OrchestralLily: A Package for Professional Music Publishing with LilyPond and LaTeX

Reinhold Kainhofer, http://reinhold.kainhofer.com, reinhold@kainhofer.com
Vienna University of Technology, Austria
and
GNU LilyPond, http://www.lilypond.org
and
Edition Kainhofer, http://www.edition-kainhofer.com, Austria

Abstract

LilyPond [Nienhuys and et al., 2010] and LaTeX provide excellent free tools to produce professional music scores ready for print and sale. Here we present the OrchestralLily package for LilyPond, which simplifies the creation of professional music scores with LilyPond and LaTeX even further. All scores are generated on-the-fly without the need to manually specify the structure for each individual score or part. Additionally, a LaTeX package for the prefatory matter is available and a templates system to create all files needed for a full edition is implemented.

Keywords

LilyPond, Music scores, Publishing, LaTeX, Software package

1 Introduction

In professional music publishing applications like Finale, Sibelius and SCORE are the predominant software packages used. However, the open source applications LilyPond and LaTeX provide excellent free alternatives for producing professionally looking music scores, as well. To ease the production of such scores even further, we developed a package called OrchestralLily for LilyPond and LaTeX. Instead of having to produce each score and instrumental part manually in LilyPond, this package produces these scores dynamically from the music definitions.

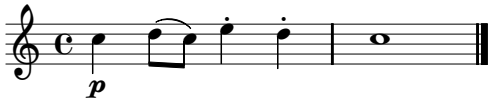
2 A Short Introduction to LilyPond

LilyPond, the music typesetting application developed under the umbrella of the GNU project, is a WYSIWYM ("What you see is what you mean") application, taking text files containing the music definitions and corresponding settings and typesetting it into a PDF file. Writing a LilyPond score is very similar to coding a software program.

In this section we will give a very short overview about the LilyPond syntax, mainly to highlight our motivation to create the OrchestralLily package, which creates the scores dynamically. Fore a more detailed overview over LilyPond we refer to the excellent Documentation of the LilyPond project: http://www.lilypond.org/Documentation/.

A very simple LilyPond score containing only one staff has the following form:

```
\version "2.13.17"
\relative c'' {
  c4\p d8[( c]) e4-. d-. |
  c1 \bar"|."
}
```



All notes are entered by their note name[1], followed optionally by the duration and additional information like beaming ([and]), dynamics (e.g. \p) and articulations (e.g. -. for staccato). When running this file through the LilyPond binary, a five-line staff with a treble clef, 4/4 time signature and C major key is implicitly created. All layouting and spacing is done by LilyPond according to best practices and standards from music engraving.

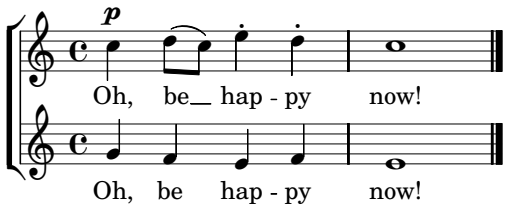
2.1 Writing Full Scores in Pure LilyPond

To produce a score containing a system with more than one staff (e.g. full orchestral scores, choral scores or vocal scores) or to produce a score with lyrics attached to the notes, LilyPond can no longer automatically create the staves, but one has to write the score structure manually into the LilyPond file:

```
\version "2.13.17"

sopmusic = \relative c'' {
  c4\p d8[( c]) e4-. d-. | c1 \bar "|." }
soplyrics = \lyricmode { Oh, be -- hap -- py
    now! }
```

[1] Using Dutch names by default: b for the note below c and the postfix -is for sharp alterations and -es for flat alterations. Other languages can easily be used by including a language file, e.g. \include "english.ily".

```
altomusic = \relative c'' {
  g4 f4 e4  f | e1 \bar "|." }
altolyrics = \lyricmode { Oh, be hap -- py
    now! }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "Soprano" {
        \dynamicUp \sopmusic
      }
    }
    \new Lyrics = "SLyrics"
        \lyricsto "Soprano" \soplyrics
    \new Staff {
      \new Voice = "Alto" {
        \dynamicUp \altomusic
      }
    }
    \new Lyrics = "ALyrics"
        \lyricsto "Alto" \altolyrics
  >>
}
```



Here, the actual music definitions require only eight lines of code, while the structure of the score requires already more lines.

As one can image, creating a full orchestral score with lots of instruments and multiple movements quickly becomes a nightmare to produce manually. Each staff and each staff group defined this way takes 3 to 7 lines of code, quickly leading to a score structure definition of hundreds of lines. For example, a large work with 23 instruments and 12 movements has 276 individual staves, not counting groups. Even worse, each movement typically has the same well-defined structure in the orchestral score. So, a lot of code is duplicated, with the only difference being the music expressions inserted into the scores.

This makes it extremely hard to maintain large orchestral scores in pure LilyPond, and even small changes to the appearance of only one instrument require lots of changes.

2.2 C++ and Scheme / Guile

Internally, LilyPond is written in C++ with Guile as embedded scripting language. Many parts of the formatting code (e.g. all graphical objects like note heads, staff lines, etc.) are defined in Guile and can be modified and overwritten easily using Scheme code embedded into the score. LilyPond even provides an extensive Scheme interface to most of the functions required to create a score. This interface is the key for our OrchestralLily package, where all scores are generated on-the-fly using Scheme.

3 OrchestralLily: An easy example

OrchestralLily uses a slightly different approach than manually writing LilyPond scores: Instead of telling LilyPond explicitly about staves and groups, the score structure is already built in, using default names, and the user only has to define some specially-named variables, containing the music, lyrics, clef, key, special settings, etc. The score is then generated on-the-fly by the following call:[2]

```
\createScore #"MovementName"
    #'("Instrument1" "Group2" "Instrument2")
```

Of course, multiple `\createScore` commands can be given in a file to produce multiple scores in the same file (in particular, this is used in large works with multiple movements, where all movements should be printed sequentially).

The specially-named variables holding the music definition mentioned above have the form

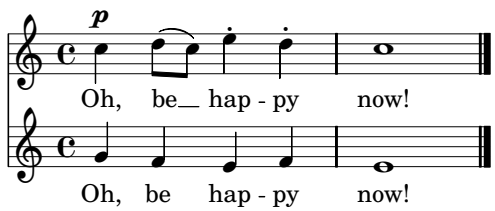
```
[MovementName]InstrumentMusic
```

where `Instrument` is replaced by the (default) abbreviation of an instrument or vocal voice and `[MovementName]` is optional for pieces with only one movement. To define a special clef, key, time signature, lyrics or special settings for a voice, one simply defines a variable containing the clef, key, time signature, etc. This special variable for each instrument is called `[MovementName]InstrumentXXX`, where `XXX` is either `Clef`, `Key`, `TimeSignature`, `Lyrics`, `ExtraSettings`, etc.

To show how this works, the two-voice example from above using OrchestralLily will now look like:

```
\version "2.13.17"
\include "orchestrallily/orchestrallily.ily"

SMusic = \relative c'' {
  c4\p d8[( c]) e4-. d-. | c1 \bar "|." }
SLyrics = \lyricmode {
  Oh, be __ hap -- py now! }
AMusic = \relative c'' {
  g4 f4 e4  f | e1 \bar "|." }
ALyrics = \lyricmode {
  Oh, be hap -- py now! }

\createScore #"" #'("S" "A")
```

[2] The hash sign `#` indicates a scheme expression, the `#'(...)` is a list in Scheme syntax.



It is clear that this automatic creation of staff groups saves a lot of effort for large-scale orchestral projects. It should be noted that OrchestralLily has a large hierarchy of orchestral instruments, including the identifier `"FullScore"` for a full orchestral score. So instead of `#'("S" "A")` above, we could have also said `#'("FullScore")` to generate a full score of all defined voices. Voices not defined will be ignored, so `#'("S" "A" "T" "B")` would have the same output, as the T and B voices are not defined and thus not included in the output.

4 Structure of a Score

To understand OrchestralLily's approach, we have to take a closer look at the organization of a full score. A music score has an intrinsic hierarchy of instruments and instrument groups, as shown in Figure 1.

This hierarchy is pre-defined in OrchestralLily and will be used, unless the LilyPond score explicitly overrides it:

- The instruments are named by their standard abbreviation (e.g. "V", "VI", etc. for violins, "Fag", "FagI" etc. for bassoon, "Ob" for oboe, "S", "A", "T", "B", "SSolo" etc. for vocal voices, etc.).
- Each group of instruments has a predefined name: "Wd" for woodwinds, "Br" for the brass instruments, "Str" for strings (except continuo, i.e. celli and basses, which are typically not included in the strings group, but placed at the bottom of a full score), "Choir", "Continuo", etc.
- Several types of scores are predefined: "LongScore", "FullScore" (like "LongScore", except that two instruments of the same type, e.g. ObI and ObII, are combined and share one staff rather than using separate staves), "VocalScore" (only the vocal voices and the piano voice), "ChoralScore" (only the vocal voices, no instruments).

The score types pre-defined in OrchestralLily adhere to the standard instrument order usually employed for full orchestral scores.

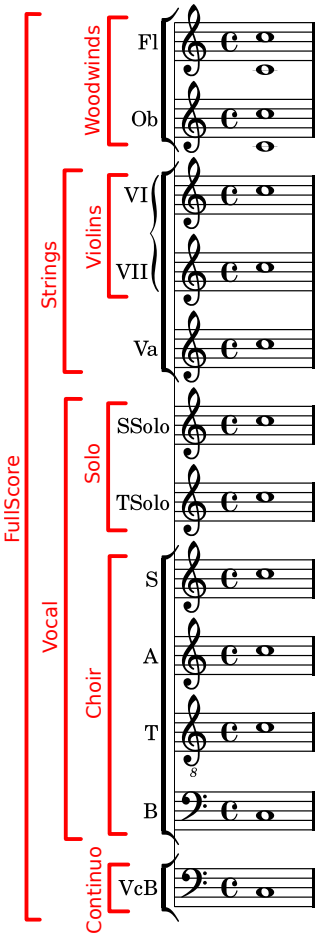


Figure 1: Hierarchy of an orchestral score

5 More complex examples

The examples so far placed the music definition and the actual score creation via `\createScore` into the same file. For larger projects, it is advisable to place the music definitions into a separate file, as we need several different score files, each of which will include this definitions file. All the examples in this section will use the following music definition file "music-definitions.ily", which defines a flute, a violin, soprano and alto, as well as a continuo part for a movement named `Cadenza`. Also, a Piano reduction is defined in this file. Notice also, that this include file already loads OrchestralLily, so we don't have to do this again in the file for each individual score.

```
\include "orchestrallily/orchestrallily.ily"
\include
    "orchestrallily/oly_settings_names.ily"

\header {
  title = "A cadenza"
}
CadenzaPieceNameTacet = "Cadenza tazet"
```

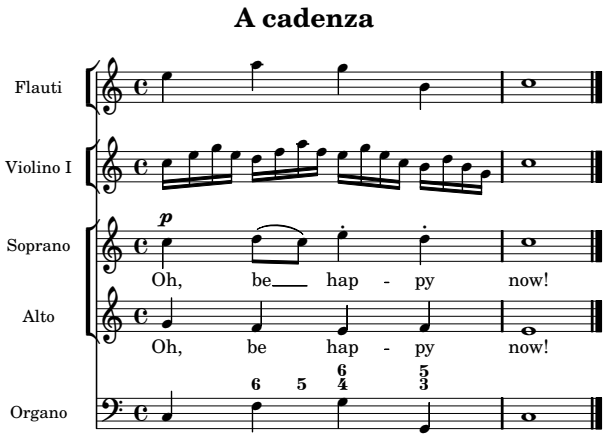
```
% Flute and Violin:
CadenzaFlIMusic = \relative c'' { e4 a g b,
    | c1 \bar "|." }
CadenzaVIMusic = \relative c'' {
 c16[ e g e] d[ f a f] e[ g e c] b[ d b g] |
 c1 \bar "|."
}

% The vocal voices:
CadenzaSMusic = \relative c'' {
  c4\p d8[( c]) e4-. d-. | c1 \bar "|." }
CadenzaSLyrics = \lyricmode {
  Oh, be __ hap -- py now! }
CadenzaAMusic = \relative c'' {
  g4 f4 e4  f | e1 \bar "|." }
CadenzaALyrics = \lyricmode {
  Oh, be hap -- py now! }

% Continuo: Organ / Celli / Bassi / Bassoon
CadenzaBCMusic = \relative c { c4 f4 g g, |
    c1 \bar "|." }
CadenzaFiguredBassMusic = \figuremode {
  s4 <6>8 <5> <6 4>4 <5 3> | s1
}

% Piano reduction:
CadenzaPIMusic = \relative c'' {
  \twoVoice {
    c16[ e g e] d[ f a f] e[ g e c] b[ d b
        g] |
  } {
    e4 a <g c>4 <b f>4
  } | % 2
  <c g e>1 \bar "|."
}
CadenzaPIIMusic = \relative c {
  <c g'>4 f <g c>4 <g d'> | % 2
  <c c,>1 \bar "|."
}
```

All variables in this file start with `Cadenza`, followed by the instrument name, which is how OrchestralLily detects that these definitions belong to a movement name Cadenza. We also defined a piece title to print before the score.

Another thing to notice here is that we also include the file "orchestrallily/oly_settings_names.ily". That file contains many instrument and score name definitions for most common instruments and causes them to be printed before each staff in the score.

5.1 The Full Score

```
\version "2.13.17"
\include
  "orchestrallily/oly_settings_fullscore.ily"
\include "music-definitions.ily"
\setCreateMIDI ##t
\setCreatePDF ##t

\createScore #"Cadenza" #'("FullScore")
```



The additional `\setCreateMIDI ##t` line causes a midi file of the score to be created in addition to the PDF file.

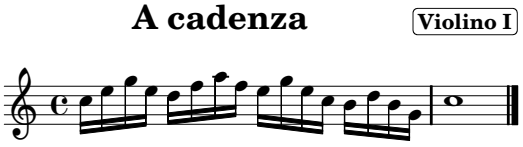
Notice that we never explicitly said that the continuo is supposed to be in bass clef. OrchestralLily already knows that the "BC" (Basso continuo) voice is in bass clef! Similarly, trombone parts will employ the correct C clef, the choir bass will also use the bass clef, etc.

If some instruments should have cue notes, we don't want to print them in the full score, so instead of `\createScore`, OrchestralLily provides the command `\createNoCuesScore`, which will additionally remove all cue notes from the printed score.

5.2 Instrumental Parts

Each instrumental part can be generated just like the full score. If one additionally defines the "instrument" header field, then the instrument name will be printed in the right upper corner, like in most printed scores.

```
\version "2.13.17"
\include "music-definitions.ily"
\include
 "orchestrallily/oly_settings_instrument.ily"
\header { instrument = \VIInstrumentName }

\createScore #"Cadenza" #'("VI")
```



If no music is defined for the given instrument for the desired movement (indicated by the first string that you pass to `createScore`), OrchestralLily will instead print a "tacet" headline. For example, if we try to create a score for the oboe, there is no oboe part defined and a "Cadenza tacet" is printed instead:

```
\version "2.13.17"
\include "music-definitions.ily"
\header { instrument = \ObIInstrumentName }

\createScore #"Cadenza" #'("ObI")
```

A cadenza

Oboe I

Cadenza tazet

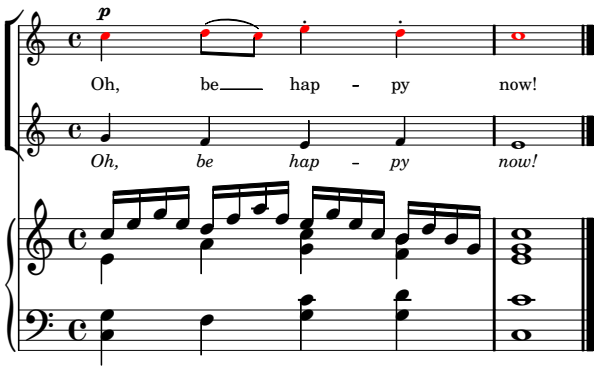
5.3 Vocal Scores and Modifying Individual Staves

To create a vocal score (remember, we have already defined the piano reduction in the definitions!), you only have to call `\createScore` for the "VocalScore" score type. To make things more interesting, here we want the staves for vocal voices to appear smaller than the piano staff. Furthermore, the note heads of the soprano voice should be colored in red and the alto lyrics printed in italic.

These special settings for S and A can be provided by placing them into `\with` blocks and saving them into appropriately named variables called `Cadenza[SA](Staff|Voice|Lyrics)Modifications`:

```
\version "2.13.17"
\include "music-definitions.ily"

CadenzaSStaffModifications = \with {
  fontSize = #-3
  \override StaffSymbol #'staff-space =
      #(magstep -3)
}
CadenzaAStaffModifications =
    \CadenzaSStaffModifications
CadenzaChStaffModifications =
    \CadenzaSStaffModifications

CadenzaALyricsModifications = \with {
    \override LyricText #'font-shape =
    #'italic }

CadenzaSVoiceModifications = \with {
    \override NoteHead #'color = #red }

\createScore #"Cadenza" #'("VocalScore")
```

A cadenza

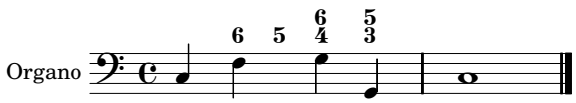


5.4 Figured Bass

The continuo part in the music definitions above is simply the bass line of the cadenza. However, most old scores additionally provide a bass figuration to indicate the harmonies to the organist. Creating such a figured bass score is also extremely simple in OrchestralLily: All you have to do is to define its corresponding variable, named `CadenzaFiguredBassMusic` in our case, where you define the appropriate bass figure in LilyPond's `\figuremode` syntax:

```
\version "2.13.17"
\include "music-definitions.ily"

CadenzaFiguredBassMusic = \figuremode {
  s4 <6>8 <5> <6 4>4 <5 3> | s1
}
\createScore #"Cadenza" #'("Continuo")
```

A cadenza



5.5 Cue Notes

Suppose that we now want to add a second flute, which will set in on the third beat. In the instrumental part, we want to print cue notes from the first flute, but in the full score (or in a combined flute part) we don't want the cue notes.

In LilyPond, one can simply create cue notes by first defining the music to be quoted via `\addQuote` and then inserting the cue notes via `\cueDuring #"quotedInstrument" { r2 }`.

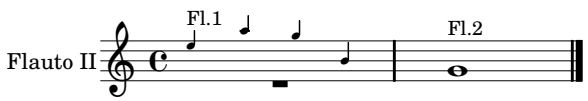
First, we add the new flute 2 part in a separate file "music-definitions-flute2.ily":

```
\addQuote #"Flute1" \CadenzaFlIMusic

CadenzaFlIIMusic = \relative c'' {
  \namedCueDuring #"Flute1" #UP "Fl.1"
      "Fl.2" { R1 } |
  g1 \bar "|."
}
```

Note that we quote the first flute directly in the music for the second flute, using the method `\namedCueDuring` (which is equivalent to LilyPond's built-in function `\cueDuring`, except that it also adds the name of the quoted instrument).

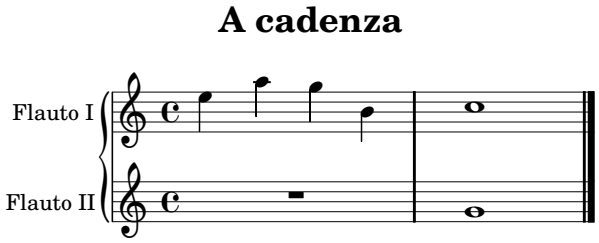
The Flute 2 part now simply is:

```
\version "2.13.17"
\include "music-definitions.ily"
\include "music-definitions-flute2.ily"

% The Flute 2 part:
\createScore #"Cadenza" #'("FlII")
```



In the full score (or a combined flutes part), however, we do not want to print the cue notes, since the notes from Flute 1 are already printed in the score. In this case, we can use `\createNoCuesScore` instead of `\createScore` to suppress the creation of any cue notes:

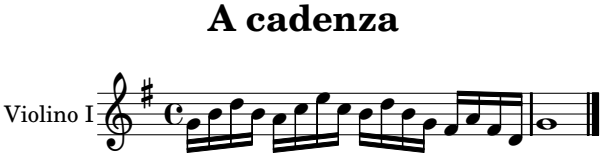
```
\version "2.13.17"
\include "music-definitions.ily"
\include "music-definitions-flute2.ily"

% remove the cues in Flute 2:
\createNoCuesScore #"Cadenza" #'("FlLong")
```



5.6 Transposition

Parts can be easily transposed (e.g. for transposing instruments between concert pitch and written pitch):

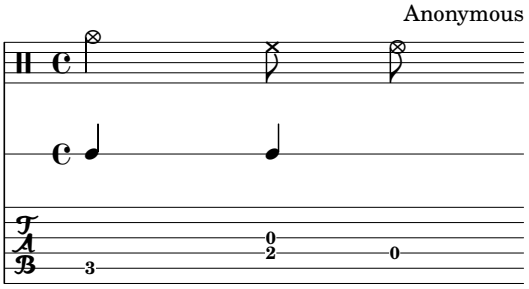
```
\version "2.13.17"
\include "music-definitions.ily"

% We need to give the key explicitly,
% so that it will also be transposed:
CadenzaVIKey = \key c \major
% Transpose to g major
CadenzaVITransposeFrom = g

\createScore #"Cadenza" #'("VI")
```



5.7 Drum Staves and other staff types

Of course, OrchestralLily is also able to print non-standard staves, like rhythmic staves or tablatures:

```
\version "2.13.17"
\include "orchestrallily/orchestrallily.ily"

\header {
  title = "Drum and tab staves"
  composer = "Anonymous"
}

drumIMusic = \drummode { crashcymbal4 hihat8
     halfopenhihat }
drumIIMusic = { c4 c4 }
tabularMusic = { c4 <e g>8 d16 r16 }

\orchestralScoreStructure #'(
  ("drumI" "DrumStaff" ())
  ("drumII" "RhythmicStaff" ())
  ("tabular" "TabStaff" ()))
\orchestralVoiceTypes #'(
  ("drumI" "DrumVoice")
  ("tabular" "TabVoice"))

\createScore #"Cadenza" #'("drumI" "drumII"
    "tabular")
```



6 Tweaking the Score

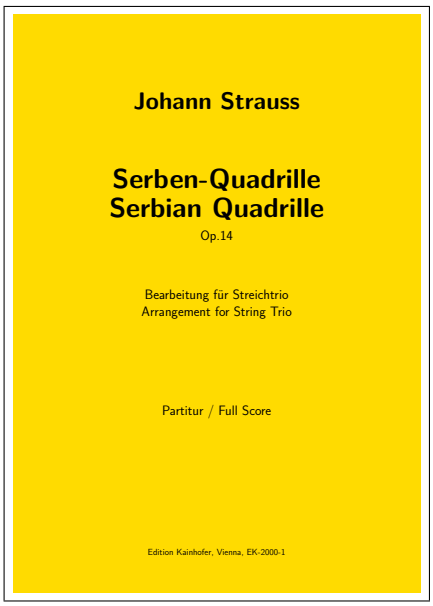
As the OrchestralLily package is implemented entirely in LilyPond syntax and Scheme code, anyone can easily adjust or extend its functionality directly in a score or in an include file, without the need to recompile or reinstall anything.

7 LaTeX for the Preface and Cover

So far, we have concentrated on creating the musical score. A professional edition, however, also features a nice title page, a preface and in many cases also a critical report. For these,

we chose LaTeX for typesetting together with a LaTeX package providing a uniform layout, many macro definitions aiding with the critical report and a beautiful title page. The music scores are directly included into the LaTeX file using the `pdfpages` package.

The templates (see next section) provided by OrchestralLily already produce a beautiful layout without the need for any special tweaks. All one has to do is fill in the missing text and the general information about the score, and the LaTeX scores will look like the following pages from a real score, typeset using OrchestralLily:

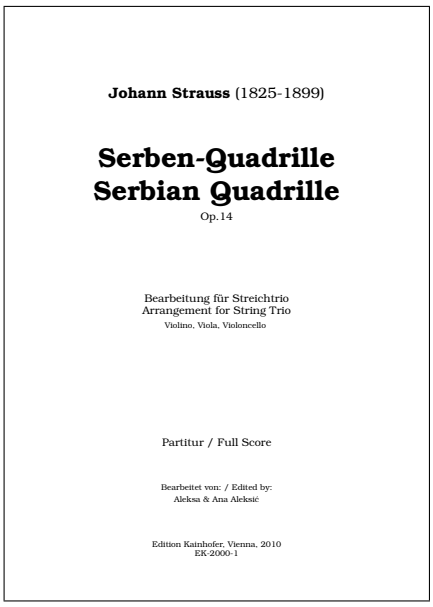

Johann Strauss

Serben-Quadrille
Serbian Quadrille
Op.14

Bearbeitung für Streichtrio
Arrangement for String Trio

Partitur / Full Score

Edition Kainhofer, Vienna, EK-2000-1


Johann Strauss (1825-1899)

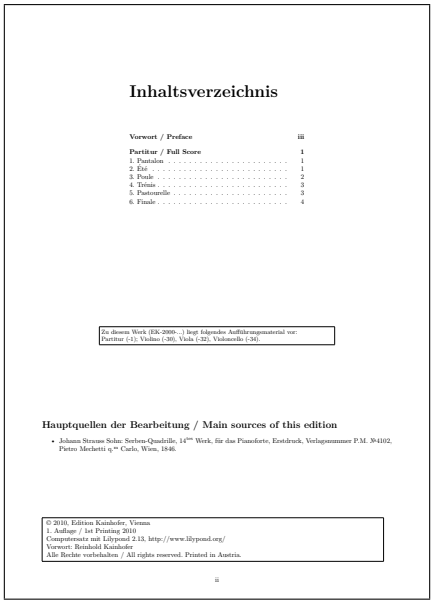
Serben-Quadrille
Serbian Quadrille
Op.14

Bearbeitung für Streichtrio
Arrangement for String Trio
Violino, Viola, Violoncello

Partitur / Full Score

Bearbeitet von: / Edited by:
Aleksa & Ana Aleksić

Edition Kainhofer, Vienna, 2010
EK-2000-1


Inhaltsverzeichnis

Zu diesem Werk (EK-2000-...) liegt folgendes Aufführungsmaterial vor:
Partitur (+1); Violino (+20), Viola (+32), Violoncello (+30).

Hauptquellen der Bearbeitung / Main sources of this edition

- Johann Strauss Sohn: Serben-Quadrille, 14ten Werk, für das Pianoforte, Erstdruck, Verlagsnummer P.M. 30-4102, Pietro Mechetti q.m Carlo, Wien, 1846.

1. Auflage / 1st Printing 2010
Compositursatz mit Lilypond 2.13, http://www.lilypond.org/
Vorwort: Reinhold Kainhofer


Vorwort

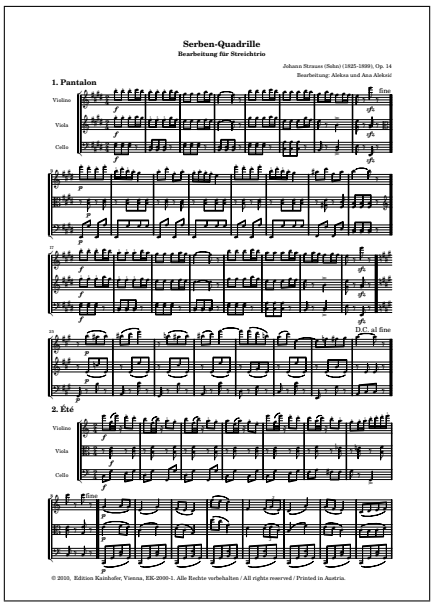
Die Serben-Quadrille ist als Opus 14 ein frühes Werk des Wiener Walzerkönigs Johann Strauss Sohn, komponiert im Auftrag von Fürst Obrenović für einen Slawenball im Jahr 1846 in Wien. In diesem Stück verarbeitet Strauss traditionelle serbische Motive in seinem klassischem Wiener Stil.
Die Rivalität von Johann Strauss Vater und Sohn ab etwa 1845 – der Vater hatte für den Sohn eine andere Laufbahn als die eines Musikers vorgesehen, jedoch wurde der Sohn von seiner Mutter entsprechend unterstützt – und die Bekanntheit des Vaters in der Wiener Musikwelt hatten Johann Strauss Sohn gezwungen, sich eher den Randgruppen der gehobenen Wiener Bevölkerung zuzuwenden. Zu dieser Zeit weilte etwa der ehemalige Serbenfürst Miloš Obrenović (1780-1860) im Exil in Wien, nachdem im Jahr 1842 sein Sohn Mihailo Obrenović nach einem Aufstand zu Gunsten von Aleksander Karađorđević als serbischer Fürst abgesetzt worden und nach Wien geflohen war. Obrenović versuchte auf verschiedene Arten wieder entsprechende Rückendeckung die Herrschaft in Serbien zurückzuerlangen (was ihm 1858 schließlich auch gelang), unter anderem durch die Veranstaltung zahlreicher Slawenbälle in Wien. Für den Ball am 28. Januar 1846 im Alt-Wiener Tanzsaal „Zur goldenen Birn" (den Grazmailen im Bezirk Wien-Landstraße) beauftragte er den jungen Johann Strauss mit einer Quadrille, die nicht nur in hoher Stückzahl in Wien vertrieb, sondern von Obrenović auch nach Serbien geschickt wurde. Die Aufführung beim Slawenball wurde ein großer Erfolg, ebenso wie die erste öffentliche Aufführung am 2. Februar. Die Zeitung „Gegenwart" schrieb so über den Slawenball: „Strauß's Quadrille wurde mit ungeheurem Beifall (kein Hyperbel!) aufgenommen."
Obwohl die Quadrille beim Ball anscheinend von Johann Strauss' Orchester aufgeführt wurde, erschien sie nur als Klavierausgabe im Druck bei Pietro Mechetti. Eine Orchesterfassung ist – wenn sie je existierte – nicht erhalten.

Die vorliegende Bearbeitung der Serben-Quadrille für Streichtrio entstand ebenfalls anlässlich eines Slawenballs in Wien: Für den Wiener Serbenball im Jahr 2006 sollte das Aleksić Streichtrio zur musikalischen Umrahmung auch die Serben-Quadrille von Johann Strauß darbieten. Mangels einer entsprechenden Bearbeitung entstand – von den Mitgliedern des Streichtrios erstellt – die vorliegende Ausgabe der Quadrille für Streichtrio in der Besetzung Violine, Bratsche und Cello. Die Melodie wird dabei praktisch ausschließlich von der Violine übernommen, die Bratsche unterstützt entweder die Melodie der Violine oder die Harmonik des Cellos.

Quellen: / Sources
[Maiff] Franz Maier: *Johann Strauß*, Kommentiertes Werkverzeichnis, Verlag Pichler, Wien, 1999.

Preface

The „Serbian Quadrille" Op.14 is an early work of the „Waltz King" Johann Strauss Jr., commissioned by prince Obrenović for a Slavic ball 1846 in Vienna. In it, Strauss uses traditional Serbian motifs in his typical Viennese style.
A quadrille – a precursor of modern square dancing – is a dance formation performed by four couples in square formation. The related music style, also called quadrille, typically consists of six figures in its Viennese style: 1. „Le Pantalon" (a pair of trousers), 2. „L'été" (the summer), 3. „La Ponle" (the hen), 4. „La Trénis" (named after dance master Trenitz), 5. „La Pastourelle" (the shepherd girl) and 6. Finale. The original French form consisted of only 5 figures, missing the Trénis.
The history of this quadrille carries some interesting aspects as well: The rivalry with his father Johann Strauss Sr., a famous dance music composer as well, who did not want his son to become a musician, and the influence of the father in Vienna had forced the son to turn his attention away from the mainstream of the Viennese high society. At that time, the former Serbian prince Miloš Obrenović (1780-1860) was living in exile in Vienna, after his son Mihailo Obrenović had been deposed in favor of Aleksander Karađorđević. Obrenović tried to regain the support of the Serbian population to return as Serbian prince (where he eventually succeeded in 1858). As one of many ways to gain popularity he organized numerous Slavic balls in Vienna.
For the ball on January 28, 1846, in the dance hall SZur goldenen Birn"(district Wien-Landstraße), he instructed the younger Strauss, with a quadrille, which was distributed not only in large quantities in Vienna, but also sent to Serbia by Obrenović. The performance at the Slavic ball was a great success, as well as the first public performance on February 2.
Although the quadrille appears to have been performed by Strauss' orchestra at the ball, it appeared in print only as a piano edition published by Pietro Mechetti. An orchestral version is not preserved.
The present arrangement of the Serb-Quadrille for string trio was also created for a Slavic ball in Vienna: At the Serbian ball in 2006, the Aleksić String Trio was asked to perform the „Serbian quadrille" by Strauss as part of the musical accompaniment. Lacking a proper arrangement, the members of the string trio decided to write this arrangement for violin, viola and cello themselves. The melody is almost entirely played by the violin, the viola supports either the melody of the violin or the harmony of the cello.


Serben-Quadrille
Bearbeitung für Streichtrio

Johann Strauss (Sohn) (1825-1899), Op. 14
Bearbeitung: Aleksa und Ana Aleksić

1. Pantalon

Violino
Viola
Cello

fine

D.C. al fine

2. Été

Violino
Viola
Cello

fine

8 Generating a Template

OrchestralLily also provides a template-based script to generate all files required for a full edition of a score: The score information, including instrumentation, movements, voices with lyrics, etc., are defined in one input file. After running the `generate_oly_score.py` script, one has the full set of files for the edition, including a Makefile. Only the music, lyrics, and the actual text of the preface and the critical report need to be filled in. Running `make` will always update all scores and produce ready-for-print files for all desired scores and instrumental parts.

A typical input file for the cadenza example used above is shown here:

```
{
 "output_dir": "Cadenza",
 "version": "2.13.11",
 "template": "EK_Full",

 "defaults": {
   "title": "A test for OrchestralLily",
   "composer": "Reinhold Kainhofer",
   "composerdate": "1977-",

   "year": "2009",
   "publisher": "Edition Kainhofer",
   "scorenumber": "EK-1040",
   "basename": "Cadenza",
   "parts": [
     {"id": "Cadenza", "piece": "A cadenza",
      "piecetacet": "Cadenza tazet"},
   ],

   "instruments": ["FlI", "FlII", "VI", "S",
        "A", "Continuo"],
   "vocalvoices": ["S", "A"],
   "scores": ["Full", "Vocal", "Choral"],
 },

 "scores": ["Cadenza"],
 "latex": {},
}
```

Running this file through the script generates one definitions file for the music definition, LilyPond files for each of the given scores (Full, vocal and choral scores), as well as for each individual instrumental part. Each of the scores will also have a LaTeX file that includes the title page, the preface (including the table of contents, which is exported by LilyPond!), the score and optionally a critical report.

All these files are tied together via a Makefile, so all one needs to do to create a first version is to copy in the music definition and run make.

9 Availability of OrchestralLily

The OrchestralLily package [Kainhofer, 2010b] is currently dual-licensed under the Creative Commons BY-NC 3.0 license [Creative Commons, 2010] as well as under the the GPL v3.0.

Its source code can be found in a public git repository [Kainhofer, 2010a]: `http://repo.or.cz/w/orchestrallily.git`.

More information about the OrchestralLily package can be found in the documentation at its homepage `http://kainhofer.com/orchestrallily/` (which is unfortunately not always kept up to date) or better directly from the source code.

10 Acknowledgements

A project like OrchestralLily would of course never be possible without the help of many people. The developers of LilyPond and of LaTeX – too many to name them explicitly here – made OrchestralLily possible in the first place by providing excellent open source applications for both music and text typesetting. The enormous flexibility and configurability of both applications (including the possibility to modify the internals and implement required features yourself) laid the foundation to turn a small project into a professional music publishing framework.

The cadenza example used throughout this article was originally written by me, until Ana Aleksić pointed out several harmonic shortcomings and helped me rewrite it. Similarly, Manfred Schiebel greatly improved my dilettantish attempts at producing a piano reduction.

References

Creative Commons. 2010. By-nc 3.0 at license. `http://creativecommons.org/licenses/by-nc/3.0/at/`.

Reinhold Kainhofer. 2010a. Git repository of OrchestralLily. `http://repo.or.cz/w/orchestrallily.git`.

Reinhold Kainhofer. 2010b. The OrchestralLily package for lilypond. `http://kainhofer.com/orchestrallily`. LilyPond and LaTeX package for professional music typesetting.

Han-Wen Nienhuys and Jan Nieuwenhuizen et al. 2010. GNU LilyPond. `http://www.lilypond.org/`. The music typesetter of the GNU project.