione.

Gordini, T. e Liessi, D. (2014). «LilyPond: un *music engraver* integrabile con LaTeX». ArsTEXnica, **18**, pp. 97–118.

Hufflen, J.-M. (2011). «Some experience with MusiXTEX». In *Proc. EuroBachoTEX 2011 Conference*, a cura di K. Berry, J. B. Ludwichowski e T. Przechlewski. Bachotek, Poland, pp. 19–30.

— (2012). «A comparison of MusiXTEX and lilypond». In *Twenty Years After. Proc. BachoTEX 2012 Conference*, a cura di T. Przechlewski, K. Berry e J. B. Ludwichowski. Bachotek, Poland, pp. 103–108.

— (2013). «Why typesetting music is so difficult». In *Typography and Communication. Proc. EuroBachoTEX 2013 Conference*, a cura di T. Przechlewski, K. Berry e J. B. Ludwichowski. Bachotek, Poland, pp. 79–84.

— (2014). «Musical symbols at the digital age». In *What Can Typography Gain from Electronic Media? Proc. BachoTEX 2014 conference*, a cura di T. Przechlewski, K. Berry, B. Jackowski e J. B. Ludwichowski. Bachotek, Poland, pp. 17–24.

— (2015). «Musical typography's dimensions». In *Various Faces of Typography. Proc. BachoTEX 2015 conference*, a cura di T. Przechlewski, K. Berry, B. Jackowski e J. B. Ludwichowski. Bachotek, Poland, pp. 61–65.

— (2017). «History of accidentals in music». TUG*boat*, **38** (2), pp. 147–156.

Jacobs, A. (1988). *The New Penguin Dictionary of Music*. Penguin Books, 4ª edizione.

Laurie, D. (2005). *M-Tx: Music from Text. Version 0.60. User's Guide*. http://icking-music-archive.org/software/mtx/mtx060.pdf.

LilyPond (2017). *Music Notation for Everyone*. http://www.lilypond.org.

MuseScore (2017). *Créer, écouter et imprimer de magnifiques partitions*. http://www.musescore.org.

NoteEdit (2014). *A Score Editor*. http://sourceforge.net/projects/noteedit.berlios.

Odyniec, A. (2016). «Skladaj nuty! czyli śpiewnik w LaTeXu». In *Convergence: TEXu wyjdź z szafy! Proc. BachoTEX 2016*, a cura di T. Przechlewski, K. Berry, B. Jackowski e J. B. Ludwichowski. pp. 7–16.

Rimsky-Korsakov, N. A. (1964). *Principles of Orchestration*. Dover Publications, Inc., New-York. Completed posthumously by Maximilian Osseyevich Steinberg. English translation of the Russian text by Edward Agate.

Rosegarden (2013). *What is Rosegarden?* http://www.rosegardenmusic.com.

Schofer, A. e Steinbach, A. (1987). «Automatisierter notensatz mit TeX». Technical report, Rheinische Friedrich-Wilhelms-Universität, Bonn.

Simons, D. (2004). *PMX—a Preprocessor for MusiXTEX. Version 2.5*. http://icking-music-archive.org/software/pmx/pmx250.pdf.

Stravinsky, I. (1921). *Chant du rossignol. Poème symphonique pour orchestre*, volume 633. Boosey & Hawkes.

Taupin, D. (1992). «MusicTeX: Using TeX to write polyphonic or instrumental music». In *EuroTEX 1992*. pp. 257–271.

Taupin, D., Mitchell, R. e Egler, A. (2002). *MusiXTEX©. Using TEX to Write Polyphonic or Instrumental Music. Version T.104*. Part of LaTeX distribution.

Unicode (2016). *Unicode 9.0.0*. http://www.unicode.org/versions/Unicode9.0.0/.

Vogt, H., Bard, M., Bielitz, M., Haller, H.-P., Raiss, H.-P. con Seipt, A. (1982). *Neue Musik seit 1945*. Philipp Reclam, Stuttgart, 3ª edizione.

▷ Jean-Michel Hufflen
FEMTO-ST (UMR CNRS 6174) & University of Bourgogne Franche-Comté,
16, route de Gray,
25030 BESANÇON CEDEX
FRANCE
jmhuffle at femto-st dot fr