# MAPS

NUMMER 47 • VOORJAAR 2017

REDACTIE

Michael Guravage, hoofdredacteur
Wybo Dekker
Frans Goddijn
Taco Hoekwater

NTG NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van TEX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde documentopmaak in het algemeen en de ontwikkeling van 'TEX and friends' in het bijzonder. De doelstellingen probeert de NTG te realiseren door onder meer het uitwisselen van informatie, het organiseren van conferenties en symposia met betrekking tot TEX en daarmee verwante programmatuur.
De NTG biedt haar leden ondermeer:

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TEX Live'-distributie op DVD/CDROM inclusief de complete CTAN software-archieven.
- Verschillende discussielijsten (mailing lists) over TEX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- De FTP server ftp.ntg.nl waarop vele honderden megabytes aan algemeen te gebruiken 'TEX-producten' staan.
- De WWW server www.ntg.nl waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties en links naar andere TEX sites.
- Korting op (buitenlandse) TEX-conferenties en -cursussen en op het lidmaatschap van andere TEX-gebruikersgroepen.

**Lid worden** kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN- zowel als SWIFT/BIC-code en selecteer shared cost. Daarnaast dient via www.ntg.nl een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40; voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht.* Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $105.

**Afmelding** kan met ingang van het volgende kalenderjaar door opzegging per e-mail aan de penningmeester.

**MAPS bijdragen** kunt u opsturen naar maps@ntg.nl, bij voorkeur in LATEX- of ConTEXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

**Productie.** De Maps wordt gezet met behulp van een LATEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.17 en luatex 1.0.0 draaiend onder MacOS X 10.12. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

**TEX** is een door professor Donald E. Knuth ontwikkelde 'opmaaktaal' voor het letterzetten van documenten, een documentopmaaksysteem. Met TEX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in mathematische teksten.
Er is een aantal op TEX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzet-mogelijkheden van TEX. Voorbeelden zijn LATEX van Leslie Lamport, AMS-TEX van Michael Spivak, en ConTEXt van Hans Hagen.

# Contents

# Redactioneel

This issue contains six articles, three whose goal is to provide TEX functionality and three that use it. Content wise they are as diverse as can be.

At the last NTG gathering in Amsterdam Kai Eigner and Ivo Geradts from TAT Zetwerk in Utrecht gave a talk about Harfbuzz – an OpenType rendering engine. Harfbuzz, which means open type in Persian, is the preferred rendering engine for Firefox, Chrome and XeTEX. In his article Kai tells us about Harfbuzz, how it works, how it differs from ConTEXt's native rendering engine and how to invoke Harbuzz from LuaTEX.

When Frans Absil isn't teaching at the Faculty of Military Sciences in Breda he is either playing, arranging or conducting music. In his article he describes the workflow he has devised to create various types of documents about music including: score excerpts, articles, e-books and YouTube movies, and the integral role TEX plays in this process.

Hans van der Meer has contributed two very practical articles to this issue. The first describes his macro package called *Block line-up*, which aligns arbitrary blocks of content. A plethora of alignment parameters can be specified as key–value pairs in the `\setupplacex` and `\startplacex` commands.

The second describes his macro package called *Take Notes*, with which one can record, filter and typeset notes according to a variety of criteria. Notes are written in xml, which well suits their matter-of-fact content. The `<takenotes>` element controls the note selection and presentation. Attributes steer the selection according to criteria related to the attributes of individual notes.

If you enjoy a good cup of espresso you will enjoy Frans Goddijn's article. Beginning at the coffee plantation, Frans carries us, and the beans, through harvest, sale, transport, roasting, grinding, tamping and brewing, resulting in a perfect cup of coffee. Frans uses roasting software named *Artisan* that uses TEX to draw roasting profiles. Join him as he watches the roast progress toward that all important *First Crack*.

Everyone who knows Willi Egger knows his passion for fine craftsmanship. Recently he turned his skill to violin making. He recorded his experience in a book, and in this article he gives us a glimpse into how he typeset his book using ConTEXt's project structure. In particular he shows us the macros he wrote to control the placement and visibility of photos, drawings and screenshots. As a bonus, his article includes two chapters from his book.

I believe this issue of the MAPS will appeal both to your five senses and your intellect. I am sure you will enjoy reading these as much as I did. We are grateful to the authors for their contributions.

Michael Guravage

# Using HarfBuzz as OpenType engine in LuaTEX

## Introduction

When TEX was released its associated font format was the bitmap pk font format derived from TEX's companion system Metafont. Over the years TEX has been adapted to new font formats such as PostScript, TrueType, and, more recently, OpenType. With respect to this last format, two major TEX variants are currently available: XeTEX and LuaTEX. Their approach to font management is quite different. XeTEX uses external libraries available on the system such as AAT, HarfBuzz, and SIL Graphite. LuaTEX, on the other hand, can use any OpenType engine which is implemented in Lua and hooked into TEX's font mechanism. The ConTEXt system contains such an engine. ConTEXt's creator, Hans Hagen, has made this engine available outside ConTEXt by developing plainTEX code for LuaTEX that calls all relevant functionality of the OpenType engine.

HarfBuzz is a transliteration of the Persian word harf-bāz, meaning 'open type'. It is a free and open text shaping engine that renders texts for OpenType fonts, and recently also for other font formats. HarfBuzz is being active developed, and its use has become widespread. HarfBuzz is the preferred rendering engine for Firefox, Chrome, and XeTEX. Basically, HarfBuzz converts Unicode text strings into glyph indices referring to specific glyphs in the font, and their positioning instructions.

Here I will describe how I was able to couple the HarfBuzz OpenType engine to LuaTEX.[1] First, I will describe what OpenType fonts and engines are. Next, I will give some reasons why it is interesting to enrich LuaTEX with HarfBuzz. Finally, I will disclose some technical details concerning the way in which I managed to couple HarfBuzz to LuaTEX.

## OpenType fonts and engines

OpenType fonts contain glyphs that are specifications of character shapes using, for example, Bézier curves, and information describing the transformation of characters into positioned glyphs. These transformations concern either glyph positions stored in the GPOS table, or substitutions stored in the GSUB table. Glyphs are not synonymous with characters. For instance, next to the glyphs that correspond to the characters '`f`' and '`i`', many fonts also contain a glyph called **fi** ligature which corresponds to the combination of the characters '`f`' and '`i`'. In other words, there is no one-to-one correspondence between characters and glyphs. Other well-known ligatures in Latin are, for instance, the **ff**, **ffi**, **fb**, and the **ffb** ligature. OpenType engines convert texts by converting series of characters into positioned glyphs. Which glyph is chosen to represent a character depends on the font, its applied features, and the character's context. The character '`f`' may be transformed into a **f** glyph, but when the font contains the glyph called **fi** ligature, and the character '`f`' is followed by the character '`i`', and the user has chosen to apply the ligature feature of the font, the '`fi`' letter combination will be transformed into the **fi** ligature.

## Why using HarfBuzz as OpenType engine in LuaTEX

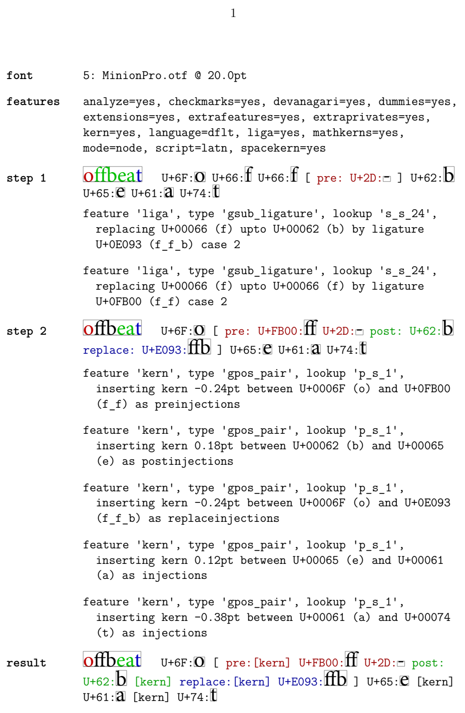
The interplay between the characters and their context, the information in the font, and the features chosen by the user is far from trivial, especially for scripts like Arabic or – even more extreme – Devanagari. It is the rendering engine which has to combine all the information and return the appropriate glyphs and their positions. Although historically TEX had no shaping engine, currently several versions of TEX have been developed which have an engine at hand. For instance, XeTEX is able to outsource the shaping to an external library. Interestingly, the 0.9999 release of XeTEX uses HarfBuzz as its OpenType rendering engine, and at present XeTEX creator Jonathan Kew is working on HarfBuzz. Behdad Esfahbod, the initiator of HarfBuzz, welcomes the association of TEX and HarfBuzz, and argues that, in the long term, 'pdfTEX's successor LuaTEX should be made to do the same thing. There is more to Unicode support than just shaping, and in those areas the TEX engines can gain a lot by building on top of existing libraries.'[2] In my opinion this remark is only partially appropriate because it passes over the (potential) power of the ConTEXt OpenType engine currently available to LuaTEX. Although it is important that LuaTEX has a powerful rendering engine at its disposal, it is not essential that this should be an external library such as HarfBuzz. At the moment, the ConTEXt OpenType engine is able to handle many different font features and scripts, and in the rare cases in which it is not yet powerful enough, it has proven to have sufficient flexibility to be adjusted appropriately. For instance, rendering Devanagari requests eccentric competence that the engine initially lacked. However, some time ago I was able to overcome this deficit by supplementing the engine with extra code that now forms part of its core. I belief that the ConTEXt OpenType engine is satisfactory as a rendering engine for LuaTEX. Still, I have developed an approach which enables the use HarfBuzz. First, I will elaborate briefly on the ConTEXt OpenType engine, what it is and how it relates to LuaTEX, and after that I will give reasons why, in my view, it is interesting to have HarfBuzz have at one's disposal as alternative engine.

ConTEXt is a system for typesetting documents build on top of LuaTEX, which is a version of TEX with a Lua scripting engine embedded. One component of ConTEXt is its OpenType engine that is completely written in Lua. This engine is designed in such way that can also be used in LuaTEX independently from ConTEXt. To do that, one should use the 'generic' mode of the ConTEXt package. In that way OpenType fonts can be used in LuaTEX. These fonts can be applied not only by specifying their font name or filename, but also by language, script, and font features. For instance,

```
\font\f = {file: MinionPro.otf: mode=node;
           language=dflt; script=latn;
           liga=yes; kern=yes;} at 20pt
```

calls the font Minion Pro, for which the language is set to default, the script to latin, and for which several font features are set, such as the implementation of kerning and ligatures. The ConTEXt OpenType engine processes this call by building a font table in Lua that can be used during the phase of rendering the text. How the engine renders text can best be made clear by giving an example created by means of the `\showotfcomposition` command in ConTEXt (see Figure 1), which displays the positioning and substitutions steps executed by the rendering engine.

The example demonstrates the implementation of the font features 'kern' (kerning) and 'liga' (ligatures) while rendering the musical term '`offbeat`'. Step by step the relevant operations are applied by the OpenType engine. Interesting in this example is the interplay with discretionaries, which are constructs in which TEX stores information about the material to be displayed at hyphenation points. Discretionaries consist of three parts called pre, post, and replace. The pre



```
                                 1

font       5: MinionPro.otf @ 20.0pt

features   analyze=yes, checkmarks=yes, devanagari=yes, dummies=yes,
           extensions=yes, extrafeatures=yes, extraprivates=yes,
           kern=yes, language=dflt, liga=yes, mathkerns=yes,
           mode=node, script=latn, spacekern=yes

step 1     offbeat  U+6F:o U+66:f U+66:f [ pre: U+2D:- ] U+62:b
           U+65:e U+61:a U+74:t

           feature 'liga', type 'gsub_ligature', lookup 's_s_24',
             replacing U+00066 (f) upto U+00062 (b) by ligature
             U+0E093 (f_f_b) case 2

           feature 'liga', type 'gsub_ligature', lookup 's_s_24',
             replacing U+00066 (f) upto U+00066 (f) by ligature
             U+0FB00 (f_f) case 2

step 2     offbeat  U+6F:o [ pre: U+FB00:ﬀ U+2D:- post: U+62:b
           replace: U+E093:ffb ] U+65:e U+61:a U+74:t

           feature 'kern', type 'gpos_pair', lookup 'p_s_1',
             inserting kern -0.24pt between U+0006F (o) and U+0FB00
             (f_f) as preinjections

           feature 'kern', type 'gpos_pair', lookup 'p_s_1',
             inserting kern 0.18pt between U+00062 (b) and U+00065
             (e) as postinjections

           feature 'kern', type 'gpos_pair', lookup 'p_s_1',
             inserting kern -0.24pt between U+0006F (o) and U+0E093
             (f_f_b) as replaceinjections

           feature 'kern', type 'gpos_pair', lookup 'p_s_1',
             inserting kern 0.12pt between U+00065 (e) and U+00061
             (a) as injections

           feature 'kern', type 'gpos_pair', lookup 'p_s_1',
             inserting kern -0.38pt between U+00061 (a) and U+00074
             (t) as injections

result     offbeat  U+6F:o [ pre:[kern] U+FB00:ﬀ U+2D:- post:
           U+62:b [kern] replace:[kern] U+E093:ffb ] U+65:e [kern]
           U+61:a [kern] U+74:t
```

**Figure 1.**

contains the material to be inserted before the line break, the post contains the material to be inserted after the line break, and the replace contains the material to be inserted if the hyphenation point isn't chosen. In simple cases the pre may contain a hyphen char while the other two are empty. However, in this example, due to the hyphenation point between **off** and **beat**, it depends on the line breaking whether the **ff**-ligature or the **ffb**-ligature will be displayed. In order to have both options available, both alternatives are built into the discretionary.

The ConTEXt OpenType engine is very powerful and flexible, and can also be used in LuaTEX independently from ConTEXt. So, why do I think that it is desirable to have HarfBuzz available as alternative engine for LuaTEX, and ConTEXt too? I will give two reasons. First, because rare cases exist for which the ConTEXt OpenType engine is not (yet) equipped to deal with, and, second, because having an alternative engine at hand is helpful for the process of developing and testing the ConTEXt engine further.

The ConTEXt OpenType engine can handle many scripts and their corresponding font features. Still, there are some scripts for which not all features are covered. As can be seen in Microsofts OpenType Specification[3] version 2, several scripts have peculiar features. Take for instance the script Syriac. Next to the regular feature ‘fina’ this script also has the features ‘fin2’ and ‘fin3’ at its disposal, which are quite eccentric. Whether they should be applied to replace the **alaph** glyph at the end of Syriac words with its appropriate form depends on certain properties of the preceding character. Actually, what is eccentric here is not the fact that the application of features depends on the properties of the neighbouring characters – that is quite common – but that instead of being stored into the font, this information has to be known by the OpenType engine. At this moment, as far as I know, the ConTEXt OpenType engine lacks this specific knowledge concerning the appropriate application of ‘fin2’ and ‘fin3’. Correct typesetting of Syriac is therefore not (yet) straightforward in ConTEXt or LuaTEX. Having HarfBuzz at hand as an alternative would overcome this lack.

Based on my experience with the flexibility of the ConTEXt OpenType engine, I expect that it will not be difficult to supplement the engine with the necessary functionality to render Syriac texts. During the past year, the engine has gone through various stages of development through which it acquired increasingly complicated skills. One, rendering Devanagari, I implemented myself. So, I believe that the ConTEXt OpenType engine will eventually meet all conceivable needs. However, my experience is that, in the process of improving the engine, it is very useful to have a reference point at hand. Although the renderings offered by HarfBuzz may not be faultless or indisputable – HarfBuzz is also still in development – they are very useful as benchmark for testing the ConTEXt OpenType engine. Therefore, it is desirable to have HarfBuzz as an alternative OpenType engine in LuaTEX and ConTEXt.

## Technical details concerning coupling HarfBuzz to LuaTEX

HarfBuzz is written in C++. Its functions can be accessed from outside using its application programming interface (API), which, in this case, is a software library. I will discuss two ways that LuaTEX is able to communicate with HarfBuzz via its API. The first is by means of SwigLib, which is a subproject of the LuaTEX project that concentrates on making external libraries available to LuaTEX using SWIG (Simplified Wrapper and Interface Generator). The second is by means of FFI (Foreign Function Interface), which is a method directly available when using LuaJitTEX. After discussing these two ways to call HarfBuzz from LuaTEX, I will discuss some of the technical details concerning the most important application of this technique, namely the use of HarfBuzz as OpenType engine in LuaTEX.

SwigLib can be regarded as a repository concerning the connection between application libraries and LuaTEX. For such connections an interface or binding between the application library and LuaTEX is needed. This interface can be created by means of so-called ‘wrapper code’ which can be generated using SWIG – a software development tool that connects programs written in C and C++ with high-level programming languages such as Lua. Recipes by means of which SWIG can produce the wrapper code for a specific application library are stored in SwigLib. In order to create the interface, the wrapper code has to be compiled. Luigi Scarso, who manages the SwigLib project, has developed the SwigLib recipe for several bindings such as those for Ghostscript and GraphicsMagick. Although he also looked at HarfBuzz, in SwigLib this binding is still in the experimental phase. (Furthermore, the material that can be found at SwigLib concerning HarfBuzz is restricted to Microsoft Windows, which happens to be not the operating system I use.) By tweaking Luigi’s recipe, and also by adding the necessary code for working with specific arrays and pointers in HarfBuzz via the binding, I managed to generate a functional interface between HarfBuzz and LuaTEX.

LuaJitTEX is a version of LuaTEX that uses LuaJIT, which is a just-in-time implementation of Lua. Due to this, it can make use of the FFI of LuaJIT to access external software libraries. The use of FFI does not involve bindings or interfaces that have to be compiled. FFI allows calling external C functions and using C data structures directly from pure Lua code. In my opinion using LuaJitTEX – which for Lua-intensive TEX runs is also much faster than LuaTEX – is preferable to using LuaTEX. However, using LuaJitTEX may have some safety risks. The FFI library is a low-level library which implies it needs to be used with care. It is not safe for use by untrusted code. In my design of the LuaJIT code by means of which I bound HarfBuzz to LuaTEX via FFI I choose to use LuaJIT functions with the same name and functionality as the Lua functions that communicate with HarfBuzz via the SwigLib binding discussed above. For instance, both in LuaTEX and LuaJitTEX, I wrote a function called ‘add_utf8’ by means of which a UTF-8 string can be delivered to HarfBuzz ready for rendering. This design of the functions enables the implementation of HarfBuzz, in the process of rendering, to be independent from the chosen binding method.[4]

To use HarfBuzz as LuaTEX’s rendering engine, one must replace the ConTEXt OpenType engine with a procedure that translates between LuaTEX and HarfBuzz via the HarfBuzz API. In principle, processing a simple series of characters in LuaTEX in this way is rather straightforward. LuaTEX passes them to HarfBuzz as UTF-8 characters, and, in return, HarfBuzz returns information about the glyph indices, i.e., which glyphs in the font have to be picked, and how they have to be positioned. In practice, however, even the simple case of a series of characters has its issues. One issue concerns glue. LuaTEX can use glue with an arbitrary width. HarfBuzz does not know this concept. Instead, it uses the space character. When calling the HarfBuzz API, some of the glues may be regarded as spaces. This identification can, for instance, be assigned depending on the glue width. In my implementation of HarfBuzz, I choose to represent glue by one space when its width is greater than zero. Similarly, the ConTEXt OpenType engine also represents such glue as a space in its OpenType rules that involve spaces. Another issue concerns attributes. In LuaTEX, characters can be enriched with information by means of so-called attributes. This is very useful, for instance, to implement colour handling. In the process of rendering, these attributes have to be preserved. However, due to the formation of ligatures and other constructs several characters can turn into just one glyph. Conversely, it is also possible that one character turns into more than one glyph. Therefore, even for a simple series of characters, some bookkeeping is required to ensure that character attributes are assigned to the corresponding glyphs.

A more complicated issue concerns the interplay between discretionaries and OpenType transformations – such as the formation of ligatures. As illustrated above (see Figure 1), the word ‘`offbeat`’ has a hyphenation point between ‘`off`’ and ‘`beat`’. Because the concrete hyphenating is not established during the stage of implementing the OpenType transformations – that happens only in the stage of line breaking which comes later – the **ff** ligature and **ffb** ligature have to be incorporated into the discretionary. Ideally, this operation of incorporating glyphs into discretionaries would be left to HarfBuzz. LuaTEX would then offer HarfBuzz something like

```
off\discretionary{-}{}{}beat
```

as input and receive something like

```
o\discretionary{ff-}{b}{ffb}eat
```

plus information about kerning in return. However, HarfBuzz has no syntax for hyphenation. Therefore, it is not possible to send and receive material that contains discretionaries. To overcome this problem, I developed a routine that rephrases material containing hyphenation points into parts free of hyphenation points covering the situations with and without hyphenations. This is not too complicated for a series of characters with only one hyphenation point. For ‘`off-beat`’ the parts would consist, on the one hand, of ‘`off-`’ and ‘`beat`’, and, on the other, ‘`offbeat`’. However, for series of characters with more than one hyphenation point, the number of parts that cover all hyphenation possibilities can become rather large. The general idea is that all these parts are presented to HarfBuzz and the result is subdivided back into discretionaries. The rationale behind this approach is that the OpenType transformations applied to characters can depend on the (series of) neighbouring characters. Because these (series of) neighbouring characters vary depending on the pending hyphenation, HarfBuzz has to be presented with all the possibilities. In the example, HarfBuzz returns **off-**, **beat**, and **offbeat** (plus information about kerning) which is rephrased in LuaTEX as

```
\discretionary{off-}{beat}{offbeat}
```

After that, a process is executed to clear out the discretionaries. As much material as possible is taken out from the discretionary. This applies, for instance, to the **o** in the example, which is present at the start of the first and second argument of the discretionary, but which can be moved to the left of it. The same holds for **eat** in the second and third argument of the same discretionary, which can be moved to the right of it. Clearing out the discretionary might not be absolutely necessary in this example: LuaTEX can handle series of characters that are completely wrapped up in discretionaries. However, in cases of more than one hyphenation point per series of characters, clearing serves a useful purpose for it can prevent the occurrence of nested discretionaries that would otherwise have to be eliminated. The result of this process is very much comparable to the result that would have been obtained from using the ConTEXt OpenType engine such as displayed at the bottom of the ‘`offbeat`’ example above (see Figure 1).

## Results and conclusion

The approach I followed to implement HarfBuzz in LuaTEX has yielded positive results. HarfBuzz performs well. The similarity between texts rendered by the ConTEXt Opentype engine and HarfBuzz surpassed my initial expectations, even for tests with complicated scripts and sophisticated fonts. Although most tests led to exactly the same results, I found a few scattered instances in which the outcome differed. For some of these differences the ConTEXt OpenType engine could be hold accountable, such as the difference in shape of the **alaph** glyph at the end of some Syrian words discussed above. However, in other cases HarfBuzz clearly dropped a stitch.

It is desirable to have HarfBuzz available as alternative engine for LuaTEX. This would enable rendering texts with features the ConTEXt OpenType engine currently does not support. More importantly, it would benefit the process of developing and testing the ConTEXt engine further. This is indeed the case, as demonstrated by improvements made to the engine based on my reports of tiny faults discovered while comparing the results of the two engines. The ConTEXt OpenType engine has proven to be a powerful and flexible engine for LuaTEX in its own right. However, it will benefit from having HarfBuzz available next to it.

## Notes

1. For source code and examples, see `https://github.com/tatzetwerk/luatex-harfbuzz`.
2. `http://behdad.org/text/` (consulted: September 2016)
3. `https://www.microsoft.com/en-us/Typography/SpecificationsOverview.aspx` (consulted: September 2016)
4. Moreover, my design of these functions is adopted from a binding called luaharfbuzz (see `https://github.com/deepakjois/luaharfbuzz`, consulted: September 2016). This binding is similar to the binding developed via SwigLib although the wrapper code is written by Deepak Jois instead of generated by means of SWIG. Consequently, next to the SwigLib and FFI binding also luaharfbuzz can be used as binding that enables my implementation of HarfBuzz as OpenType engine in LuaTEX.

Kai Eigner, TAT Zetwerk, Nederland, eigner@tatzetwerk.nl

# Music document publishing with LaTeX

**Abstract**
This article presents an overview of how to create various document types about music, such as articles, e-books and web presentations. It discusses the workflow, the set-up of a specific typesetting environment with definitions, tools and additional software.

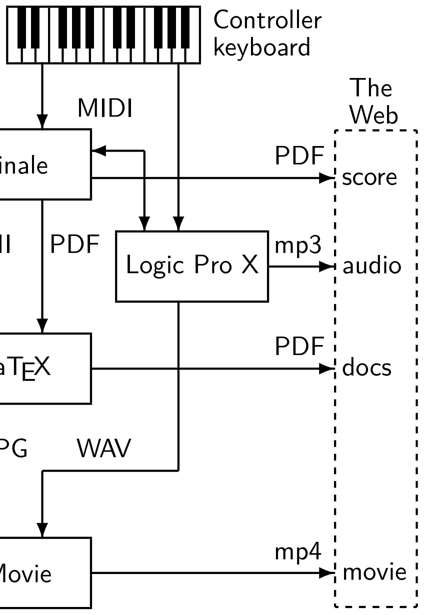


**Figure 1.** The workflow for producing music documents. LaTeX is used to generate both PDF files for the website Document Library and JPG images for producing YouTube video channel episodes.

## 1 Introduction

The author is a long-time user of TeX and LaTeX in the science research and education field, but also privately and in hobby projects. As an author I still appreciate the fundamental aspects of computer typesetting with LaTeX: separation of content and layout, re-usability, ASCII document source files and free public domain tools.

As a parttime composer, arranger and online publisher I developed a workflow for creating various types of documents about music. Since there is only limited time for these activities, convenience, reliability and efficiency are most relevant aspects for the workflow and tooling. Publishing about music involves text, graphics, musical scores and audio; this means that there is a mix of software products involved with their own file formats and import and export options.

In this article I will illustrate the workflow for various document types and show some examples. For more products and full documents, visit the website at URL http://www.fransabsil.nl.

## 2 The software toolset

The publication process is based on the workflow sketched in overview in Figure 1. This requires a set of software tools:

- Computer typesetting software LaTeX with the TeXShop shell on Mac OS X.
- The MakeMusic Finale music notation software, version 2014.5.
- Logic Pro X digital audio workstation (DAW) and sequencer software, version 10.2.4. This includes software instruments, synthesizers and Apple sample libraries. In addition there are the Native Instruments Komplete series of products (synthesizers, samplers, audio processing), some Vienna Symphonic Library samples (strings) and the Sample Modeling woodwind and brass instruments.
- Website content is created by programming HTML pages in an ASCII editor, with CSS page layout formatting and JavaScript applications.
- Video production is done with iMovie, using JPG still photographs, video fragments and the monitor window grabbing tool.
- Some graphics are programmed as PostScript source code, others are JPG or PDF export files.

The documents, audio and video files are published on the internet. PDF documents and mp3 files are part of a website media library. Mp3 audio is exported at 160 kbps medium quality stereo, a deliberate choice. The movie episodes are uploaded to the YouTube channel.

I am aware of various software alternatives, but this is what I have become familiar with over the years as an experienced user.

## 3 The recurring elements

A regular set of LaTEX packages is loaded in the document header:

- ☐ amssymb, special mathematical symbols in music definitions (see below);
- ☐ [english]babel, with the English language hyphenation patterns;
- ☐ epstopdf, conversion of graphics file formats;
- ☐ eurosym, 'We're in the € money';
- ☐ fancyhdr for specifying document header and footer labels;
- ☐ geometry for page layout;
- ☐ graphicx for including figures (score fragments) and various other graphics;
- ☐ hyperref for creating and using internal links and document metadata;
- ☐ ifthen for flag setting with Boolean variables (see below);
- ☐ palatino, font definition;
- ☐ xcolor for colouring it up.

In various document types we find recurring elements:

- ☐ The *FA logo*. This logo is a Postscript source exported as both JPG and PDF file (see below). The two musical pitches in the logo happen to be my initials.
- ☐ A set of *bibliography* files for use in BibTEX. There is a number of bibliographies for specific categories such as musical composition, arranging, analysis, harmony, counterpoint and instrumentation.
- ☐ *List environments* for examples, exercises, etc. These have a specific layout and generate appropriate numbering.
- ☐ A set of *definitions of music notation elements*, implemented as LaTEX commands, using \newcommand. These include chord structures and musical terminology for melody, counterpoint, instrumentation, document header and footer labels, etc.
- ☐ Special *diagrams* for key relationships, atonal chord structures, instrument voicings, etc.
- ☐ A number of *Boolean flags* for creating demo and full versions of e-books. Depending on the value of these flag variables, the compiler will skip document sections and replace them with blank pages, figure and table frames.
- ☐ Internal *anchors and hyperlinks* for navigating the online PDF versions of the documents. These use the hyperref package, where also metadata attributes such as title, author, subject and and a keyword list are set.

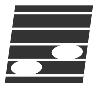


**Figure 2.** The FA logo. This PostScript source file draws a stylized slanted 5-line musical staff with two open notes at the pitches $F$ and $A$ in the treble clef.

Below is the PostScript source code of the logo design.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 35 31
%%FA Logo, fgj absil, 2000
%% Definitions
/mt {moveto} def        /lt {lineto} def
/rlt {rlineto} def      /setlw {setlinewidth} def
/mm2pt {2.835 mul} def
%% FA Logo
/drawFAlogo {
  /logo_width 100 def /logo_height 100 def
  gsave
  [0.3 0 0.04 0.3 0 0] concat
  % dark square
  newpath
  0 0 mt
  logo_width 0 rlt
  0 logo_height rlt
  logo_width neg 0 rlt
  closepath
  0.2 setgray fill
  % 5 lines
  [10 30 50 70 90] {
    /ystaffline exch def
    newpath
    0 ystaffline mt
    logo_width  0 rlt
    1 setgray
    3 setlinewidth stroke
  } forall
  % 2 notes on F and A line
  1 0.5 scale
  newpath
  logo_width 4 div 40 20 0 360 arc
  1 setgray fill
  newpath
  logo_width 4 div 3 mul 80 20 0 360 arc
  1 setgray fill
  grestore
} def
0 0 mt
drawFAlogo
showpage
```

The logo design is shown in Figure 2. Blue colour and various greyscale versions have been exported as JPG and PDF files.

```
% Example List Environment definition (from the Arranging Book)
% Counter for Example [chapter].[examplenum]
\newcounter{examplenum}[chapter]
\renewcommand\theexamplenum{\thechapter.\arabic{examplenum}}
% Environment for Example: name = example, narg=1
\newenvironment{example}[1]{% begin environment definition
   \refstepcounter{examplenum} % add 1 to example counter
   \begin{sloppypar}
   \begin{quote}
     \colorbox{blue}{% white text in blue background bar
       \textcolor{white}{\makebox[0.155\textwidth][l]{%
          \textsf{\normalsize Example~\theexamplenum}% end textsf
           } % end of makebox
         } % end of textcolor
       } % end of blue colorbox
      % end of text/makebox/textcolor/colorbox
      \par\nopagebreak[4]
      \coltxtbf{\normalsize #1} % example title in boldface blue
      \par\nopagebreak[4]
      \rmfamily % font for example body text
} % end of begin environment definition
{% begin of end environment definition
  \textcolor{blue}
    {\hspace*{\fill}\rule[6pt]{0.2\textwidth}{1pt}
       \rule[6pt]{6pt}{6pt}}%
  \end{quote}%
  \end{sloppypar}%
} % end of example environment
```

**Figure 3.** The environment definition for the examples in the *Arranging by Examples* book. It creates an indented block, topped with a blue rectangular header with white text and automatic numbering. The example title is printed in blue, the body text in black and the example is closed with a blue line and rectangle at the bottom.

A number of the specific document elements will now be demonstrated. The definition of the `list` environment for the *Arranging by Examples* book (see Section 4.3) is shown in Figure 3. This is not completely foolproof as sometimes it yields nasty pagebreaks, in particular at the closing horizontal blue line.

The usage of the `example` environment is demonstrated with a brief example from the central part of the *Arranging by Examples* book. This book chapter presents techniques for writing 5-part *sectional harmony* settings in a jazz big band style; the itemized list discusses characteristics of the solution to the given problem.

Example 3.1

**Sectional harmony in five parts, mixed voicing, ‘drop 2’.**

See the score fragment in Figure 4 with five instrumental parts (P1 to P5) and the harmony for the rhythm section (H). The lead voice P1 contains non-chordal and non-diatonic tones. The assignment is to write the parts using mixed voicing. The basic harmony is $Fm_7 - B\flat_7 - Gm_7 - C_7 - Fm_7$.

- ☐ The solution shows the ‘drop 2’ voicing as applied to a five-part saxophone section. The lead part P1 at the top is now supported by an inner voice P4.
- ☐ The bottom voice has the same intervallic relationship with the lead as in the four-part section.
- ☐ It is fairly regularly used unless the bottom voice gets into a too low register.

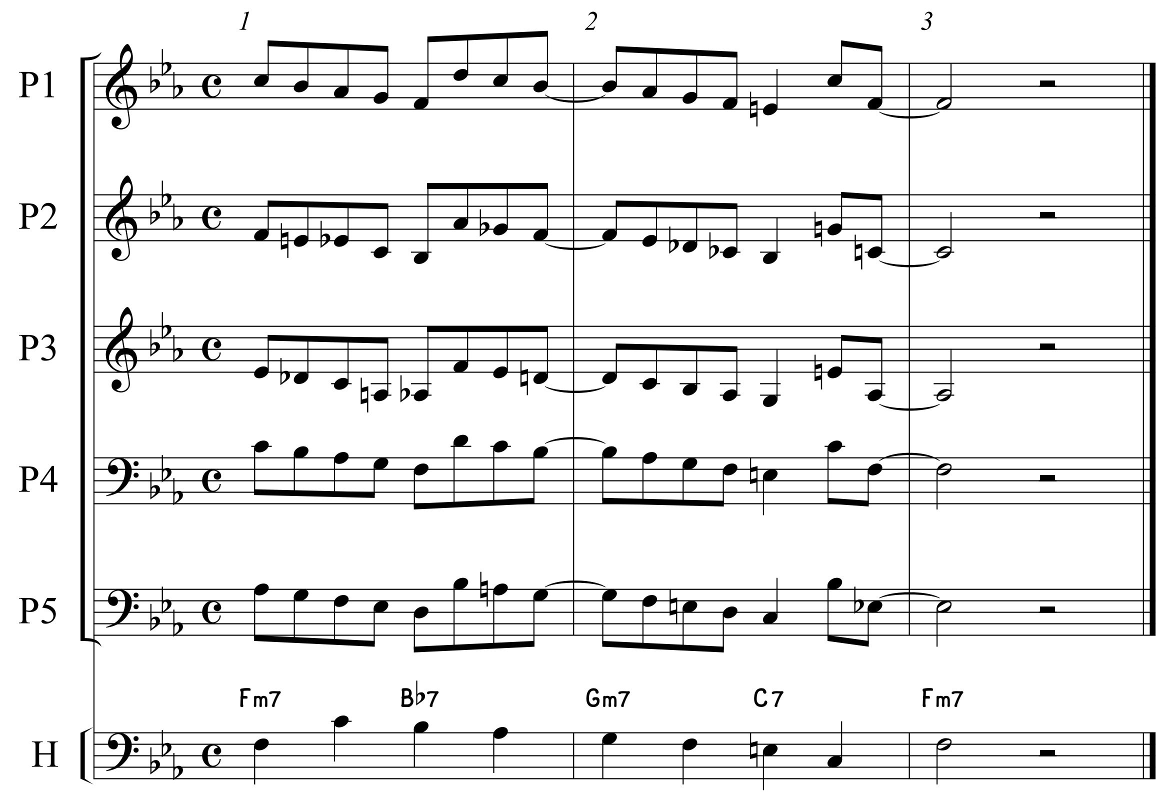


**Figure 4.** Sectional harmony in five parts. Mixed 'drop 2' voicing for given lead voice (P1) and harmony (H). This is a traditional setting for a 5-piece saxophone section in a jazz big band. (From: *Arranging by Examples: The Practical Guide to Jazz and Pop Orchestra Arranging.*)

When importing figures such as the score fragment in Figure 4, the text labels (here the part names, measure numbers and chord symbols) should have at least the size of the figure caption font. The default font size settings for expressions and staff names in the Finale music notation software are too small. There is no Finale to TEX export option; leaving out all text in the score and replacement with LaTEX fonts is possible but requires too much additional work (see the score example in Figure 7).

LaTEX commands have been defined for chord structures, melodic and other aspects. Chord structures will be entered with logical, mnemonic names for a jazz musician who is familiar with altered and extended jazz harmony. Examples of such definitions are shown in Figure 5.

Here are the calls to these commands:

```
$C\addsix$,
$B\halfdim$
$D\plusfive$,
$G\minnine$,
$A\ninesusfour$
\Seq{3}{2}{5},
\diatparch,
\chrdescmv,
\conmotop,
\dpped{b\flat},
\meter{6}{8}
```

and the printed results are $C^6$ (major triad with added 6th), $B_{\emptyset 7}$ (the half-diminished chord, with lowered 5th and minor 7th), $D_7^{\sharp 5}$ (altered dominant 7th chord with raised 5th), $G_7^{\flat 9}$ (extended dominant 7th chord with lowered 9th), $A_7^{9/\mathrm{sus4}}$ (the dominant 9th chord with suspended 4th), Seq(3×2m;$R_5$) (a chord sequence with three groups of two measures repeating at descending 5ths), $\vec{H}\|_d$ (diatonic parallel chords), $P_m^i(\searrow)$ (a chromatically descending middle voice), $\Sigma$(cm<) (opening contrary motion between top and bottom voice), $\overline{P}_D(b\flat)$, (dominant pedal point on pitch $b\flat$), $\left[{6 \atop 8}\right]$ (time signature, meter). Some definition calls require input in mathematical mode, others in text mode.

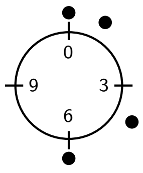
```
\newcommand\addsix{^{6}}
\newcommand\halfdim{_{\emptyset 7}}
\newcommand\plusfive{_{7}^{\sharp5}}
\newcommand\minnine{_{7}^{\flat9}}
\newcommand\ninesusfour{_{7}^{9/\mathrm{sus}4}}
\newcommand\Seq[3]{Seq($#1\!\times\!#2\mathrm{m};\!R_{#3}$)}
\newcommand\diatparch{$\vec{H}{\parallel}_{d}\,$}
\newcommand\chrdescmv{$P^{i}_{m}(\searrow)$\/}
\newcommand\conmotop{$\Sigma$(cm$<$)\/}
\newcommand\dpped[1]{$\overline{P}_{D}(#1)$}
\newcommand{\meter}[2]{$\left[ {#1 \atop #2} \right]$}
```

**Figure 5.** Definitions of chord structures and melodic aspects as LaTEX commands.

The *pitch disc* is a diagram for representing the 12 pitch classes from the chromatic scale as dots at the edge of a clock face. Below is the picture environment source code that generates this graphic element.

```
\setlength{\unitlength}{1.2mm}
% Pitch Disc template
\begin{picture}(28,25)
%\put(0,0)
  {\framebox(28,25)[tl]{}}
\thicklines
% Clock face with numbers
\put(15,15){\circle{20}}
\put(15,22){\line(0,-1){2}}
\put(14.4,18){\scriptsize 0}
\put(22,15){\line(-1,0){2}}
\put(18.3,14.4){\scriptsize 3}
\put(15,8){\line(0,1){2}}
\put(14.4,11){\scriptsize 6}
\put(8,15){\line(1,0){2}}
\put(10.6,14.4){\scriptsize 9}
% mark pitches (closed circles)
\put(15,23){\circle*{1.5}}
\put(19,21.93){\circle*{1.5}}
%\put(21.93,19){\circle*{1.5}}
%\put(23,15){\circle*{1.5}}
\put(21.93,11){\circle*{1.5}}
%\put(19,8.07){\circle*{1.5}}
\put(15,7){\circle*{1.5}}
%\put(11,8.07){\circle*{1.5}}
%\put(8.07,11){\circle*{1.5}}
%\put(7,15){\circle*{1.5}}
%\put(8.07,19){\circle*{1.5}}
%\put(11,21.93){\circle*{1.5}}
% text label
\put(1,1){4-Z15$=\{0,1,4,6\}$}
\end{picture}
```

Note that all 12 pitches are present in the template and can be displayed when required (remove comments). The result is shown in Figure 6. The circular (modulo 12) character of the pitches through octave transposition is clear. This diagram is most useful when studying pitch-class set properties and transformations in atonal music. They are used frequently in the YouTube video episodes (see Section 4.4).



4-Z15 $= \{0, 1, 4, 6\}$

**Figure 6.** The pitch disc. The dots represent the 12 pitch classes $\{0, 1, 2, \ldots, 11\}$ from the chromatic scale in the form of a clock face. Shown is the Pitch-Class Set 4-Z15, the all-interval tetrachord.

## 4 Document types

In the process various document types are created: score excerpts, articles, e-books and YouTube movies. These products are added to the Website Document Library, where they receive a HTML annotation and link, and a set of keywords for the search engine (a JavaScript tool for internal search on the website).

All documents are characterized by a very restricted use of colour. Most articles are in black and white. The e-books are limited to two colours only: the main text is black and occasionally there are blue elements, such as the internal hyperlinks for the online version, and lines, arrows and labels in figures and score excerpts. This makes printing easy and cheap. Only occasionally there are colour photographs and diagrams; multicolour never is an essential element.

### 4.1 Score excerpts

There are excerpts of full musical scores in the two categories Compositions and Arrangements of the Website. These contain approximately one third of the total set of score pages. These are exported as either Encapsulated PostScript or PDF figures from the Finale music notation software and then imported into a template LaTeX document, that generates the appropriate headers and footers with title, copyright and logo elements.

Recent Finale versions also have a reliable PDF export option for sections of score pages, such as a single staff or staff system. This is most convenient as it generates relatively small file sizes and the import in LaTeX documents allows the use of the `pdflatex` compiler. Thus compilation is much quicker than for the old approach with PostScript source files or `pstricks` figures.

### 4.2 Articles

In the Website Document Library there are articles about music composition and arranging techniques, DAW software aspects and audio signal processing. Some articles may consist of a single page diagram only; see the musical instrument range chart on the website.

Many documents contain score fragments. A simple example is shown in Figure 7. The notation in a (reduced) score with multiple staves is the typical source material for a musical analysis. The start is to annotate the printed score with pencil markings, here replaced with blue graphical elements in the LaTeX `picture` environment. First there is the harmonic analysis (see the chord symbols below the bottom staff), then other elements such as chordal functions (numbers), melody direction (arrows), and non-harmonic pitches (+-sign) are labeled.

The annotated score will be used for the description of the piece and the detailed analysis tables (see Section 4.3).

As the figure demonstrates, the graphic elements in the simple `picture` environment, such as lines, vectors, circles and Bezier curves and text in boxes are sufficiently powerful for score annotation. Correct placement requires multiple compiler runs. And once again the font size may be an issue when text is combined with musical notation.

The reduced score example in Figure 8 is the slow introduction (60 BPM) to my arrangement of this jazz standard for a studio orchestra workshop. It is part of an article about composing with two layers of chord structures in perfect fifths. The score fragment is played by tremolo strings at mezzopiano (*mp*) dynamic level, with timpani and celesta fills on the fermata chords in m. 4 and 8 (not shown here). The first violin lead carries the melody of the song. Below the score excerpt is the *tension curve* based on the Hindemith scheme, illustrating the dissonance level of these chord structures. A well-composed musical phrase will have maximum tension at approximately the Golden Section length, or, better, duration.

This example demonstrates a few additional problems with musical scores. Unless the page layout is carefully 'tuned' in Finale, i.e., typically no more than four to five measures per staff system, the total width becomes the limiting factor for the figure scaling. For a fragment from a full orchestral score with more than say 15 staves in a system, the page height sets the magnification limits. In Finale these scores are designed with a layout for a portrait orientation A4 page size or, more likely, for A3 paper size, taking into account that the conductor has to be able to read the music during rehearsal from a distance of 50 to 70 centimeters. Inclusion of such scores in a LaTeX document with appropriate page margins leads to display space conflicts.

### 4.3 E-books

Currently there are three e-books on the website:

1. *Musical Analysis: Visiting the Great Composers.* This book contains a detailed analysis of hundreds of movements from the tonal period masterworks, written between say 1700 and 1950. Composers are covered in separate chapters. For each piece there is a description, diagrams with an overall formal analysis (proportion of essential sections) and key relations (proximity to the basic key, the tonic, and remoteness during devlopment sections), and a set of tables with a measure-by-measure analysis of the harmony, melody (theme, motif, variation), compositional techniques (counterpoint, parallel, direction of motion, climax) and instrumentation. The current 5th edition contains 736 pages.
2. *Arranging by Examples: The Practical Guide to Jazz and Pop Orchestra Arranging.* This book is in three parts: Preparations, Techniques and Assembling the Piece. Based on a number of score examples there is detailed discussion of arranging techniques and aspects, especially in the middle part. The current 3rd edition is 243 pages long.
3. *A Guide to Schillinger's Theory of Harmony.* In its 2nd edition this 167 page book helps understanding the element of rhythm as it was presented by Russian theorist, composer and artist Joseph Schillinger in his *Schillinger System of Musical Composition.* This unique approach involved a lot of (integer number and set theory) mathematics and therefore has been considered a somewhat weird outlier in the field of music theory.[1]

From these e-books we will discuss a number of additional document elements. Most books consist of many chapters, tables and figures. This requires adaptation of the indentation and width of numbers in the table of contents. The settings are shown in Figure 9 (see *The*

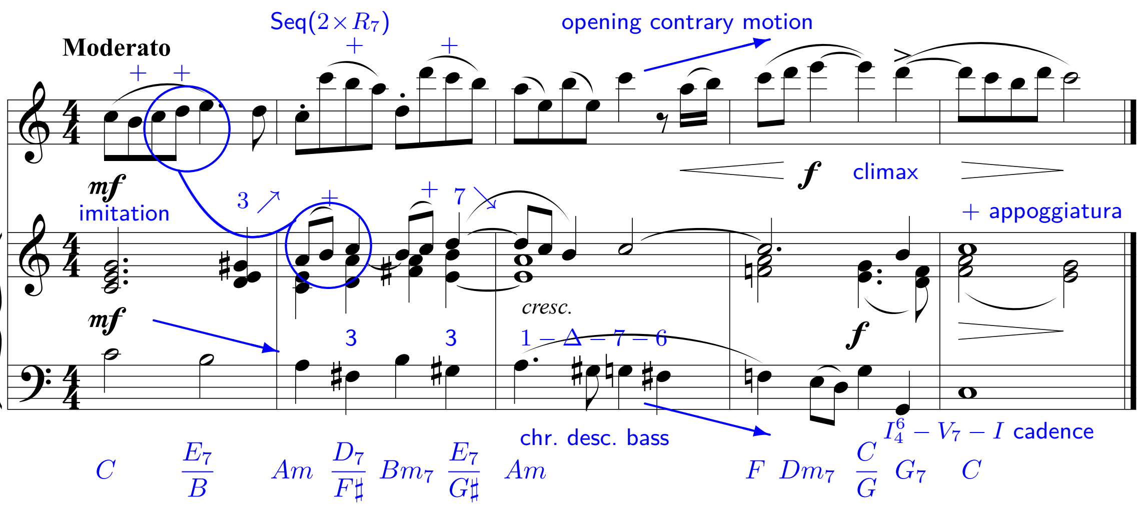


**Figure 7.** Simple score fragment with a melody in the upper staff and a harmony accompaniment. This is overlayed with annotation marks (text, symbols, arrows, Bezier curves) labeling all sorts of musical aspects of this phrase. Here the annotations are reproduced as blue elements in the LaTEX picture environment.

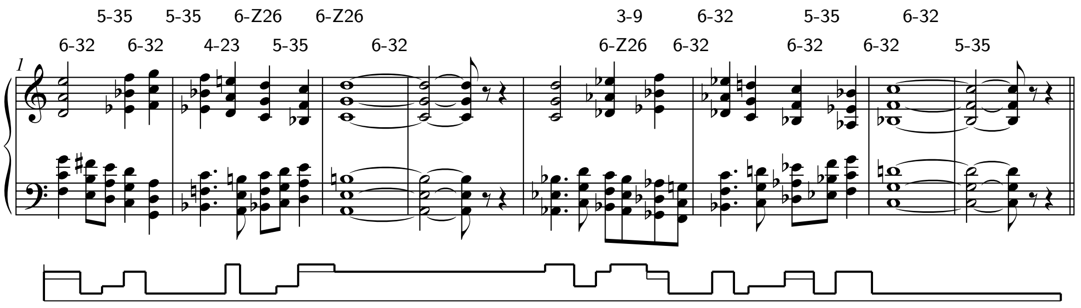

5-35 5-35 6-Z26 6-Z26 3-9 6-32 5-35 6-32
6-32 6-32 4-23 5-35 6-32 6-Z26 6-32 6-32 6-32 5-35

**Figure 8.** Tension curve. Gene de Paul, *I'll Remember April*, arrangement for studio orchestra by the author, Introduction. Pitch-class set labels are indicated above the staff. The tension level contour line is shown below the staff (reduced vertical scale). The thin line represents the Hindemith dissonance ranking correction. (From: *Harmonic Structures: Analysis and Application of Two Layers of Three-part Chord Structures in Perfect Fifths*.)

*LaTEX Companion* book, p. 52).

The books all use the recommended approach of having the chapter content in separate files. During the compilation there is frequent use of the `includeonly` command for single chapter error checking and the `draft` option in order to do a quick layout inspection and repair overfull boxes. For some documents the `hyperref` attribute `bookmarks=false` is set during the error correction process in order to prevent the compiler stopping (a well-known bug reported on the internet).

The *Musical Analysis* book contains *formal analysis* diagrams. An example with two movements from the Mozart *Symphony No. 41 in C Major KV 551, 'Jupiter'* is shown in Figure 10. The movement header has metadata from the original score such as the tempo and style (Allegro vivace), main key, meter and length. Simple diagrams in the `picture` environment with some (dashed) rectangular frames and text labels illustrate the relative proportions, such as the 'classical' exposition - development - recapitulation sections of a sonata form movement. The second movement is in ternary ABA' form, here with a coda.[2]

In the same book we also find *key relation diagrams*, such as the one from the Mahler *Symphony No. 6, 'Tragic'* in Figure 11. This is another diagram in the `picture` environment, with a regular grid of muscial key labels, line and vector segments and Bezier curves with dotted ends. The third movement, Andante moderato, starts in the main key $E\flat$ (rectangular frame around the tonic key root) and then starts to move through the keys. The diagram shows the tonic-dominant 'circle of fifths' relationship along the horizontal axis (e.g., $\ldots - C -$

```
\makeatletter
\renewcommand\l@section{\@dottedtocline{1}{1.5em}{2.5em}}
\renewcommand\l@subsection{\@dottedtocline{2}{4.0em}{2.9em}}
\renewcommand\l@figure{\@dottedtocline{1}{1.5em}{2.9em}}
\renewcommand\l@table{\l@figure}
\makeatother
```

**Figure 9.** Table of contents layout for books with many (i.e., more than 10) chapters, (sub)sections and figures. Number indentation and width at various document levels has been modified from the default values.

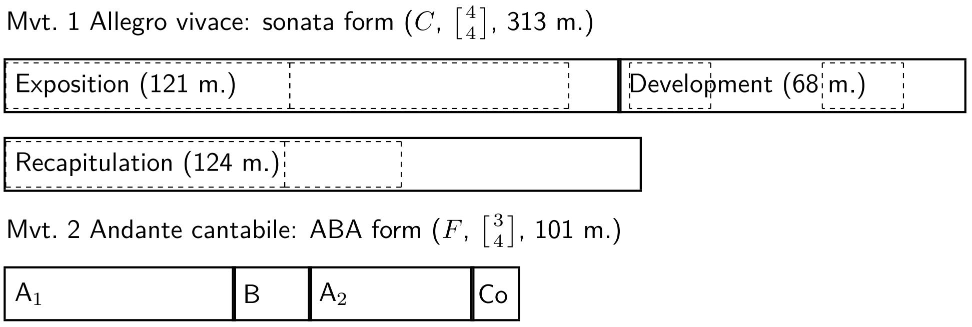

Mvt. 1 Allegro vivace: sonata form ($C$, $\left[\frac{4}{4}\right]$, 313 m.)
Exposition (121 m.)
Development (68 m.)
Recapitulation (124 m.)
Mvt. 2 Andante cantabile: ABA form ($F$, $\left[\frac{3}{4}\right]$, 101 m.)
A$_1$
B
A$_2$
Co

**Figure 10.** Mozart, Symphony No. 41 in C Major KV 551 *'Jupiter'*, Movement 1 and 2, formal analysis diagram with movement metadata and relative proportions of movement sections. (From: *Musical Analysis: Visiting the Great Composers.*)

$G-D-\ldots$) and the (sub)mediant parallel relationships along the vertical axis (e.g., $\ldots-Cm-C-Am-A-\ldots$). The number of steps from the main key position is a measure for the key remoteness. The 'octave modulo' property yields multiple key occurrences (see the remote keys $E$ and $B$). Placement of the line, arrow and curved segments requires a bit of trial-and-error for avoidance of crossing the labels and maintain legibility. Numbers in this diagram support easy reading of the path and are used in the text description. Comparing the envelope of the curve in the Mahler movement with earlier pieces from the tonal music repertoire immediately shows the more complex key patterns in the late Romantic period.

The detailed analysis results are presented as tables. The result for the simple score fragment in Figure 7 is shown in Table 1. The first column refers to the measure numbers, the second column lists the melodic material (themes such as $\mathrm{M}_{1.1}$, variations on a theme M', inverted motif I(M), etc.). The 3rd column shows the current key; this contrived example has a brief modulation from the major key $C$ to the relative parallel key $Am$ and back to the tonic key. The 4th column has the harmonic analysis in jazz chord notation (root - type - extensions - inversions), while the last column presents comments, often as 'coded' symbols.

Vertical orientation of the tables with measure numbers on a new table row runs contrary to musical notation, but is the only option to manage the content (experiments with a horizontal layout with a single measure in a table column proved too cumbersome to handle; the variable number of chords in a measure yields overfull horizontal boxes).

This example contains a chord sequence (m. 2), chromatically descending bass (m. 3), opening contrary motion (melody top part going up, bass descending), and closing climax with a second-inversion group full cadence in m. 4-5.

From the table it can be seen that horizontal spacing is essential in limiting the table width and being able to cover multiple measures in a single table row. Chord and pitch notation are in mathematical mode, which leads to wide spacing in chord progressions (see the 4th column). Most symbols in the 5th column have been defined with negative spacing in order to reduce the width.

The *Arranging by Examples* book contains *voicing diagrams*, such as the one for woodwinds shown in Figure 12. The diagram in the `picture` environment consists of open circles for each player, line and vector segments. It shows the vertical ordering of the instruments. The various options yield a different blending of sounds (timbre) and orchestral balance.

In the book *A Guide to Schillinger's Theory of Harmony* there is frequent use of alternative rhythm notation, either as integer numbers, as ticking metronomes or clocks, and as notes on a staff. Three PostScript icons help the reader to distinguish between pitch or various rhythm notations. These icons are shown in Figure 14.

The different systems of notation in the book about rhythm are illustrated in Figure 13.

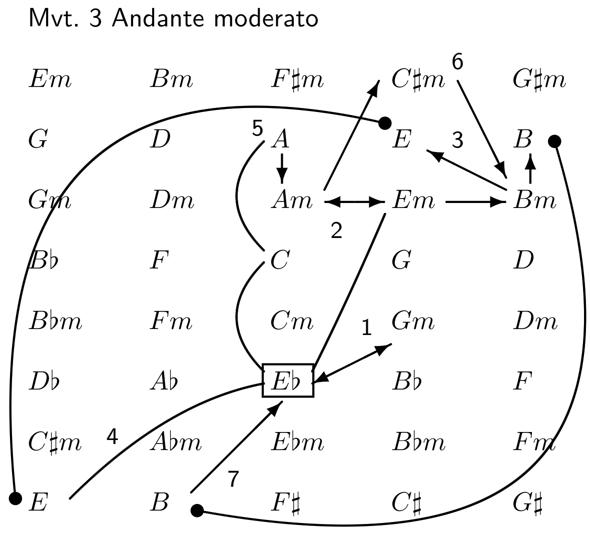


**Figure 11.** Mahler, Symphony No. 6 *'Tragic'*, Mvt. 3 Andante moderato, key relationship diagram showing the main key $E\flat$ and the modulation path in this movement. (From: *Musical Analysis: Visiting the Great Composers*.)

**Table 1.** Analysis table for the simple score example in Fig. 7.

| **m** | **M** | **R** | **H** | **Comment** |
|---|---|---|---|---|
| 1 | M | $C$ | $C - E_7/G\sharp-$ | main theme, mod |
| 2 | ⋮ | $Am$ | $Am - D_7/F\sharp - Bm - E_7/G\sharp-$ | Seq($2\times R_7$) |
| 3 | ⋮ | | $Am - A_{\Delta 7}/G\sharp - Am/G - Am^6/F\sharp-$ | $P^i_B(\searrow)$ , $\Sigma$(cm<) |
| 4 | ⋮ | $C$ | $F - Dm_7 - (C^6_4 = C/G) - G_7-$ | climax, $S^6_4$ cadence |
| 5 | $\perp$ | | $F/C - C$ | appoggiaturas |

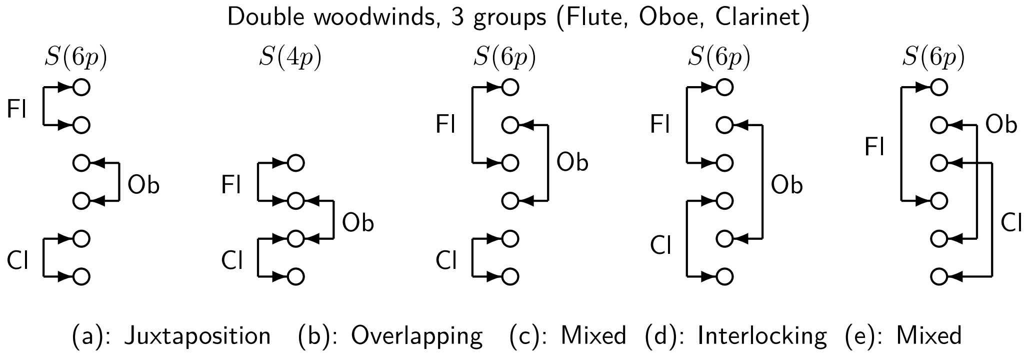


**Figure 12.** Woodwind voicing. Non-exhaustive set of diagrams for double woodwinds, demonstrating juxtaposition (overlaying), overlapping, interlocking (dovetailing), enclosing and mixed voicing. The diagram demonstrates lead (upper part) to lowest voicing options for 4- and 6-part chord structures, $S(4p), S(6p)$. (From: *Arranging by Examples: The Practical Guide to Jazz and Pop Orchestra Arranging*.)

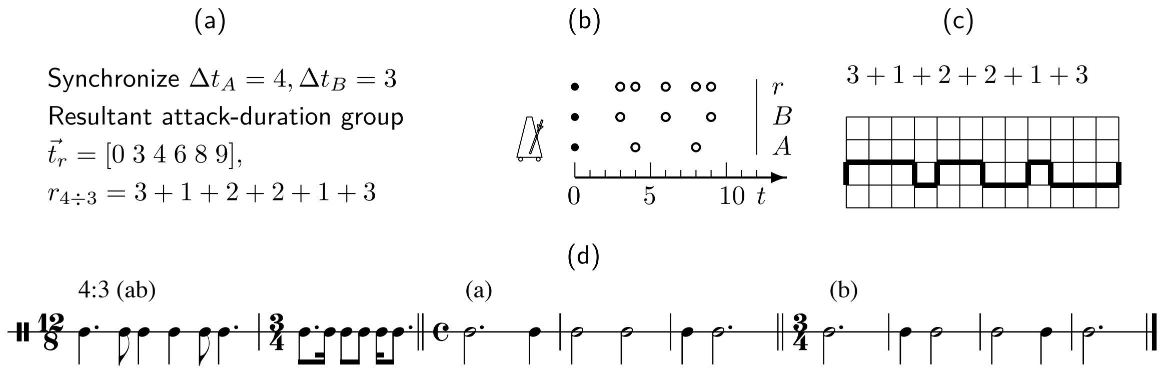

(a)
Synchronize $\Delta t_A = 4, \Delta t_B = 3$
Resultant attack-duration group
$\vec{t_r} = [0\ 3\ 4\ 6\ 8\ 9]$,
$r_{4\div 3} = 3+1+2+2+1+3$
(b)
$r$
$B$
$A$
0
5
10 $t$
(c)
$3+1+2+2+1+3$
(d)
4:3 (ab)
(a)
(b)

**Figure 13.** Non-uniform binary synchronization. The resultant $r_{a\div b}$ of two generators $A$ and $B$ that tick at different time intervals (interference pattern after starting simultaneously). $\Delta t_A = 4, \Delta t_B = 3$. The rhythm is $3t + t + 2t + 2t + t + 3t$ at time unit $t$. (a): Mathematical notation (integer numbers), (b): Timeline notation (ticking metronomes), (c): original Schillinger notation (graph paper), (d): musical rhythm notation on a single line staff for various time signatures (meter groupings). (From: *A Guide to Schillinger's Theory of Harmony*.)

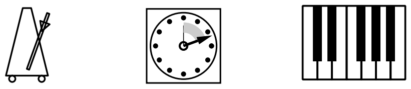


**Figure 14.** PostScript icons for rhythm notation. Shown are the metronome, clock and keyboard.

### 4.4 YouTube movies

Attracting an audience as a composer and arranger requires having an online presence. Traditionally this meant an internet website. However, these days this implies creating content on social media and video channels such as YouTube and audio files on SoundCloud or equivalent online media services.

A playlist was created on the YouTube channel, with a number of episodes about musical composition and arranging techniques. These episodes have a duration between five and ten minutes. They have a common structure: the goal and stepwise approach is announced in the introduction, then there will be the fundamentals of the technique, a section with the processing of the basic music elements and a demonstration in a couple of complete examples with reduced score and audio track.

The LaTEX typesetting tools were also involved in the making of these video productions. This requires the full workflow path from Figure 1, starting with a MIDI file and finally creating a movie for online publication. The graphics are still images created with the `beamer` package for presentations. A somewhat modified version of the `Madrid` theme is used, that has the short title and page numbers in the footer. This theme does not need navigation hyperlinks in the header, since each slide will be incorporated as a still image in a movie. The beamer class attribute option `[aspectratio=169]` adapts the slide size to HD format, but makes reading of the resulting wide lines of text cumbersome, so I stuck with the 3:4 screen ratio.

Using Apple Preview each page from the PDF presentation is exported as a high resolution 600 dpi JPG image at best conversion quality, yielding a file size between 1 and 2 Mb. See the example of a single slide image in Figure 15. It is part of an episode based on the *Schillinger System of Musical Composition* that demonstrates three techniques for key modulation using a given melody.

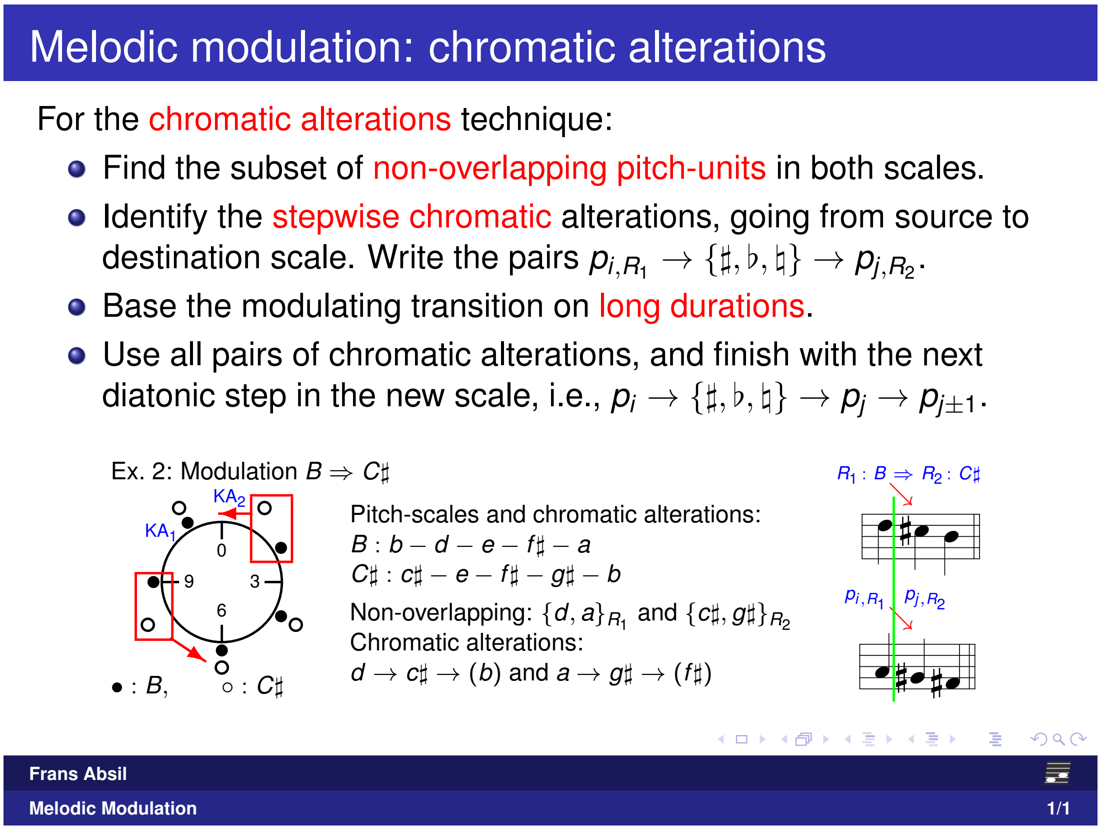
The audio content is exported from the Logic Pro X DAW; stereo WAV files are bounced at 44.1 kHz. In iMovie audio fragments are synchronized as a background track with the video stills. Movies are uploaded to the web, annotated and receive a link from the website and social media.

Currently there are 14 video episodes on YouTube with highly condensed content. Mastering the techniques will require frequent pausing and rewinding the episode, taking notes and (hours of) practicing. Game and media composers trying to widen their skill toolbox highly appreciate this playlist.

## 5 Conclusion

The workflow as described in the previous sections has been established over the years. It has been adapted as new document types were required and software components were updated. Within LaTEX there is no use of advanced features or fancy packages; the standard environment with a set of new commands for specific symbols and a few layout modifications is more than sufficient to create the books, articles and web content.

Most changes have to do with the creation or inclusion of graphics and figures. The old approach with Encapsulated PostScript figures and the `pstricks` package



## Melodic modulation: chromatic alterations

For the chromatic alterations technique:

- Find the subset of non-overlapping pitch-units in both scales.
- Identify the stepwise chromatic alterations, going from source to destination scale. Write the pairs $p_{i,R_1} \rightarrow \{\sharp, \flat, \natural\} \rightarrow p_{j,R_2}$.
- Base the modulating transition on long durations.
- Use all pairs of chromatic alterations, and finish with the next diatonic step in the new scale, i.e., $p_i \rightarrow \{\sharp, \flat, \natural\} \rightarrow p_j \rightarrow p_{j\pm1}$.

Ex. 2: Modulation $B \Rightarrow C\sharp$

Pitch-scales and chromatic alterations:
$B : b - d - e - f\sharp - a$
$C\sharp : c\sharp - e - f\sharp - g\sharp - b$
Non-overlapping: $\{d, a\}_{R_1}$ and $\{c\sharp, g\sharp\}_{R_2}$
Chromatic alterations:
$d \rightarrow c\sharp \rightarrow (b)$ and $a \rightarrow g\sharp \rightarrow (f\sharp)$

Frans Absil
Melodic Modulation 1/1

**Figure 15.** Slide for the YouTube video episode *Melodic Modulation*, as created with the beamer presentation package and converted into JPG for use in iMovie. The slide has header and footer information, and contains an itemized list, the pitch class disc image and an annotated score fragment. (From: *Melodic Modulation* movie.)

required the two-step dvitops compilation. This has been replaced by creating simple figures in the picture environment and importing PDF figures created with other software. For more complex graphics the free Inskcape software is a great alternative: it generates SVG figures and exports to PDF. Integration with LaTEX is fairly straightforward; all recent content has been created by using PDF figures and the pdflatex compiler. Problems with score page scaling have been addressed; sometimes a reference to the full PDF score on the website is the best solution.

The workflow is satisfactory and manageable; updating editions is easy by adding new chapters and sections. Long tables for detailed musical analysis are slightly problematic; column width and table height need correcting during the compilation. When modifying existing symbols the requirement is to reduce the symbol width. The compromise between typesetting 'beauty' and creating content within a limited timeframe requires a most pragmatic approach. This article illustrates the current 'design point' for creating various types of documents about classical, jazz and popular music. The author obviously remains open to suggestions for workflow improvement; the bottom line is *'Let's face the music and dance!'*

## Notes

1. The famous Berklee School of Music in Boston, MA in the US originally started as an institute working with remote learning courses based the Schillinger System. It has evolved into an internationally renowned place for training jazz and pop musicians and, nowadays, audio mixing engineers and media music composers.
2. The fourth and closing fast movement from the *'Jupiter'* symphony is a genius masterpiece, known for using intricate multiple subject counterpoint.

Frans Absil

# Block line-up

*Putting items inline or ontop*

**Abstract**
Described is a module for the placement of items either on the same horizontal line or on top of each other. Alignment and separation of the items can be varied in horizontal and vertical direction as required. Titles can be added and their location, style and color specified.

**Keywords**
combinations, line-up, placement of blocks

## Introduction

Being somewhat a do-it-yourself'er I decided to develop a macro for lining up blocks of text, figures, etc. Of course in ConTEXt this can be done with `\startcombinations`, but that didn't satisfy me. Either because my lack of understanding the full power of its parametrization, or because of its inherent architecture. Anayway, the incentive to produce something of my own became irresistable.

## Structure

The number of line-up macros is kept as low as possible. The following displays all there is and explains the general structure. Loading of the module `hvdm-plc` will not be shown in the examples.

```
\usemodule[hvdm-plc]
\setupplacex[..,..=..,..]
\startplacex[..,..=..,..]
  \startcontent[..,..=..,..] ... \stopcontent
  ...
  \startcontent[..,..=..,..] ... \stopcontent
\stopplacex
```

In illustration of the capabilities of the module the examples below will use small framed blocks containing text. But there are no specific restrictions on what can be given as content, such as figures etc.

There are two directions for placing the blocks, horizontally and vertically. The former is called up by value `inline` for the parameter `alternative`. It is the default value and therefore will be left out in subsequent examples. Placement in the vertical direction is chosen by value `ontop`. Below both cases are illustrated.

```
\startplacex[alternative=inline]
  \startcontent..\stopcontent
  \startcontent..\stopcontent
\stopplacex
```

block-1 block-2

```
\startplacex[alternative=ontop]
  \startcontent..\stopcontent
  \startcontent..\stopcontent
\stopplacex
```

block-1

block-2

Preceding and following commands can be given with parameters `before` and `after`. Below before is used to precede the blocks with a `\thinrule`.

```
\startplacex[before=\thinrule]
```

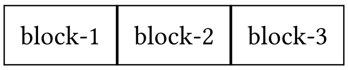
---

block-1 block-2

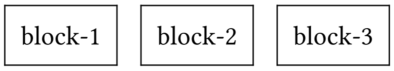
## Alignment inline

The blocks placed by inline can be horizontally aligned left, middle (default) and right. It will be no surprise that this is determined by parameter `align`. By the way, other values will raise an error as to protect the user from typos. The distance between the blocks may be varied by the `distance` parameter. This parameter can take both a dimension like 3pt or 5mm as well as the value `fill`. The latter will stretch the space between the content. In the next three examples these possibilities are illustrated.

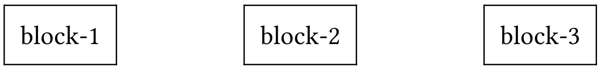
```
\startplacex[align=left,distance=0mm]
```



```
\startplacex[align=right,distance=3mm]
```

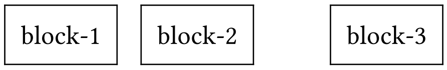


```
\startplacex[distance=fill]
```



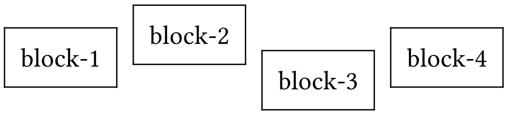
The next example of this sort shows that the distance after each content block can be specified separately. Note however that this cannot be done when the distance parameter has `fill` as its value.

```
\startplacex[align=middle,distance=3mm]
  \startcontent..\stopcontent
  \startcontent[distance=10mm]..\stopcontent
  \startcontent..\stopcontent
\stopplacex
```



Finally it is shown that individual blocks can be shifted up or down by using the parameter `shift` on `\startcontent`. A positive value shifts the block upwards, negative is downwards.

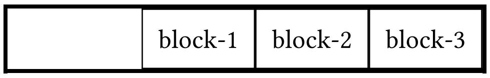
```
\startcontent..\stopcontent
\startcontent[shift=3mm]..\stopcontent
\startcontent[shift=-3mm]..\stopcontent
\startcontent..\stopcontent
```



## Alignment and width

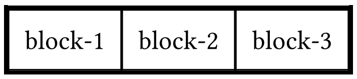
The `width` parameter has the default value empty (set by `width=`) which for inline means the current value of `\textwidth`. Another value can be set as is demonstrated below, the framing is added to make the effect clear.

```
\startplacex[width=0.8\textwidth,align=right]
```

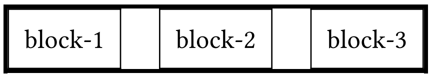


A value `width=fit` sets everything to its natural width. But note that the `align` parameter has no effect in this case. For example, centering across the page must be accomplished by the user herself with `\midaligned` or `\startalignment`.

```
\startplacex[width=fit]
```



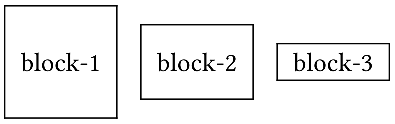
```
\startplacex[width=fit,distance=5mm]
```



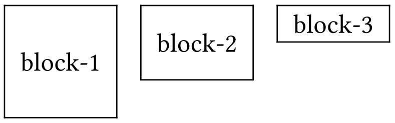
## Alignment in case of height differences

When the blocks are differing in height, there alignment in the vertical direction can be threefold. The first example shows a centered alignment, the second aligns them at the top and the third at the bottom. Thus the `location` parameter can have the values `top`, `center` (default) and `bottom`.

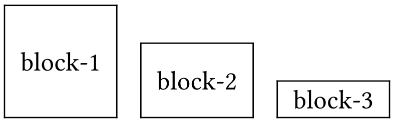
```
\startplacex[location=center,..]
```



```
\startplacex[location=top,..]
```



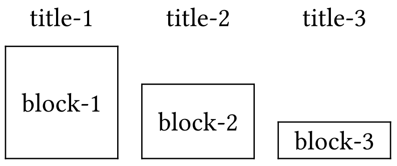
```
\startplacex[location=bottom,..]
```



## Titles for inline

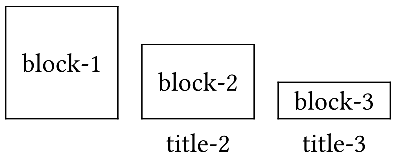
Each of the blocks placed can have a title, although by default this option is off and any title given is simply ignored. These titles are collectively placed either above or below the blocks, all in a straight row. The following two examples have the blocks aligned at the bottom and the titles above and below, respectively by setting the parameter `titlelocation` to either `top` or `bottom`.

```
\startplacex[titlelocation=top,..]
  \startcontent[title=title-1]..\stopcontent
  ..
```



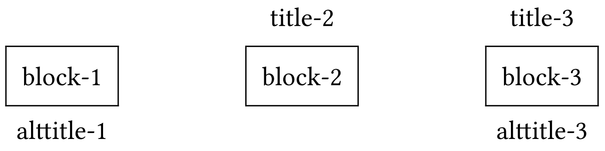
Note in the next example that some titles can be left empty.

```
\startplacex[titlelocation=bottom,..]
  \startcontent..\stopcontent
  \startcontent[title=title-2]..\stopcontent
  \startcontent[title=title-3]..\stopcontent
\stopplacex
```



Intermediate change of the location of a title is done by specifying `alttitle` instead of title. Both can be in effect at the same time.

```
\startplacex[titlelocation=top,..]
  \startcontent[alttitle=alttitle-1]..\stopcontent
  \startcontent[title=title-2]..\stopcontent
  \startcontent[title=title-3,alttitle=alttitle-3]..
\stopplacex
```



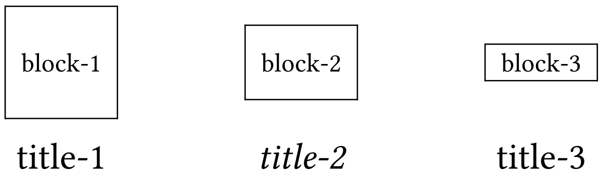
Style and color of the title are variable. Parameter `titlestyle` sets the style and `titlecolor` the title's color. They can be globally defined on the `\startplacex` or locally on the `\startcontent` as the following example demonstrates.

The following values are available for the `titlestyle` option: `bold`, `italic`, `bolditalic`, `italicbold`, `slant`, `slanted`, `boldslanted`, `slantedbold`, `smallcaps`, `oldstyle`, `mediaeval`, `normal`, `serif`, `regular`, `roman`, `sans`, `sansserif`, `mono`, `type`, `teletype`, `handwritten`, `calligraphic`, `big`, `verybig`, `heavy`, `veryheavy`, `small`, `tiny`, `smallmono`, `tinymono`. The values given here translate to appropriate font commands.

As of this moment it is not possible to put commands as `\it` in an attribute and have it executed properly. To cope with this a name in the `titlestyle` attribute not listed here, is supposed to be the name of a macro and executed as such inside a `\csname-\endcsname` pair. Thus `it` will effectively become `\it`.

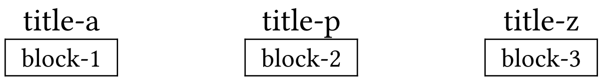
The distance between the line of titles and the blocks can be regulated with parameter `titledistance`, taking a dimension.

```
\startplacex[titlefont=14pt,titledistance=3mm,..]
  \startcontent[title=title-1]..
  \startcontent[title=title-2,titlestyle=it]..
```

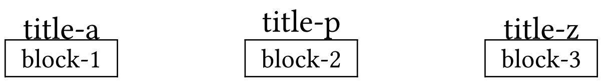


In order to avoid wobbling of the baselines of the titles, a strut is added to each title by default. The strut is placed right after the application of the `titlefont` and `titlestyle` options. Subsequent changes in font and style should be handled by the user. In case the strut is not wanted, set the value of `titlestrut` to `no`. The next example shows the effect of descenders in the title in combination with the strut respectively on and off.

```
\startplacex[titlestrut=on,titledistance=0mm,..]
```



```
\startplacex[titlestrut=off,titledistance=0mm,..]
```

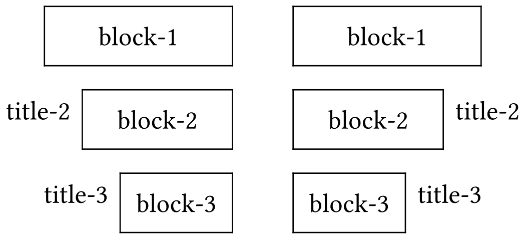


Note that by keeping the default `titlelocation=none` (leaving it empty has the same effect) or setting this on a `\startplacex`, all typesetting of titles is suppressed notwithstanding their presence in `\startcontent[ti-tle=something]`. They will appear after specifying a location for the `titleocation` parameter or giving it on `\setupplacex`.

## Alignment ontop

Remember that this placement requires that `alternative` is set to `ontop` or the use of `\startplacexontop`, because ontop is not the default. As with the inline alignment here too the blocks can be aligned at the left or the right side or in the middle (the default). Parameter `distance` sets the blocks apart in the vertical direction.

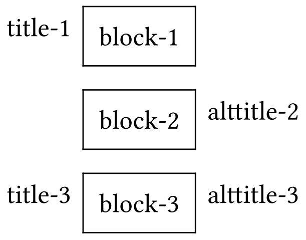
```
\startplacex[align=right,titlelocation=left,..]
\startplacex[align=left,titlelocation=right,..]
```



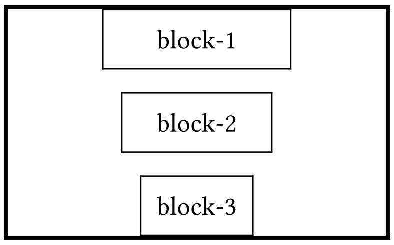
Freely placing titles to the left or to the right finds a natural restriction in the alignment of the blocks with respect to the margin of the page. An alignment to the right will force the titles to the left in spite of setting `titlelocation=right`; the same applies mutatis mutandis to the `alttitle`'s.

For a middle alignment the restriction on the placing of the titles does not apply. Here they can be placed left and right at will or even one on both sides. The relevant parameter here is `alttitle` which places the title on the side opposite the location given by `titlelocation`.

```
\startplacex[align=middle,titlelocation=left,..]
  \startcontent[title=title-1]..\stopcontent
  \startcontent[alttitle=alttitle-2]..
  \startcontent[title=title-3,alttitle=alttitle-3]..
\stopplacex
```



```
\startplacex[width=5cm,height=3cm,distance=fill,..]
```



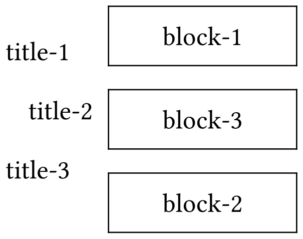
Finally above is shown that one can specify a total height and use `distance=fill` to spread the contents evenly in the vertical dimension. By default the width of the ontop placement is the general `\textwidth` of the environment, but special dimensions and the value `fit` can be given too. For a fixed width one has to take care of the horizontal alignment.

## Alignment ontop and title variants

For ontop there is somewhat more variation in title placement than for the inline configuration. The example below shows two of these variations. First the parameter `titledistance` is used both globally and locally to provide an offset between the block and the title.

Secondly, the parameter `titleposition` can be used to shift the title up and down with respect to block it belongs to. The value can vary between 0 and 1 with 0.5 as the default, by which the baseline of the title is placed halfway the height of the box. In the example this parameter is used to shift the upper and lower titles towards each other.

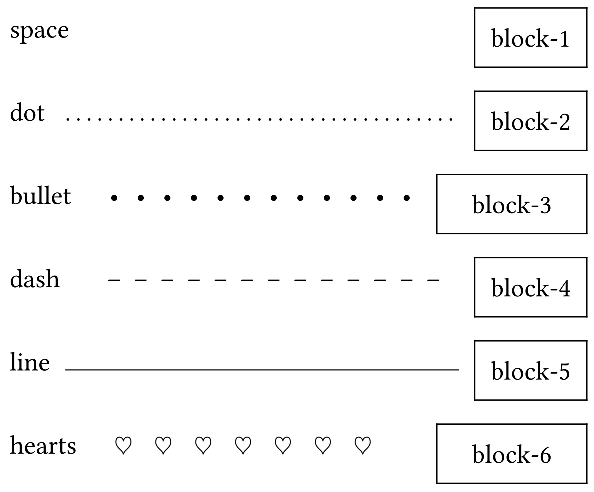
```
\startplacex[titledistance=10mm,..]
  \startcontent[title=title-1,titleposition=0.1]...
  \startcontent[title=title-2,titledistance=2mm]..
  \startcontent[title=title-1,titleposition=1.0]...
\stopplacex
```



Instead of keeping the distance between block and title constant, they might also be aligned respectively against the left or right side of the environment. The gap between the title and the block can be filled by either space, a line, dots, dashes or bullets; the default is filling with space. The `titledistance` will separate the fill on both sides from its surroundings. The parameter `titlefill` does this and can be set either globally or for each content block separately.

The last content block in the next example demonstrates the use of an arbitrary macro by giving its name as the value for `titlefill`. The macro expansion of `\heartsuit` will be placed inside a texhbox and multiplied with `\leaders`.

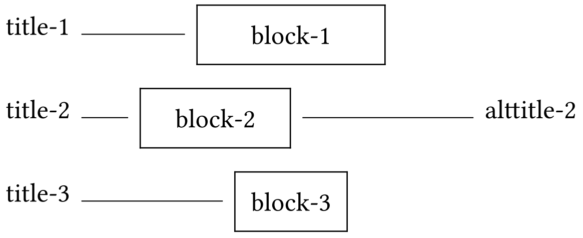
Note however, that in all cases the width of the title is not taken into account for the alignment of the content blocks in order to line them up in a consistent way. As a result, titles can protrude in the margin or overlap other elements in the text: caveat user.



```
\def\hearts{\hbox to1.5em{\heartsuit\hss}}
\startplacex[align=right,titlelocation=left,..]
  \startcontent[title=space,titlefill=space]..
  \startcontent[title=dot,titlefill=dot]..
  \startcontent[title=bullet,titlefill=bullet]..
  \startcontent[title=dash,titlefill=dash]..
  \startcontent[title=line,titlefill=line]..
  \startcontent[title=hearts,titlefill=hearts]..
\stopplacex
```

Like for inline, individual blocks can be shifted, in this case to the left or to the right. Note however that here in left/right alignment a left/right shift is prohibited too.

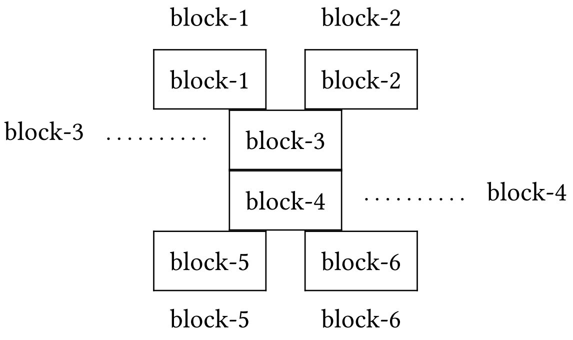
```
\startcontent[title=..,alttitle..,shift=-10mm]..
```



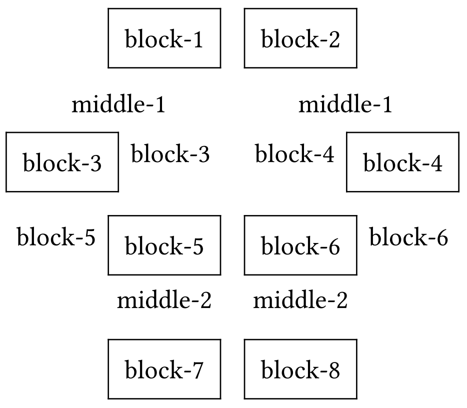
## Combined placements

Two similar combinations of placements inside each other are demonstrated in full. In the first example they are simply stacked together, in the second example they are nested inside `\startcontent..\stopcontent` pairs.

```
\startplacexontop[distance=5mm,
    titledistance=2mm,titlefill=dot]
  \startplacexinline[titlelocation=top]
    \startcontent[title=block-1]..\stopcontent
    \startcontent[title=block-2]..\stopcontent
  \stopplacexinline
  \startplacexontop[titlelocation=left]
    \startcontent[title=block-3]..\stopcontent
  \stopplacexontop
  \startplacexontop[titlelocation=right]
    \startcontent[title=block-4]..\stopcontent
  \stopplacexontop
  \startplacexinline[titlelocation=bottom]
    \startcontent[title=block-5]..\stopcontent
    \startcontent[title=block-6]..\stopcontent
  \stopplacexinline
\stopplacexontop
```



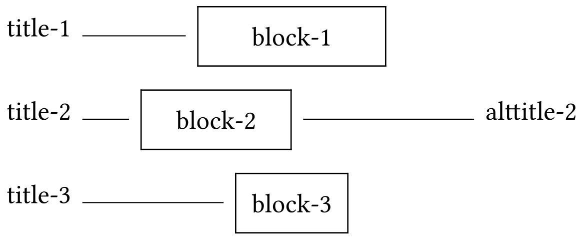
```
\startplacexontop[distance=3mm]
  \startcontent
    \startplacexinline
      \startcontent[title=block-1]..\stopcontent
      \startcontent[title=block-2]..\stopcontent
    \stopplacexinline
  \stopcontent
  \startcontent
    \startplacexinline[distance=0mm,titlelocation=top]
      \startcontent[title=middle-1]
        \startplacexontop[align=left,width=30mm]
          \startcontent[title=block-3]..\stopcontent
        \stopplacexontop
      \stopcontent
      \startcontent[title=middle-1]
        \startplacexontop[align=right,width=30mm]
          \startcontent[title=block-4]..\stopcontent
        \stopplacexontop
      \stopcontent
    \stopplacexinline
  \stopcontent
  \startcontent
   \startplacexinline[titlelocation=bottom]
         \startcontent[title=middle-2]
        \startplacexontop[align=right,width=fit]
          \startcontent[title=block-5]..\stopcontent
        \stopplacexontop
      \stopcontent
      \startcontent[title=middle-2]
        \startplacexontop[align=left,width=fit]
          \startcontent[title=block-6]..\stopcontent
        \stopplacexontop
      \stopcontent
    \stopplacexinline
  \stopcontent
  \startcontent
    \startplacexinline
      \startcontent[title=block-7]..\stopcontent
      \startcontent[title=block-8]..\stopcontent
    \stopplacexinline
  \stopcontent
\stopplacexontop
```



## XML-interface

The services of this module are available in XML through implementation of `<placex>` `<placexinline>` `<placexontop>` and `<placexcontent>`. These are sufficient for use with the XML/HTML module of this author, previously in the MAPS under the title *A bit of HTML and a bit of ConTeXt.* They take the same attributes as the corresponding macros. A previous example follows in XML. Note that here the TEX-content must be contained inside a `<tex>`-node, in order to invoke `\xmlflushcontext` instead of `\xmlflush`. When used exclusively in an XML-environment, this `<tex> ... </tex>` enclosure should be omitted.

```
<placexontop distance="3mm" titlelocation="left"
             titlefill="line">
  <placexcontent title="title-1">
    <tex>...</tex>
  </placexcontent>
  <placexcontent shift="-10mm" title="title-2"
                 alttitle="alttitle-2">
    <tex>...</tex>
  </placexcontent>
  <placexcontent title="title-3">
    <tex>...</tex>
  </placexcontent>
</placexontop>
```



## Summary

```
\setupplacex[..=..]
  same as \startplacex
```

```
\startplacex[..=..]
  alternative = inline ontop
  location = top center bottom
  align = left, middle, right
  width = DIMENSION fit \textwidth
  height = DIMENSION fit
  before = COMMAND
  after = COMMAND
  distance = DIMENSION fill
  titlelocation = none left right top bottom
  titledistance = DIMENSION 1ex
  titlecolor = COLOR
  titlefill = none space dot bullet dash line COMMAND
  titleleaders = leaders cleaders, xleaders
  titleposition = 0 .. 0.5 .. 1
  titlefont = small .. 10pt ..
  titlestyle = COMMAND
  titlestrut = yes/on no/off
```

```
\startplacexinline[..=..]
  same as \startplacex[alternative=inline,..]
```

```
\startplacexontop[..=..]
  same as \startplacex[alternative=ontop,..]
```

```
\startcontent[..=..]
  title = TEXT
  alttitle = TEXT
  shift = DIMENSION
  next inherited from \startplacex
  distance = DIMENSION
  titledistance = DIMENSION
  titlecolor = COLOR
  titlefill = none space dot bullet dash line COMMAND
  titleleaders = leaders cleaders, xleaders
  titleposition = 0 .. 0.5 .. 1
  titlefont = small .. 10pt ..
  titlestyle = COMMAND
  titlestrut = yes/on no/off
```

```
<placex attributes> ... </placex>
<placexinline attributes> ... </placexinline>
<placexontop attributes> ... </placexontop>
<placecontent attributes> ... </placecontent>
```

Hans van der Meer
H.vanderMeer@uva.nl

# Take Notes

*Notes handling module*

**Abstract**
Module for processing notes. Notes are classified according to categorie and contain information about subject, date of intake, etc. The presentation of notes can be filtered according to several criteria.

## Introduction

Active email lists, as for example the ConTEXt-list, produce many emails discussing some problem or other topics. Often these discussions form threads chaining succesive postings; where each posting incorporates much content of the previous one. Collecting all the emails for later reference produces a lot of redundancy. Moreover, the continuous repetition of previous content tends to render reading a tedious business.

Clearly, better accessibility can be attained when the threads worth retaining are condensed to a single or a few successive notes describing the gist of the discussion. This idea motivated the development of this module. It serves as a means to collect, store, select and reproduce the stored notes. Usage of this module is not restricted to storing email discussions, of course. It can be applied to everything one deems worth remembering.

## Overall structure

This module is dependent on three other modules: `hvdm-lua`, `hvdm-ctx` and `hvdm-xml`; these are loaded from within the main module file `hvdm-note`.

Notes can be stored in files and have the following general structure:

```
<note attributes>
  <subject>  ... </subject>
  <author>   ... </author>
  <short>    ... </short>
  <source>   ... </source>
  <text>     ... </text>
  <remark>   ... </remark>
</note>
```

In order to have them processed properly they should be enclosed inside an arbitrary node; which functions as for that particular tree. For example choosing <notes> as root:

```
<!-- Collected ConTeXt discussion notes -->
<notes>
  <note> ... </note>
  <note> ... </note>
  ...
</notes>
```

Notes are typeset with the following setup:

```
\usemodule[hvdm-note]
<takenotes attributes>
  <!-- Included from file -->
  <include file="notes-1.xml"/>
  <include file="notes-2.xml"/>
  ...
  <!-- Local -->
  <notes>
    <note> ... </note>
    ...
  </notes>
</takenotes>
```

## Note structure

The data pertaining to a note are split between the attributes on the `<note>` and its child nodes. Attributes are used to classify the notes and will serve as keys in the filtering of notes during the processing stage. These attributes are

- ☐ `category` – Freely chosen designation for the topic under which the subsequent notes are categorized by the user. **Must be present on each note or a TEX-error will result.**
- ☐ `subcategory` – A category can be refined into subcategories, making finer meshed searches possible.
- ☐ `date` – The date pertaining to the note; presumably the date on which the content was produced, or the date it was made. However, in case of a series of notes on historical events one may use the date attribute for the date of the event.
  Usually a date will have eight digits `yyyymmdd`. Thus 20160427 will be interpreted as 27-04-2016, but `yymmdd` is acceptable too and designates a year in the 21th century. When date has four digits only, it will be no surprise that it will be interpreted as a year.

A missing or empty date will never match true on a filter operation. The same holds for dates given as 'May 12, 1981'; such dates are printed as encountered, but not taken into account in a search.
- ☐ `key` – A number of comma separated keywords that can be used to filter out notes of a particular interest. A missing key will always match false on a filter operation.
- ☐ `language` – Signifies the language in which the content of the note is written, the `\language`-setting adapts accordingly. Legends like 'remark' etc. however, are typeset in the language of the current document. To change this behavior see the example at the end of this article. For languages other then English, Dutch, German and French you will have to add the translations yourself; it is easy enough, see the list at the end of the source code.
- ☐ `obsolete` – Signals the obsolence of the note, and is used to suppress output without the need to remove it from the dataset. If present, any value other than `no` makes the note obsolete.

The subnodes carry the information proper. These are:

- ☐ `<subject>` – Short description of the subject of the note.
- ☐ `<author>` – Author of the information or the writer of the note. More than one author node can be given.
- ☐ `<source>` – The source of the information. For example: email, internet, personal communication, etc.
- ☐ `<remark>` – Special remark to be made.
- ☐ `<text>` – The content proper of the note.

In the output the order in which the subnodes appear is fixed. Empty subnodes are suppressed, which makes it easier to work with a template.

Template for a note:

```
<note category=""
      subcategory=""
      date=""
      key=""
      language=""
      obsolete="">
<subject></subject>
<author></author>
<source></source>
<remark></remark>
<text></text>
</note>
```

### Takenotes structure

The `<takenotes>` governs the extraction and presentation from the collection of notes. Attributes steer the selection of notes according to criteria related to the attributes on the note. All these selection criteria may be specified independently from each other.

- ☐ `category` – When present, the value of this attribute is compared to the `category` attribute on each note. If they are equal, then that note is selected. The comparison of the attributes is done regardless of case.
- ☐ `date` – An attribute `from` carries the lowest date on the note that will be selected. Conversely `until` selects the last date included in the output. If both are present they constitute an interval. Dates must be formatted as four, six or eight digit values as was explained above.
- ☐ `key` – The content of this attribute will be matched to the comma separated list on the note. If there is match, than this note will be selected.
- ☐ `language` – Filter notes on having explicitely a language in the given comma separated list of twoletter language codes, for example `language="en,nl"`.
- ☐ `obsolete` – Set to `yes` in order to include obsolete notes in the output.
- ☐ `list` – This attribute is kind of special. When not empty it generates a list of the filtered notes according to the selector given in the attribute. Values for the selector are: `category`, `language`, `subject`, `author`, `source`.

Other attributes on `<takenotes>` regulate the output of the subnodes. These are the same as mentioned above: `subject`, `author`, `source`, `text`, `remark`. For example, `author="off"` will suppress the output of the `<author>` subnode. All are on by default.

There are flags to influence the output. These too are off by default.

- ☐ `framed` – Put each note into a frame, thus preventing it being broken between pages. The background can be colored by setting `backgroundcolor`. Implies `nobreak`.
- ☐ `nobreak` – Prevents breakup of a note over a page boundary by placing it in a `vbox`. The `framed` option takes precedence over `nobreak`.
- ☐ `newpage` – Start a newpage at each new note.
- ☐ `numbered` – Separate notes by numbered rules.
- ☐ `verbose` – Extra information goes into TEX's log.

Finally some embellishments. The content of the notes can be given different font and style from the rest of the document. An example of the use of a different font for the text would be to typeset programcode in a monospaced font.

The following values are available for the `style` options: `bf`, `it`, `bi`, `bs`, `sl`, `tt`, `sc`, `rm`, `ss`, `tf`, `tfa`, `tfb`, `tfc`, `tfd`, `tfx`, `tfxx`, `ttx`, `ttxx`, `os`, `hw`, `cg`, `bold`, `italic`, `bolditalic`, `italicbold`, `slant`, `slanted`, `boldslanted`, `slantedbold`, smallcaps, oldstyle, mediaeval, normal, serif, regular, roman, sans, sansserif, mono, type, teletype, handwritten, calligraphic, big, verybig, heavy, veryheavy, small, tiny, smallmono, tinymono. The values given here translate to appropriate font commands. Several font commands can be combined as for example legendstyle="bf,os", but not all combinations are effective.

- ☐ legendfont – Font used for the items.
- ☐ legendstyle – Style used for the items.
- ☐ textfont – Font used for the text.
- ☐ textstyle – Style used for the text.
- ☐ textoffset – Space between the text node and the legends above..
- ☐ textcolor – Color for the note; default is black.
- ☐ obsoletecolor – Color for an obsoleted note; default is darkgray.
- ☐ background – Effective only when framed or nobreak is chosen; default is white. Used to make the notes stand out against their environment.
- ☐ indent – The legends are not indented but for the text it can be set; default is none.
- ☐ The legends have zero whitespace but for the text it can be set; default is none.

**Example**
Example of a note in the Dutch language as designated by the attribute nl. The attribute use="de" on the <vocabulary>, however, forces the legends in German.

*Verfallen*
*Thema: Cryptografie cursus*
*Kategorie: Cryptografie/cursus*
*Datum: 1-1-2018*
*Schluessel: cryptografie,cryptoanalyse*
*Autor: Hans van der Meer*
*Bemerkung: Diese Kurs ist in Niederländisch.*

Algemene inleiding tot de cryptografie en cryptoanalyse met oefeningen voor studenten.

```
<takenotes obsolete="yes" verbose="on"
           category="cryptografie"
           from="2000-01-01" until="2099-12-31"
           legendfont="small" legendstyle="it,os"
           framed="on" textoffset="1mm">
<vocabulary use="de"/>
<notes>
<note category="Cryptografie" subcategory="cursus"
      key="cryptografie,cryptoanalyse"
      date="20180101" language="nl" obsolete="yes">
  <subject>Cryptografie cursus</subject>
  <author>Hans van der Meer</author>
  <text>
    Algemene inleiding tot de cryptografie en
    cryptoanalyse met oefeningen voor studenten.
  </text>
  <remark>
    Diese Kurs ist in Niederländisch.
  </remark>
</note>
</notes>
</takenotes>
```

Example of the output when extracting the subjects from a series of notes. Selectionfilters may be put in action while processing the data thus reducing the output to the areas of interest. Note that output related options like framed, style options, etc. are ineffective while producing a list.

1. Example of xmlstrip usage
2. Information about xmlstrip
3. Initialize startuseMPgraphic variables
4. Unresolved xmlstrip issue
5. Variables on startuseMPgraphic

```
<takenotes list="subject">
<notes>
    <note>...</note> ...
</notes>
</takenotes>
```

Hans van der Meer
H.vanderMeer@uva.nl

# Profiling Coffee / the hidden formula

*Which goes to show how little we know*

## Introduction

Caffeine may be the substance that lures you back to that cup of coffee, but once you have acquired the taste of *specialty* coffee which has been grown, harvested, dried, selected, transported, roasted, ground and extracted with care, dedication and fine equipment, it's the subtle and rich quality of the beverage that provides you with a blissful experience, again and again.

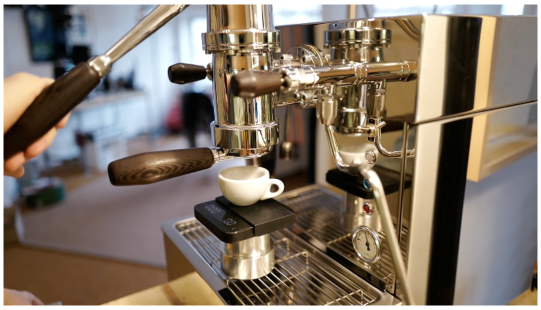


**Figure 2.** My londinium li-p home machine

**Figure 1.** Enjoying Espresso

Several processes assist the bean to develop into the best it can be.

Roasting, for instance, can be done slowly or in a hurry, with a steadily rising temperature or with a 'profile' that is continuously changing in intensity, first enveloping the bean in heat energy and then ever more slowly adding to the temperature, almost teasing the bean to finally yield its aromas in a softly crackling volumetric boost.

And luckily, deep in there, the best roast profile can be defined as a single math formula and the software that controls the roasting at my home has a TEX typesetting method built in to display that formula.

As common as grinders are, very few people know enough about it to explain the effect of the profile of cutting edges one finds on the burrs. Even though everyone can take out the burrs and look at them, their secrets are hiding in plain sight.

Extracting then, at last, an espresso from fine grinds packed into a puck, involves at least three kinds of profile: temperature, pressure and flow.

All these work together to create the satisfaction that coffee can bring to your palate and mind.

Even though coffee is one of the most common drinks we know, there's still a lot of invention going on and it's a great pleasure to be friends with a few of the original minds working at innovation.

## Simple recipe

For espresso, we take a dose of beans, say 14g, grind them fine into a basket, tamp with some pressure but not too firmly, lock the basket into a brew group, let water flow in at about 93°c, wait a few moments to let the coffee puck get all wet and apply a pressure between 4 and 9 bar to extract about 20g of delicious brew, dissolving between 18-22 percent of the coffee grinds. The high pressure can come from a little motor but I love the hand powered or spring powered lever machines. Hand power allows you to change the pressure as you go, just like you use a fine fountain pen, pressing for more flow, pulling light for a fine stream of ink or, in this case, coffee. A spring lever, once released by hand, applies a specific and beautiful pressure profile on the coffee puck and while the cooling brew head causes the temperature to slowly fall, so do the first droplets fall under high pressure and as the flow increases, pressure declines. The effect is, if all else goes well, a wonderful spectrum of taste unfolding like a fan being spread to show all the intricate patterns.

**Figure 3.** Coffee refractometer

A coffee refractometer can be used to test what percentage of the grinds have been dissolved in the water. Some like the brew with sugar, some with milk, others prefer it pure, with a tiger skin of mottled crema covering the blackness.

All very simple, it seems. But taken from the start it is quite a journey, and a pleasant one. Although I would like to be a guide, I have gradually learned that with every revealed fact there are a few more that remain elusive. All we know about coffee, all that we try to measure and pin down is the shadow of something that passed by at high speed, too fast to really see and fully understand.

**Figure 4.** Conical burrs out of a grinder

**Figure 5.** Grinding

## Tamping

When the grinds have fallen into the espresso filter basket, they are nearly ready to be rained on with hot water, but not just yet. First a tamper is used to carefully level the grinds into a firm *coffee puck* which will be firm enough to stay in place once the water hits it, but also leaving enough room for the water to quickly immerse all of the grinds which then change to become a flexible and permeable mass yielding the most pleasant coffee solubles to the water that passes through. It is important to know what tamper to use.

Jan van der Weel, a friend of mine and I do a lot of research together, and one of the projects we did for KTC, a Dutch coffee trade magazine, was about tampers. A close fitted flat tamper will do fine.

**Figure 6.** Article about tampers

**Figure 7.** Espresso

## Flow

Using a dedicated scale for coffee, one can monitor the flow of the extraction into the cup. At the start of the extraction, the pressure on the water above the puck is high but just a few droplets are falling from the filter basket. Then there is more of a flow and an increasing stream is landing in the cup. As the flow rate increases, the concentration of solubles is declining and you can see the color going from deep brown towards yellow. The richest concentration of taste is in the early very salty moments, but it needs the dilution of later, more watery fluids to reveal the most of itself in the most pleasant way. When using a spring powered lever, the pressure is highest at the start and as the spring expands and relaxes, the extraction pressure decreases; which has the advantage of working towards a mild extraction once the coffee puck is nearly spent. Many of the most expensive modern espressomachines have electronic controls to emulate more or less this pressure profile of the classic lever machines.

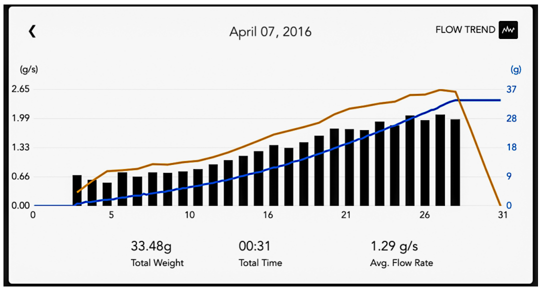


**Figure 8.** Flow

## Extraction temperature

As the hot water hits the coffee puck, the grinds quickly get wet and hot, then cool down a moment until the pressure builds up and more hot water starts to flow through the puck, dissolving (hopefully) the best compounds of the grinds, and moving out to make room for more hot water. Inserting a tiny temperature probe inside the coffee grinds we managed to log the typical temperature profile of an extraction.

The espresso machine needs to be built in a way to have hot water from the boiler to always keep the brew group at a temperature that is ideal to start the next extraction, without getting too hot or cooling down during idle hours. As simple as the copper tubing looks inside a classic espresso machine, the proper dimension of these pipes secretly enable both the right extraction temperature and the correct post-extraction temperature.

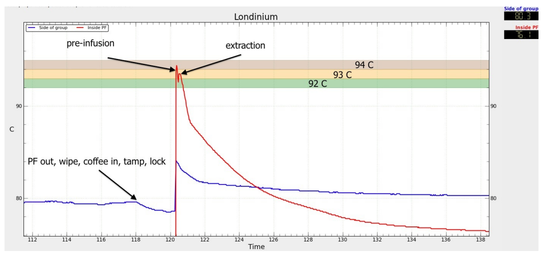


**Figure 9.** Extraction temperature profile

## Coffee plantation

I've had a few coffee plants, but in our climate they look rather boring. Deep green leathery leaves. Some friends even had flowers and then coffee cherries on theirs.

Most coffee varieties originate from Ethiopia, but today they grow in many tropical and subtropical countries; preferably around 1500m above sea level with not too much sun and some mist in the early morning.

Harvest of the coffee cherries can be once or twice a year. Picking the ripe berries is the most crucial moment in the entire journey from plantation to cup, but it is also the worst paying job to have. For as little as $ 12 a day, a picker must select only the best looking ripe red or yellow berries and leave the unripe green ones for the next round, but one gets paid per full basket so there is an incentive to be less critical and also pick some unripe berries in order to fill the basket quicker.

Next, a truckload of berries is washed in water. Any unripe or damaged beans tend to float whare they are scooped off and thrown out. The thin layer of fruit flesh around the coffee bean is partly removed by pressing the berries though the narrow of a roller mill. The rest can be washed off with lots of water after the wet mass has been left to ferment for about 24 hours. Then the wet beans are dried in the sun for a number of days until the internal moisture in the beans is about 9–11 percent.

From the moment of picking all the way to the last moments of the espresso served to you, the quality can never be improved but only damaged by any faults in the process. So harvesting is as important to our coffee as it is far away from our coffee pot.

The wet processing needs care because beans must not be damaged and fermentation may not be overdone, but a riskier method of dry processing is used in regions where water is scarce. Here the complete berries are dried in the sun during the day, covered during the night and meanwhile any bad looking berries can be sorted out and discarded. After about four weeks, the beans are finaly separated from the dried flesh. You can imagine how fruit drying in the open air in a hot climate can attract all kinds of bad luck, from birds and insects to fungi, but if all goes well, the reward can be a superior complex sweetness in the beans and an higher market value.

The dried beans are often intensely sorted and sorted again to remove any odd, damaged, discolored or insect bitten beans. It is quite incredible how much time and effort is spent on the beans by people who have never even tasted the kind of espresso that we casually order at a bar. In some countries this sorting is not done at the farm but at a washing station where less time and care is spent on selection, and then you have to sort the beans yourself before roasting. The result can be absolutely superior. It is worth all the extra work, and it makes you respect the people who do this every working day.

**Figure 10.** Sorting Mandheling beans

## Trading

The next step, selling the beans, is dependent on the options that are open to the farmer or the collective of farmers. In some countries, the state tries to regulate all trade and the harvest is sold in bulk, naturally with some kickbacks to the powerful and corrupt people involved. An excellent harvest can be indifferently mixed with inferior truckloads, and, in those cases, we never discover that lone farmer producing miraculous beans. Elsewhere, international traders manage the collection, testing, branding and transportation to huge warehouses, and the commodity is offered, sold and resold by people on computers who never see any of the product they move on their screens. The volume can differ immensely from a number of sea containers to a hundred 60k bags or microlots of a dozen bags or less.

Luckily there is a growing network of roasters and coffee businesses who connect with farmers directly. Buyers and sellers work together to improve coffee quality, and to ensure that the farmers and their workers get better pay. Investments are made to help the community, and in some regions to improve the way the small family businesses are organized. In a macho culture, for instance, just paying more for the harvest can be useless if the father goes to town to sell his beans and then when he has more money, he stays in town longer to party, returning home a poor as he left. Also, some farmers fly out themselves and visit buyers and roasters to hand out samples and book direct orders. Last year, Marianela Montero of Costa Rica visted me in Amsterdam. Her family owns a coffee plantation, and she has travelled most of the world in between harvests to learn about the international trade. Travelling the world is not new to Marianela since she has competed intranationally as a swimmer. Her father, coffee producer Don Carlos Montero, is also a swimming coach and when Marianela was younger, he built her a competition grade swimming pool at home so she had perfect training facilities. At a trade convention in Seattle she noticed that some big international volume traders took her presence very seriously as she was one of the first from her region to connect directly with many buyers on the other end of the long and complex trade trajectory.

[redacted]

Figure 11. Marianela Montero visiting the author

Roasters reaching out to farmers and helping them improve their life and business is not merely a fair trade move by the roasters. It also builds some protection against the big brands who might otherwise buy up complete harvests of the farms that a small roaster has discovered. Nespresso for instance make so much money on their cups, they can easily afford to pay whatever the currently popular beans would cost, if only just to prevent any comptetition from growing to a significant size.

## Cupping

At several moments along the itinerary of the beans, the quality and market value has to be established, and because sophisticated roasting devices, grinders and espresso machines are scarce in most producing parts of the world, this is done by a ritual called *cupping*. First the green beans are roasted in a small, somewhat crude roaster and the next day these beans are ground coarsely.

[redacted]

Figure 12. A cupping session

[redacted]

Figure 13. Late night cupping discussion hosted by Kees Kraakman (r) in Amsterdam. Center is Adam Craig from New York, presently coffee enrepeneur in Amsterdam

The grinds are smelled, distributed over cups and immersed in hot water. The freshly roasted coffee grinds emanate carbon dioxide and come floating on top of the hot water. This crust is broken and the aroma rising from the water is sniffed and noted by all in attendance.

Next, the floating coffee particles are scooped off and removed and deep spoons are used to slurp the coffee. It takes much training to be able to uniformly taste and evaluate the coffee this way.

This process called *grading* has been more or less standardized and there are over 3,500 licensed *Q graders* who provide credible and internationally accepted descriptions of taste and quality. A coffee that scores higher than 90 is considered excellent.

## Monsooned

On their way, the beans are mostly packed in sealed plastic food safe bags in a coat of burlap bag with the basic information about trader and origin stamped on the burlap. This wrapping ensures that the beans retain the freshness they had when leaving the farm.

There are some exceptions. More than a century ago, beans from the Malabar coast in the British Indies were transported to Europe by ship, and during the journey of several months, rainstorms would wet the beans and hot spells would dry the cargo. This resulted in the beans arriving in a weathered state a bit pale and larger than normal. Everyone learned to love their specific roasted taste quality of musty spice and chocolate so much that nowadays, when transportation is fast and sophisticated, the beans are purposely exposed to monsoon rains and drying cycles in order to acquire that *Monsooned Malabar* finesse.

Figure 14. Monsooned Malabar beans from India

After a short stay in warehouses in harbor cities, the beans are delivered to the roasters and that's where the real fun begins for us. Typically, roasting can take any time between 3 and 20 minutes, and during that time a dazzling amount of changes take place inside the bean, most of these in the last 25 percent of the roast.

Figure 15. Green bean before roast

## Roasting machines

You can roast the beans in several ways. The basic thing is to carefully add head, and keep the mass of beans in motion so every bean is getting an equal heat treatment.

Heat and movement can be delivered to the beans in several ways.

Most machines have a metal outer drum that is heated by flame or electrical heating elements. An inner drum, similar to that in a washing machine, continually tumbles the beans. The heavy mass of the metal radiates heat into the bean mass of several kilograms or even dozens of kilograms that is sloshing around in the inner drum and air is blown through the middle drum to clear out smoke and to remove the chaff of silver skin that comes off the beans during the roast.

With *fluid bed roasting* the magic comes in the form of hot air flowing into a vertical roast chamber from the bottom up, pushing through the beans, with the beans dancing around on top of the bed of hot air. This needs a lot of energy from the heat source (gas or electricity) and enough *push* in the flow to keep the beans aloft without blowing them out of the chimney altogether. For instance, when my friend Tije de Jong and I tried to build such a roaster, we used a leaf blower and it blew the beans all through the building, but when we fixed the heat nozzle on it, the blower pushed mostly backwards. The blower had more speed than push.

Mike Sivetz, a North-American legend of the coffee world, built a first fluid bed roaster in 1975 and many of his machines are still in use. Today, the Fracino factories in Birmingham build a small fluid bed roaster for home use and that is what I have today, with some home made extensions.

Figure 16. Roastilino *fluid bed* roaster

More than a century ago, in the early morning, the smell of freshly roasted coffee flowed throught the narrow streets of fast growing cities. Then industrial roasters took over, and home roasting became all but obsolete. Today, the art and craft of coffee roasting is recalled and innovated.

Modern roasting machines mix elements of fluid bed and rolling drum but in 2015, Tije de Jong in Amsterdam came up with a radical new and amazingly simple idea. He built a roaster out of scrap parts with a household sieve to hold and shake the beans in and a paint stripper to heat the beans, using a probe inside the beans to monitor the bean temperature. This roaster is cheap and easy to copy, and it could enable many home roasters to rediscover the art of roasting.

A paint stripper serves as a heat source and a little electric motor does the shaking. We tried several paint strippers and the simple one performed just as well as an expensive one with digital display. Reading the bean temperature and watching a timer, Tije notes the roast values in a matrix printed on paper and at the end of the roast he connects the dots to reveal the profile.

It was not the first time Tije used his extensive workshop for a coffee project. In 2014, Tije built a *transparent* *portafilter*, a little compartment that enabled us to see what happens to the coffee grinds during the extraction. Quite a challenge to find and combine synthetic and metal parts to hold together under vast changes in pressure and temperature with sudden variance of nearly 100°C and shocks from atmospheric pressure to 9 bar. Roemer Overdiep, a Dutch designer and cameraman, rented a *high speed camera* and filmed the events in detail. The video clips were an instant hit on the coffee channels with tens of thousands of views as no one had been able to visualize at such resolution what actually happens inside the normally hermetic brew head.

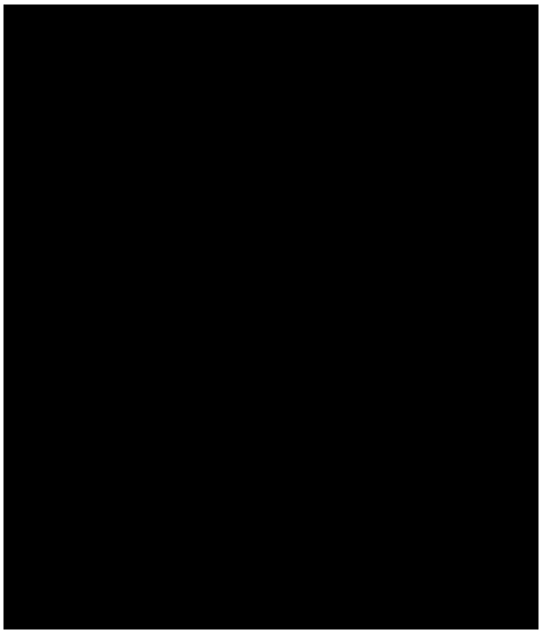


**Figure 17.** Tije's roaster featured in an online coffee magazine

**Figure 18.** Gábor and Attila, coffee friends from Budapest, next to version 2.0 of Tije's *Shake, Not Stir* roaster

## Roasting

Inside any type of roaster the beans go through a basic experience, a kind of roller coaster ride of an ever increasing temperature, but the rate at which the temperature increases, the so called *Rate of Rise*, can vary depending on the ideas of the roast operator.

You can divide the roast trip of the beans in three phases.

The first and easy part of a roast track is the ramp up to around 150°C. Not much is happening yet besides the beans losing moisture to the hot air, so this is called the *Drying phase*. With some imagination, if you smell the air coming out of the roaster, you are reminded of freshly cut grass, hay, a horse's stable, sometimes even a sweaty horse brought in on a hot summer's evening when a thunderstorm has broke through. The beans change from green to yellowish in color.

If this first part is kept too short and intense, the resulting coffee may be winning in clarity or lacking in body, tasting a bit sour. If it is done more slowly, a more full bodied coffee might be won, but if overdone the brew could taste dull.

Then you raise the heat enough for the beans to gradually start to change color. Browning can be seen in the *Maillard phase*; which is the same event involving amino acids and sugars that gives flavor to steaks, cookies and bread.

If kept short, this might contribute to a simple bright coffee and if prolonged, this phase may bring about a more complex coffee; which could be a good thing if you were expecting much from the deliciously fragrant green beans.

The smell in this middle phase is like toasted bread. During this time, more and more subtle changes take place inside the bean, but it all builds up towards the most spectacular moments at the start of phase three:

The *Development phase* begins with an audible *first crack*. The remaining moisture inside the cells of the coffee bean is heated above boiling point, and all these cells become microscopically small pressure cookers kickstarting a large number of chemical reactions that contribute to the tastes and aromas we associate with coffee. Almost like minuscule popcorn, first a few and then many of these immensely small cells inside the bean begin to pop, and much of the built up pressure and heat bursts out.

Imagine a hot room in a turkish sauna bath where it has become unbearably hot and suddenly the doors and windows fly open and hot steam is blowing out. Standing outside, you'd feel the heat rush coming at you and you could think that the little hut got very hot all of a sudden, when in fact the build up of heat was gradual and only now it is flowing outward.

You might smell *coffee* even though it's more of a sharp smell that I associate with *coffee roastery*. There can also be a *Second Crack* later on but that is less audible and if you roast that far, you also approach the phase of full carbonization of the bean, and you are basically roasting charcoal and starting a fire.

During this third phase after *First Crack*, The beans are turning a darker brown and many different volatile compounds are carried from the beans and out of the roaster. The roastmaster often takes samples during this time, looking at the bean colour, smelling the beans, trying to decide about the best moment to stop the roast and cool the beans. Overall, this *development phase* takes around 25 percent of the total time, so if he knows when *First Crack* started, he will roughly be able to estimate when the beans will have the desired roasted quality, but this also depends on the heat applied towards the end, on the airflow through the bean mass and the motion of the beans inside the roast chamber.

If this phase is lengthened, a more wonderful complexity and a creamy mouthfeel could be the result but if overdone, you have wasted the beans and the coffee will taste flat. Kept too short, this phase would allow more sweetness, possibly at the cost of being just simpler in taste of maybe even like chewing on grass instead of coffee.

Then the door is opened, and the fuming hot beans rush out to be cooled. It's important to get enough cool air on the beans to make the roast stop, but a thermal shock to room temperature is not needed. The mass of beans can be stirred or left resting, some roastmasters even spray water or water with sugar on the beans. All that is done will affect the taste of the coffee later on.

## Controlling

I built the controller for my roaster with the help of Wa'il al-Wohaibi from Ryadh in Saudi Arabia. He sent me many tips and pointers for the hardware. At the heart of the device is a PID module. It is a data bridge beween the computer and the roasting machine. The computer tells the PID controller how to roast, the controller drives the heating element in the roasting machine to stay on track. At the same time the controller measures real time temperature inside the bean mass and communicates these data to the computer where a program called *Artisan* processes the information. *Artisan* displays the roast profile as a graph of temperature over time, and looks ahead on the roast plan to provide the controller with the latest temperature targets. It automatically marks the roast phases, and even uses computer speech over the speakers to alert the operator about the events.

**Figure 19.** Wa'il al-Wohaibi talking about coffee roasting on a Saudi television talk show

The *Artisan* roast software that I use is written and developed by Marko Luther from Fürstenfeldbruck in Germany. It is the only piece of roasting software that both loggs what happens while, at the same time, controlling the process like an *autopilot* or *cruise control*.

Many players in the coffee world keep their knowledge and methods to themselves or share these in very small portions during expensive training, but there is also a growing network of friends and colleagues who freely share all they do and know. The Artisan software is open source and free, just like TEX and in fact there is a kernel of TEX in the roast program.

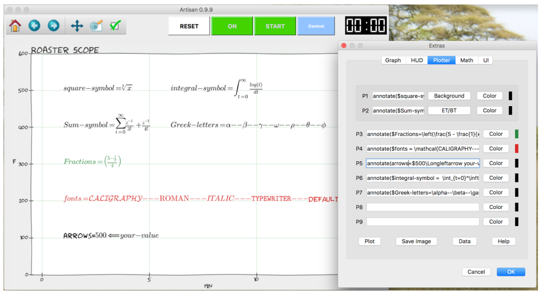


**Figure 20.** Look! TEX sighting in coffee software

Together with a friend, Marko Luther has also built a full-spectrum color sensor with 4 white LEDs, a 7-segment display and an Arduino micro controller to measure the roast color of coffee grinds. Usually these cost several thousands of Euros but they managed to produce this open-source device for a few hundred, using a 3D printer; the result of combining low cost high quality components and many many hours of work.

**Figure 21.** Tonino roast color meter, hardware and software (open source) by Marko Luther

## Designing a roast

Before roasting a new bean, I measure the moisure of the bean, its density, weight and size. This way I can predict if the beans will easily dance on the hot air, or if the airflow might have trouble permeating the bean mass. If, for instance, the beans have a moisture content of 9.7%, I look in my logs to see if I roasted a bean like that before. If so, I can estimate how long the roast could be, at what point *First Crack* might occur, and then the proportions of the phases become apparent.

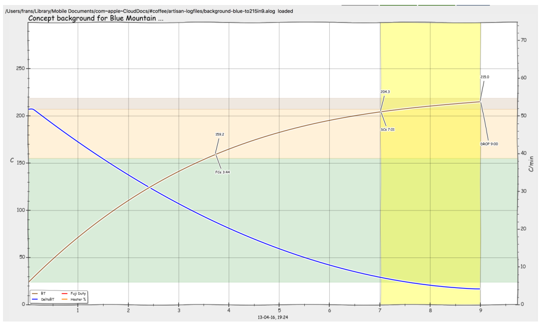


**Figure 22.** A typical design of a roast profile of mine

In my roaster I aim for a fast start followed by a gradual decline of the *Rate of Rise* (the blue line). The brown curve is the planned *Bean Temperature*. The green area of the temperature is where mostly drying takes place, the pink level is between 155°C and *First Crack* and the grey level is the temperature area where the *development* happens. On the time line, the *development phase* is the vertical yellow bar.

I want a high Rate of Rise at the very start that keeps gradually declining towards the last moments of the roast when it is ideally near zero. This way, there is a constant flow of extra energy towards the beans, even at the moment of *First Crack* when the pores fly open and hot aromatic fumes are squirting out of the popping bean cells. At no time is the energy buildup allowed to sag. It is merely increased ever more gently towards the end to support the process inside the bean during which all the complex taste and aroma is created.

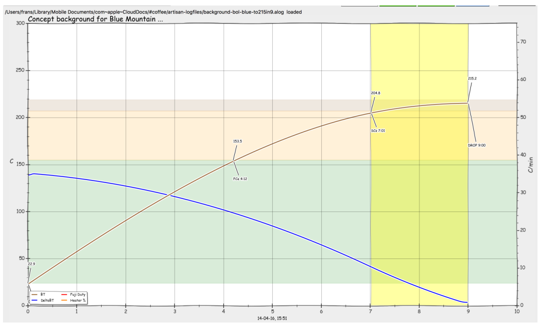


**Figure 23.** A more *traditional* profile

Most roasters use a profile like the one above. The ramp up to *First Crack* is almost a straight line so the *Rate of Rise* is fairly constant. Most large drum roasters have such a heavy metal mass that it is not very easy to influence once it is going. Like a big heavy truck going up a hill. Around *First Crack* much energy is released from the bean mass and there, the profile mostly turns to a less steep rise. Some roasters boost the energy towards the end to quickly get a certain targeted deep brown color of the beans.

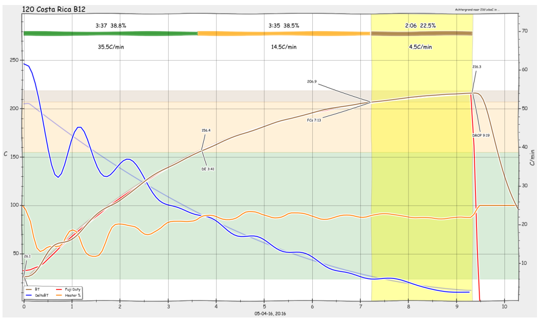


**Figure 24.** An actual roast log

In figure 24 the brown line of the actual roast is practically covering the red target curve. The orange line shows the intensity of the heater controlled by the PID system and the dark blue line the actual *Rate of Rise* value meanders ever more closely over the light blue target line. *Rate of Rise* was first an average of 35.5°C/min, then 14.5°C/min and a mere 4.5°C/min during the last phase. Both the *Drying phase* and the *Maillard phase* have been about 38% of the total roast and the *Development phase* was 22.5%

After a rest of ten days, the espresso from this roast was boring, but it gradually improved during the week after that, and then it really spread its wings to be most enjoyable.

In my experience this is a good way to go from bean to cup but who knows, I might think differently later on. Figuring out how to do it is like finding your way in a town house with the floor plan of a different building in your mind. You bump your head often but the more you browse around, the more you discover.

> "After winning, I realized how little I knew" said Gwilym Davies in 2014 when he was interviewed in his barista training centre near Prague, five years after becoming the World Barista Champion. "I realized that none of us knew that much which made me much more confident and I was in the wonderful position where I had proved myself. I was allowed to say *I don't know!* I was allowed to make mistakes because I was the world champion. I'd proved myself and it was okay to say that I'm still learning. I didn't have to feel like my identity was being challenged because I didn't know something, I didn't have to make something up. I could just be honest and go *Hmm, I don't know…* It's fun."

Recently Talor Browne, a very experienced roaster and coffee expert, mentioned in an interview how even some of the most famous coffee gurus are still searching:

> "Even at the top, there is no known knowledge, only trial and error. Everyone, even Tim, is kind of bumbling along in the dark, looking for the light switch. It bothers me to see people be looked up to so much when it's all so unknown. The things I do know are: if your raw coffee is delicious, it's actually pretty hard to mess it up."
>
> Marko Luther commented:
>
> "Nice quote. I couldn't express this better and share Talor's view fully. I had that impression already in 2013 when I talked to Morten and Tim in person at the European SCAE convention in Nice.
>
> So there seems to be a clear change in taste on modification of the roasting process, but it seems impossible to grab the exact details of this influence and use these to make a tool that brings out what you are looking for in a coffee.
>
> Even worse, I have the feeling that coffee changes dramatically and without much control or predictability during the time after roast and by small modifications of the drink preparation process.
>
> All we have by now are only some rather rough general wisdoms.
>
> Brewing hotter or extracting longer tends to make your brew more bitter, while extracting cooler and extracting shorter tends to move to the bright and sour side.
>
> The same in roasting. Shorter, lighter roasts might be brighter, while darker roasts turn towards bitter. In the middle there must be the *sweet spot*.
>
> Then we know that there is the danger of underdeveloping a roast (still green in the middle), especially with gentle, but short roasts. Hotter short roasts are better developed, as are longer roasts. Too gentle and long roasts could end up boring.
>
> But why are some roasts that tasted horrible a week out of the roaster, excellent if tasted after 3 months?
>
> And why is my grinder choking up the shot every now and then, without me changing anything in bean choice or grind setting.
>
> Sure, changes in humidity might be the reason. But humidity is not changing too much during winter times in my kitchen.
>
> I am lost, but others too."

That is maybe the best: being blissfully lost amidst the most delicious coffee.

**Figure 25.** Roasted bean

Meanwhile, the roasted bean is smiling at us, mysteriously, with a self satisfied gleam on its cheeks. We will be studying it for a while yet.

Frans Goddijn

**Figure 26.** Do you like sugar with that?

# Violin making

*Setting up ConTEXt for typesetting the book*

**Abstract**
Woodworking is one of my passions. The project of making my own violin is some kind of crown to the whole development. Throughout the violin making lessons notes were made, sketches drawn and photos taken. At home all sketches were turned into drawings. All information is put together in a ConTEXt project from which it is possible to compile/typeset a book. This article describes the setup of the book in ConTEXt, shows the functioning of some macros and presents two chapters of the book.

## Introduction

From youth I have been involved in woodworking. When it first was related to model-making it turned into furniture making, restoring doors and door-frames, restoring houses. In recent years I started woodturning and making tools and jigs in order to facilitate wood working. – I was and still am intrigued by the luthier's work. After having followed a course in building a hurdy-gurdy, from which the resonance body was already built upon arrival, the wish to make an instrument from scratch grew. – In 2015 I turned 60. I decided to live a special, crazy and fanciful year. The project was to learn how to build a violin. – Okay why a violin, the queen of instruments, an instrument really difficult to make? – Friends who are familiar with the luthier's craft tell me that, once you are able to build a violin, you can build any other stringed instrument. And the skills I have learned can easily be applied to general wood working.

It is not easy to find a teacher who is willing to share both his knowledge and his workshop. Nowadays violin makers have difficulty making a living from their work. The competition with Far Eastern products is enormous. Furthermore luthiers prefer not to share their workshops containing very sharp tools and machinery…Thanks to contacts from BachoTEX Zbigniew Kegel the luthier of the music school at Solna street in Poznań agreed to be my teacher. – The lessons started in January 2015. The instrument produced its first sounds in March 2016. I spent 4 visits to Poland and got 250 lessons. Not all of them however were spent on the first instrument, because we started to build a second violin, when the first had to be varnished.

When I started to follow the lessons I wanted to take notes, make photos and drawings from each stage of development; which I wanted to combine in an instruction book. From earlier experiences it was clear to me that I would use ConTEXt for typesetting this book. The actual version of the book contains 27 chapters with some 65 photos and 129 drawings, the book is in A5 format and fills 300 pages.

## Setting up ConTEXt

To begin with the book is set up in a project structure. The highest level is the file
project_instrumentenbau.tex
At this moment This file contains only one link to the product: pr_geige.tex
The product contains all links to the chapters which are setup as `components`, e.g., c_formbau.tex.
On the top level of the project structure there is an environment file containing general setups. These can easily be reused in forthcoming products. On the product level there is a second setups file containing product specific setups.
In order to keep it simple, and to get started, I setup ConTEXt with standard available fonts and paper dimensions:

```
\setupbodyfont[pagella,rm,10pt]

\mainlanguage[de]

\setuppapersize[A5][A5]

\setuplayout
  [topspace=15mm,
   backspace=15mm,
   height=middle,
   width=middle,
   marking=on,
   location=middle]
```

I place the page numbers centered in the footer:

```
\setuppagenumbering
  [location=footer,alternative=doublesided]
```

In order to give ConTEXt some more freedom for making up the text-block I set the tolerance to:

```
\setuptolerance[tolerant,stretch]
```

I want to have a single symbol for all levels of itemizations:

```
\setupitemgroup
   [itemize]
   [each][packed][symbol=2,headstyle=bold]
```

The French title and title page will be setup individually, so the
`\startstandardmakeup...\stopstandardmakeup`
should not center the contents vertically.

```
\setupmakeup[standard][bottom=,top=]
```

The book contains three different types of floats i.e. photos, drawings and screenshots. For these floats to be placed in separate lists three new floats are defined, which inherit from the standard figure-float.

```
\definefloat
   [Foto]
   [Fotos]
   [figure]

\definefloat
   [Skizze]
   [Skizzen]
   [figure]

\definefloat
   [Screenshot]
   [Screenshots]
   [figure]
```

Each newly defined float gets its own label:

```
\setuplabeltext[de][Foto=Foto ]
\setuplabeltext[de][Skizze=Skizze ]
\setuplabeltext[de][Screenshot=Screenshot ]
```

While writing the texts it appeared that it could be handy to have macro's for placing the floats. There are single floats, two floats next to each other called a combo and 4 floats together called a block.

Because there are so many photos and drawings, it was handy to introduce two modes that switch off the placement of the photo and drawing floats while keeping their descriptions and references. For ease of use float replacements of photos are printed with blue color, drawing replacements are printed with green color.

```
\define[3]\PutFoto{%
 \doifmodeelse{Fotos}
   {\startplacefloat
    [Foto]
    [title=#2,reference=foto:#1]
    {\externalfigure[#1][width=#3]}
    \stopplacefloat}
   {\blank[small]
    \color[blue]
    {\bf Foto:\ } #2 (ref: foto:#1)
    \blank[small]}}

\define[3]\PutSkizze{%
 \doifmodeelse{Skizzen}
   {\startplacefloat
    [Skizze]
    [title=#2,reference=skizze:#1]
    {\externalfigure[#1][width=#3]}
   \stopplacefloat}
    {\blank[small]
    \color[green]
    {Skizze:\ } #2 (ref: skizze:#1)
    \blank[small]}}

\define[7]\PutFotoCombo{%
 \doifmodeelse{Fotos}
  {\startplacefloat[Foto]
   [title=#1,reference=fotocombo:#2-#5]
    {\startcombination[2*1]
     {\externalfigure[#2][width=#3]}{#4}
     {\externalfigure[#5][width=#6]}{#7}
     \stopcombination}
   \stopplacefloat}
  {\blank[small]\color[blue]{%
   \startlines
    {\bf Foto Combo:\ } #1 \crlf
     \startnarrower[left]
      {\bf Foto 1:\ } #4 (ref: fotocombo:#2-#5)
      {\bf Foto 2:\ } #7 (ref: fotocombo:#2-#5)
      \blank[small]
     \stopnarrower
      \stoplines}}}

\define[7]\PutSkizzenCombo{%
 \doifmodeelse{Skizzen}
  {\startplacefloat[Skizze]
   [title=#1,reference=skizzencombo:#2-#5]
   {\startcombination[2*1]
    {\externalfigure[#2][width=#3]}{#4}
    {\externalfigure[#5][width=#6]}{#7}
   \stopcombination}
  \stopplacefloat}
   {\blank[small]\color[green]{%
    \startlines
     {\bf Skizzen Combo:\ } #1
     \startnarrower[left]
```

```
      {\bf Skizze 1:\ } #4
      (ref: skizzencombo:#2-#5)
      {\bf Skizze 2:\ } #7
      (ref: skizzencombo:#2-#5)
      \blank[small]
     \stopnarrower
     \stoplines}}}

\define[9]\PutFotoBlock{%
 \doifmodeelse{Fotos}
  {\startplacefloat[Foto]
   [title=#1,reference=fotoblock:#2-#4-#6-#8]
   {\startcombination[2*2]
    {\externalfigure[#2][width=.45\textwidth]}{#3}
    {\externalfigure[#4][width=.45\textwidth]}{#5}
    {\externalfigure[#6][width=.45\textwidth]}{#7}
    {\externalfigure[#8][width=.45\textwidth]}{#9}
   \stopcombination}
   \stopplacefloat}
  {\blank[small]\color[blue]{%
   \startlines
   {\bf Foto Block:\ } #1
   \startnarrower[left]
    {\bf Foto 1:\ } #3
    (ref: fotoblock:#2-#4-#6-#8)
    {\bf Foto 2:\ } #5
    (ref: fotoblock:#2-#4-#6-#8)
    {\bf Foto 3:\ } #7
    (ref: fotoblock:#2-#4-#6-#8)
    {\bf Foto 4:\ } #9
    (ref: fotoblock:#2-#4-#6-#8)
    \blank[small]
   \stopnarrower
   \stoplines}}}
```

While writing the texts I realized that in my notes often contained exclamation marks; reminders to pay careful attention to certain actions and details. To make these issues stand out better I defined a macro that places a triangle with exclamation mark in it into the margin. – **Uwaga** is the Polish word for danger.

```
\define\Symbolsize{1.5\bodyfontsize}

\useexternalfigure
 [uwaga]
 [warning-pic]
 [repeat=yes,height=\Symbolsize]

\define\Uwaga{%
 \inmargin{\tbox{\externalfigure[uwaga][]}}}
```

## Page-design for printing on an Indigo-press

When wanting to have the book printed by a professional printer e.g. through pro-book.nl, they told me to provide them a pdf-file containing all the pages not impositioned, with a bleeding margin of 3mm on all 4 sides. This would lead to the following setup in ConTEXt.

```
% A5: width=148 mm, height=210 mm
% Bleed on all 4 sides 3 mm
\definepapersize[Book][width=154mm,
                       height=216mm]
\setuppapersize[Book][Book]
```

Now the lay-out of the page needs to be determined. The white-space around the text area can be set up in many different ways. The classical way of doing this is to divide the width of the page in 9 or 12 equal parts. The inner margin will be 1 part and the outer margin 2 parts. The same applies to the height. More information on defining typesetting-areas is published in MAPS30, 2004. – Although this gives a very pleasant arrangement of the text-area on the paper, the white-space around the text-area is rather big.

A modern approach to defining the whitespace is to make the inner and outer margin equal, and the top whitespace equal to the whitespace at the lower end of the page. In the classical page lay-out running headers and the footer-area do not count in the calculation of the text-height.

I do not have a running header, but I use the footer for placing the page number. The height variable is a combination of the actual text-height, the header- and footer-height and the header- and footer-distance.

```
\setuplayout
 [topspace=18mm,  % actual top whitespace
                  %   + 3 mm bleed
  backspace=18mm, % actual inner whitespace
                  %   + 3 mm bleed
  width=118mm,    % pagewidth - inner and
                  %   outer margin
  height=189mm,   % pageheight - top and bottom
                  %   whitespace + footer
                  %   + footerdistance
  header=0mm,     % no running header
  headerdistance=0mm,
  footer=4mm,
  footerdistance=5mm]
```

## Printing in a copy-shop

Producing the book to be printed by a copy-shop another requires a different approach. Most likely they are able to print on A3 (297 × 420 mm) or SRA3 which is a size of 450 × 320 mm.

The thing is, these papers have the grain direction in the width; which is the shorter measure given above. So accommodating, e.g., A5 with the grain direction in the length of the spine results in two pages on the recto and two pages on the verso side. – I know that this is not economical, but if you place A5 on A4 landscape, one ends up with the paper having the wrong grain direction.

As discussed in the previous section whitespace around the text-area can be defined in different ways, a basic symmetrical design could look as follows:

```
\setuppapersize[A5][A3,landscape]

\setuparranging[2*2*4]
```

Keep in mind, that it is safe to compile the book with arranging enabled from the command line with

```
  context --arrange yourfile.tex
```

In this case ConTEXt will arrange the pages in the last run only and all the lists are correctly filled in.

## General structure of the book

The book gets a classical organization of the different elements.

French title
Title
Impressum

TOC
Table of photos
Table of drawings
Table of screenshots

Introduction

All chapters…which start with a local TOC.

Literature

## References

– Pro-book: http://www.pro-book.nl/home
– Willi Egger, Help! — The Typesetting Area (English) MAPS 30, 2004, 52-59

## Summary

Through 2015 and the first quarter of 2016 I was able to make my first violin under guidance of my teacher Zbigniew Kegel. The collected information, photos and drawings are put together into a book which is typeset with ConTEXt.

W. Egger
w.egger@boede.nl

## Sample text without placement of the floats

### Materialwahl und Aufzeichnen der Form

Die Innenform besteht aus zwei Brettern Birkensperrholz von 12 + 18 mm. Die Abmessungen betragen 400 × 250 mm. Beide Bretter werden zuerst mit 4 Nägeln aufeinander befestigt. Auf dem dickeren Brett wird die Mittellinie angerissen (Ahle) und mit Bleistift eingefärbt.

Das 12 mm dicke Brett wird später die Boden-Seite.

Präzise wird die Halbschablone der Zargenform an diese Mittellinie angelegt und der Umriss auf das Brett übertragen. Als Positionierungshilfe wird auch ein waagrechter Strich von der Schablone auf die Form übertragen.

Skizze: Aufzeichnen der Innenform (ref: skizze:S65)

Nun werden Ausklinkungen für den oberen und unteren Klotz sowie für die Eckklötzchen eingezeichnet.

Vier Schraubenlöcher werden angerissen, womit die beiden Teile der Innenform zusammengeschraubt werden können.

...

### Bearbeiten der Ausklinkungen für die Klötze

| | | |
|---|---|---|
| 2 | Oberer und Unterer Klotz | 50 × 15 mm |
| 4 | Eckklötze | 30 × (3)-4 mm. |

Die Ausklinkungen werden mit Stechbeitel, Raspel, Feile und Schleifpapierfeile so bearbeitet, dass die Grundflächen exakt plan sind. Auch müssen die Flächen rechtwinklig zu den Oberflächen der Form verlaufen.

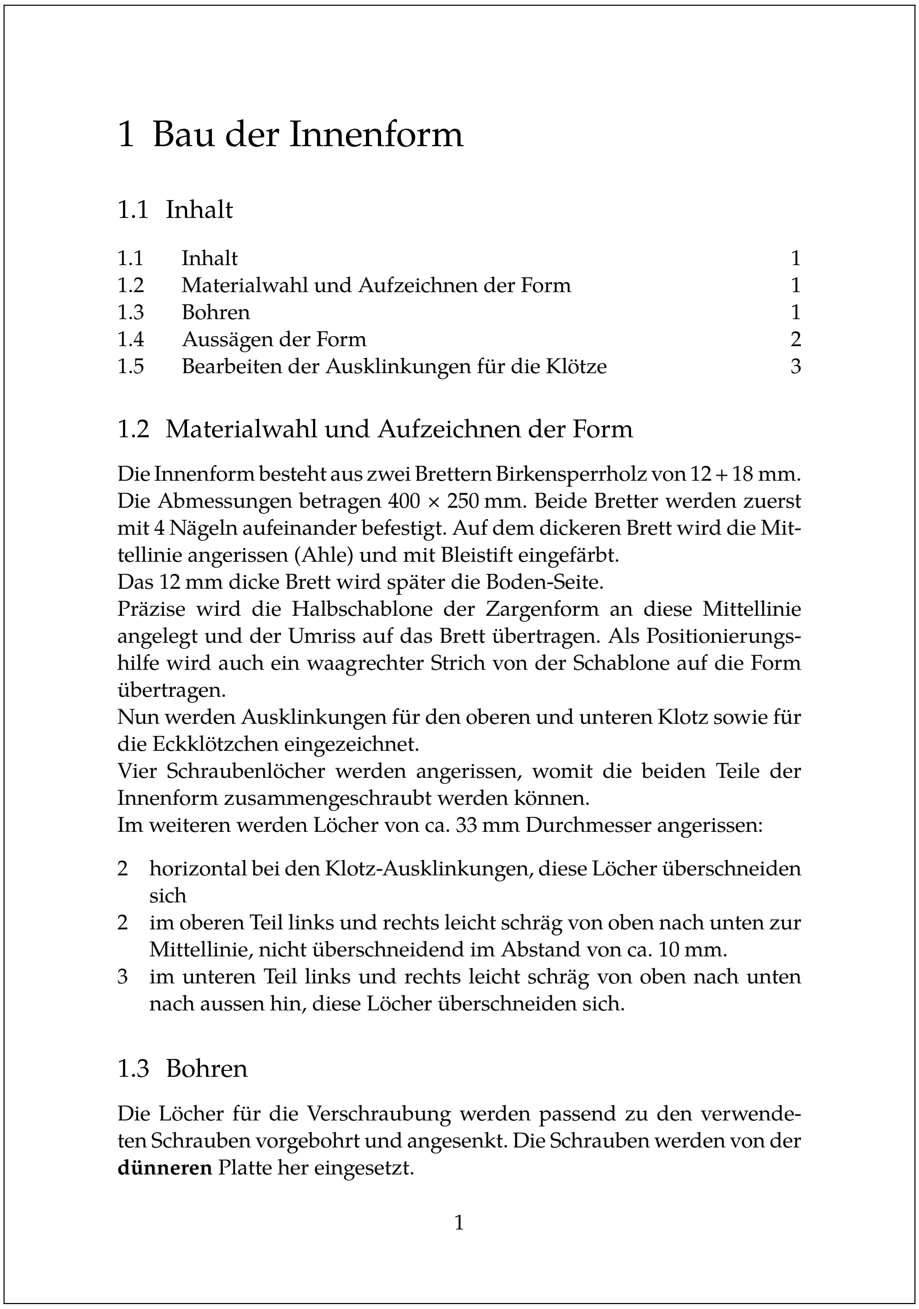
**Foto:** Die Innenform (ref: foto:F64)

**Foto Combo:** Ausklinkungen

**Foto 1:** Halsklotz (ref: fotocombo:F65-F66) **Foto 2:** Eckklotz (ref: fotocombo:F65-F66)

Zum Schluss werden die Leimklemmen-Löcher leicht verputzt.

## Sample pages for building the inner form



# 1 Bau der Innenform

## 1.1 Inhalt

## 1.2 Materialwahl und Aufzeichnen der Form

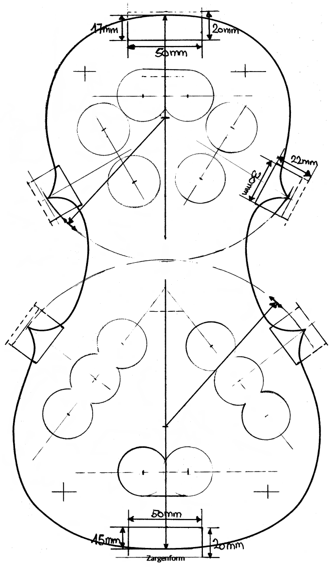
Die Innenform besteht aus zwei Brettern Birkensperrholz von 12 + 18 mm. Die Abmessungen betragen 400 × 250 mm. Beide Bretter werden zuerst mit 4 Nägeln aufeinander befestigt. Auf dem dickeren Brett wird die Mittellinie angerissen (Ahle) und mit Bleistift eingefärbt.
Das 12 mm dicke Brett wird später die Boden-Seite.
Präzise wird die Halbschablone der Zargenform an diese Mittellinie angelegt und der Umriss auf das Brett übertragen. Als Positionierungshilfe wird auch ein waagrechter Strich von der Schablone auf die Form übertragen.
Nun werden Ausklinkungen für den oberen und unteren Klotz sowie für die Eckklötzchen eingezeichnet.
Vier Schraubenlöcher werden angerissen, womit die beiden Teile der Innenform zusammengeschraubt werden können.
Im weiteren werden Löcher von ca. 33 mm Durchmesser angerissen:

2 horizontal bei den Klotz-Ausklinkungen, diese Löcher überschneiden sich
2 im oberen Teil links und rechts leicht schräg von oben nach unten zur Mittellinie, nicht überschneidend im Abstand von ca. 10 mm.
3 im unteren Teil links und rechts leicht schräg von oben nach unten nach aussen hin, diese Löcher überschneiden sich.

## 1.3 Bohren

Die Löcher für die Verschraubung werden passend zu den verwendeten Schrauben vorgebohrt und angesenkt. Die Schrauben werden von der **dünneren** Platte her eingesetzt.

1



**Skizze 1.1** Aufzeichnen der Innenform

Nach der Verschraubung können die Löcher für die Leimzangen gebohrt werden. Bei den horizontalen Lochpaaren wird der Zapfen der sich überschneidenden Kreise weggesägt und mit der Feile geglättet.

## 1.4 Aussägen der Form

Die Form wird auf der Bandsäge heraus gesägt. Hierbei bleibt man mit dem Sägeschnitt 1 mm von der Anrisslinie der Form weg.
Mit Hilfe des Tellerschleifers wird die Kontur soweit nachgearbeitet, dass die Anrisslinie gerade noch sichtbar ist. Wichtig ist, dass die Seitenwände genau im rechten Winkel zur Oberfläche der Form geschliffen werden. Stellen, die mit dem Tellerschleifer nicht erreicht werden können werden am Schleifzylinder bearbeitet. Im Ganzen dient die Form glatt und ohne Wellen zu sein.

## 1.5 Bearbeiten der Ausklinkungen für die Klötze

| | | |
|---|---|---|
| 2 | Oberer und Unterer Klotz | 50 × 15 mm |
| 4 | Eckklötze | 30 × (3)-4 mm. |

Die Ausklinkungen werden mit Stechbeitel, Raspel, Feile und Schleifpapierfeile so bearbeitet, dass die Grundflächen exakt plan sind. Auch müssen die Flächen rechtwinklig zu den Oberflächen der Form verlaufen.

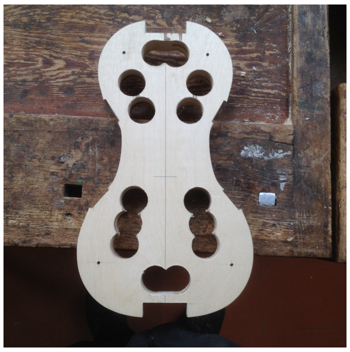
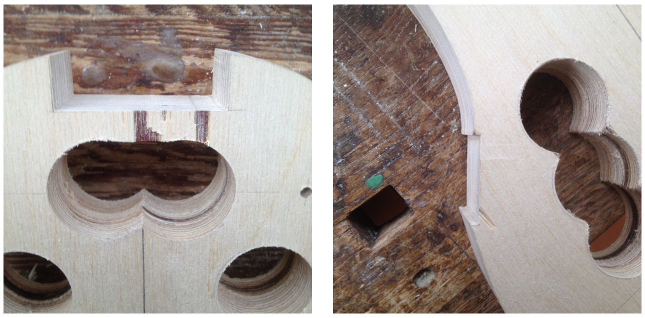


**Foto 1.2** Die Innenform

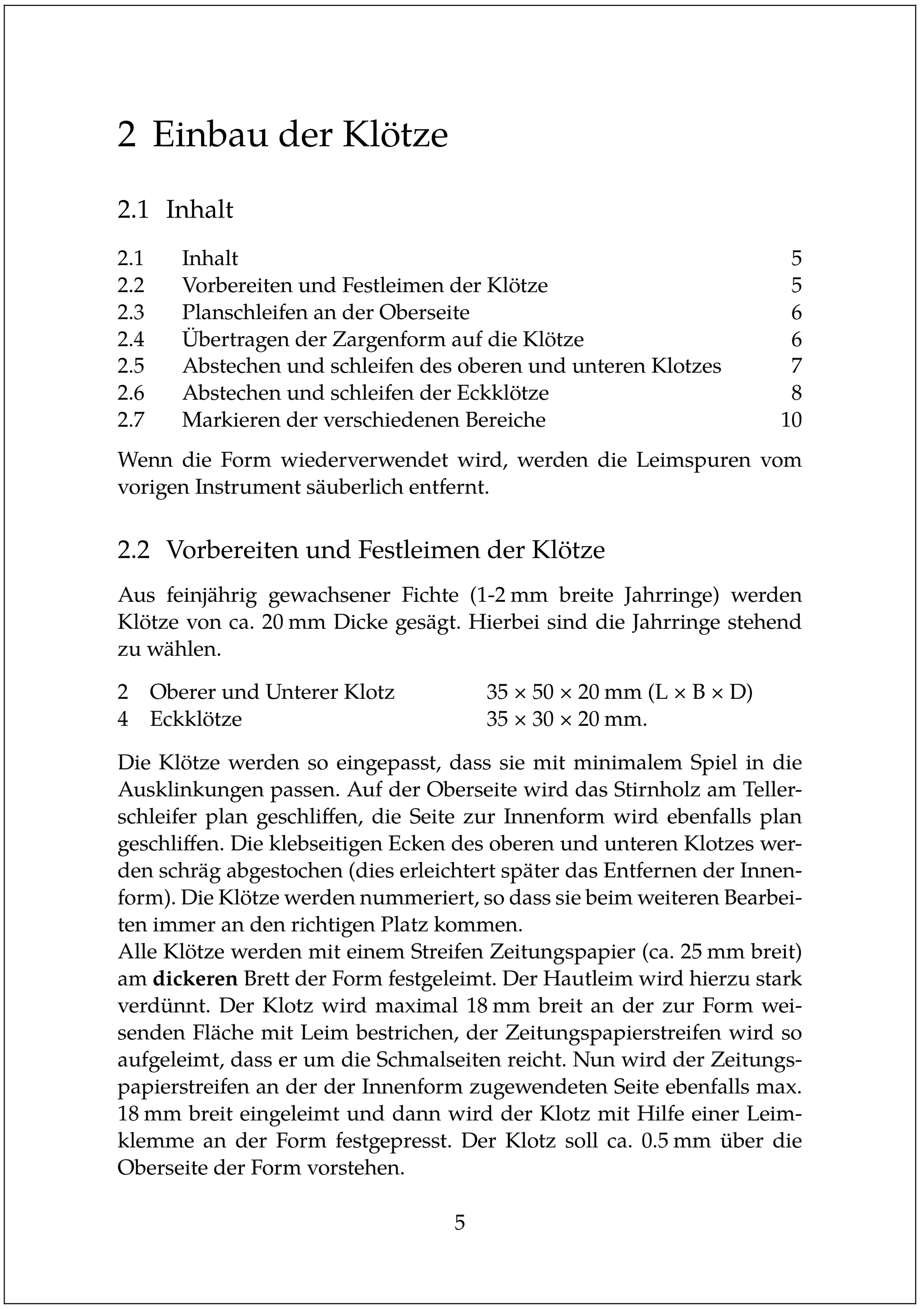


Halsklotz Eckklotz

**Foto 1.3** Ausklinkungen

Zum Schluss werden die Leimklemmen-Löcher leicht verputzt.

3



# 2 Einbau der Klötze

## 2.1 Inhalt

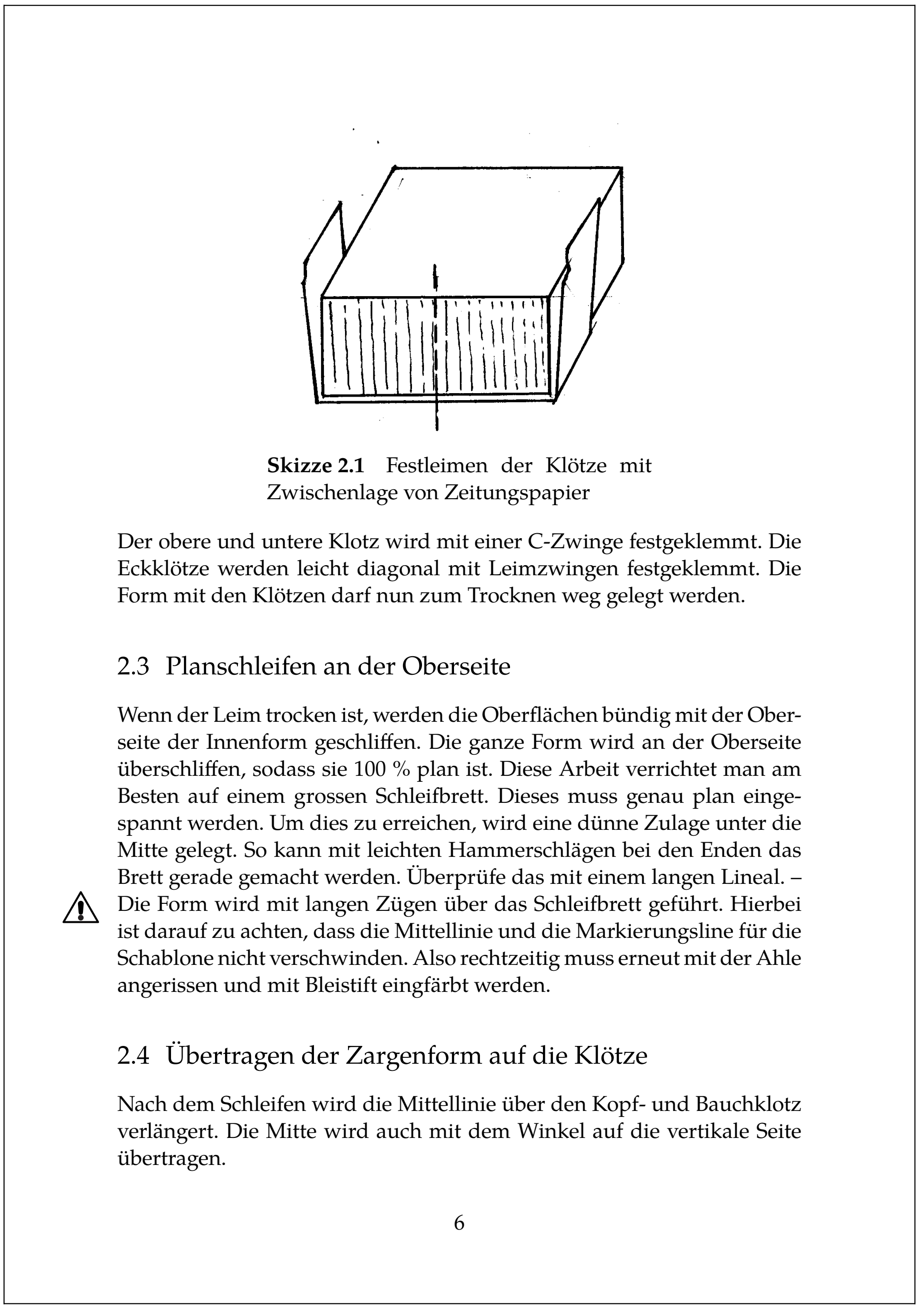
Wenn die Form wiederverwendet wird, werden die Leimspuren vom vorigen Instrument säuberlich entfernt.

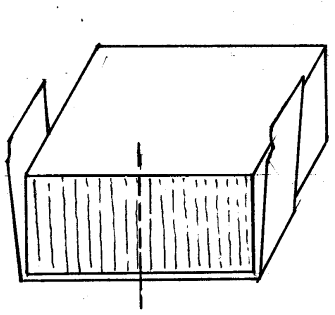
## 2.2 Vorbereiten und Festleimen der Klötze

Aus feinjährig gewachsener Fichte (1-2 mm breite Jahrringe) werden Klötze von ca. 20 mm Dicke gesägt. Hierbei sind die Jahrringe stehend zu wählen.

| | | |
|---|---|---|
| 2 | Oberer und Unterer Klotz | 35 × 50 × 20 mm (L × B × D) |
| 4 | Eckklötze | 35 × 30 × 20 mm. |

Die Klötze werden so eingepasst, dass sie mit minimalem Spiel in die Ausklinkungen passen. Auf der Oberseite wird das Stirnholz am Tellerschleifer plan geschliffen, die Seite zur Innenform wird ebenfalls plan geschliffen. Die klebseitigen Ecken des oberen und unteren Klotzes werden schräg abgestochen (dies erleichtert später das Entfernen der Innenform). Die Klötze werden nummeriert, so dass sie beim weiteren Bearbeiten immer an den richtigen Platz kommen.
Alle Klötze werden mit einem Streifen Zeitungspapier (ca. 25 mm breit) am **dickeren** Brett der Form festgeleimt. Der Hautleim wird hierzu stark verdünnt. Der Klotz wird maximal 18 mm breit an der zur Form weisenden Fläche mit Leim bestrichen, der Zeitungspapierstreifen wird so aufgeleimt, dass er um die Schmalseiten reicht. Nun wird der Zeitungspapierstreifen an der der Innenform zugewendeten Seite ebenfalls max. 18 mm breit eingeleimt und dann wird der Klotz mit Hilfe einer Leimklemme an der Form festgepresst. Der Klotz soll ca. 0.5 mm über die Oberseite der Form vorstehen.





**Skizze 2.1** Festleimen der Klötze mit Zwischenlage von Zeitungspapier

Der obere und untere Klotz wird mit einer C-Zwinge festgeklemmt. Die Eckklötze werden leicht diagonal mit Leimzwingen festgeklemmt. Die Form mit den Klötzen darf nun zum Trocknen weg gelegt werden.

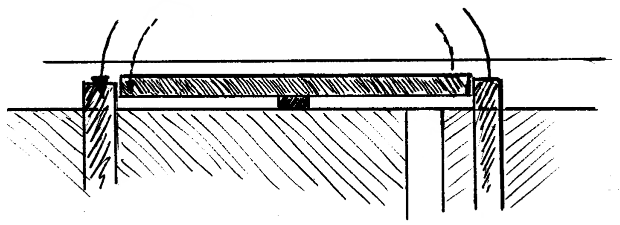
## 2.3 Planschleifen an der Oberseite

Wenn der Leim trocken ist, werden die Oberflächen bündig mit der Oberseite der Innenform geschliffen. Die ganze Form wird an der Oberseite überschliffen, sodass sie 100 % plan ist. Diese Arbeit verrichtet man am Besten auf einem grossen Schleifbrett. Dieses muss genau plan eingespannt werden. Um dies zu erreichen, wird eine dünne Zulage unter die Mitte gelegt. So kann mit leichten Hammerschlägen bei den Enden das Brett gerade gemacht werden. Überprüfe das mit einem langen Lineal. – Die Form wird mit langen Zügen über das Schleifbrett geführt. Hierbei ist darauf zu achten, dass die Mittellinie und die Markierungsline für die Schablone nicht verschwinden. Also rechtzeitig muss erneut mit der Ahle angerissen und mit Bleistift eingfärbt werden.

## 2.4 Übertragen der Zargenform auf die Klötze

Nach dem Schleifen wird die Mittellinie über den Kopf- und Bauchklotz verlängert. Die Mitte wird auch mit dem Winkel auf die vertikale Seite übertragen.

6



**Skizze 2.2** Austarieren des Schleifbrettes

Mit Hilfe der Zargenschablone werden nun die Formen der Klötze übernommen. Zur Überprüfung der Symmetrie de Eckklötze wird mit einem Zirkel ausgehend von der Mittellinie in der Nähe der Hilfslöcher ein Kreis geschlagen und die Spitzen so mit einer ca. 7 mm langen Linie markiert. Siehe hierfür auch in Skizze 1.1 auf Seite 2

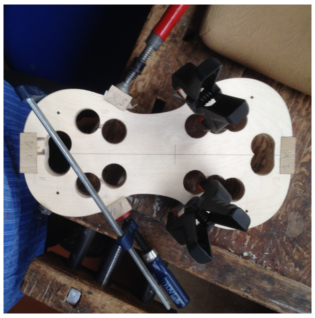


**Foto 2.3** Eingeleimte Klötze

## 2.5 Abstechen und schleifen des oberen und unteren Klotzes

Nun kann der obere und untere Klotz leicht ausserhalb der angerissenen Linie abgestochen werden. Danach werden die Flächen geglättet. Man achte auf einen fliessenden Übergang der Klötze auf die Innenform. Die Aussenfläche der Klötze muss rechtwinklig zur Oberfläche der Form geschliffen werden.

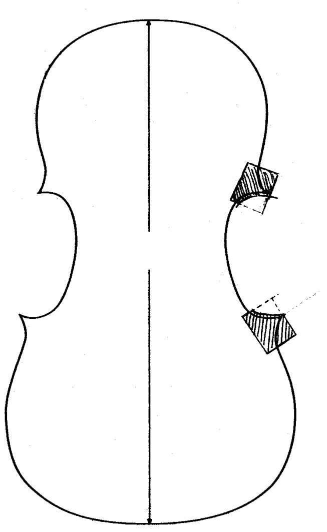
**Foto 2.4** Bauchklotz der Form angepast

Wenn alles stimmt, wird die Mittellinie auf die Stirnseite des oberen und unteren Klotzes übertragen.

## 2.6 Abstechen und schleifen der Eckklötze

Bei den Eckklötzen wird nun die Bogenform grob vorgestochen (Hohleisen 7/14) und die Aussenseite bis zur Zirkellinie mit dem Stechbeitel abgestochen. Die entstandene Fläche wird rechtwinklig zur Form-Oberseite geschliffen. Die Spitzen der Klötze werden auf der Aussenseite des Eckklotzes vertikal angezeichnet.
Nun werden die **Innenflächen** der C-förmigen Mitteleinbuchtung bis auf einen Millimeter von der angerissenen Linie entfernt mit dem Hohleisen 7/14 vertikal abgestochen. Achte hierbei auf den Faserverlauf – Unterschneiden der Vertikalen darf nicht vorkommen! Also erst einen feinen Span schneiden und sehen wie der verläuft. Eventuell muss man von der Gegenseite her schneiden.
Mit einem Schleifzylinder 32 mm wird die Rundung ausgeschliffen, sodass die Spitze und die vertikale Linie erhalten bleiben und an der Innenseite ein sanfter Übergang auf die Innenform entsteht. Man muss die Form umkehren, sodass sie Plan aufliegen kann. Immer wieder muss das Schleifresultat überprüft werden!

8



**Skizze 2.5** Abstechen der Eckklötze

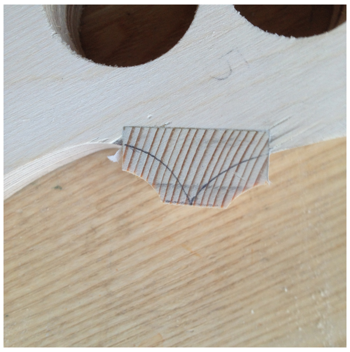


**Foto 2.6** Abstechen der Eckklötze