MASARYKOVA UNIVERZITA

FAKULTA INFORMATIKY

Synchronizovaná animace notového zápisu

BAKALÁŘSKÁ PRÁCE

Jiří Szabó

Brno, jaro 2012

Prohlášení

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

.....

Vedoucí práce: Mgr. et Mgr. Vít Baisa

Poděkování

Touto cestou bych rád poděkoval Mgr. et Mgr. Vítu Baisovi za vedení této práce a za ochotu a rady při jejím vytváření.

Shrnutí

Bakalářská práce „Synchronizovaná animace notového zápisu“ se věnuje tvorbě programu pro počítačový nástroj GNU LilyPond, jenž umí z předloženého zdrojového kódu v jazyce LilyPond vygenerovat video notového zápisu dané skladby. Posun zápisu je pak synchronizován na právě přehrávané tóny.

Klíčová slova

LilyPond, MIDI, notový zápis, synchronizace, audio, video.

Obsah

1. Úvod

1.1. Uvedení do problematiky

Notový zápis patří mezi nejspolehlivější metody, jak zaznamenávat hudbu. První záznamy použití jsou datovány do období dva tisíce let před naším letopočtem, základy jeho moderní podoby byly položeny během druhé poloviny šestnáctého a první poloviny sedmnáctého století.

Odborník (nebo alespoň hudebně vzdělaný člověk) při pohledu na podobný zápis okamžitě odhalí základní vlastnosti dané skladby, laik je ztracen. Dokáže sice identifikovat notovou osnovu, rozmístěným značkám ovšem nerozumí a pohled na ně mu neřekne nic hlubšího.

Technologický pokrok s sebou přinesl nové možnosti. Skládání hudby, její převedení do slyšitelné podoby a následné vystavení na Internetu už v dnešní době nepředstavuje žádný problém. Lehký zádrhel může nastat ve chvíli, kdy chce autor svoji skladbu zveřejnit v podobě videa, například na serveru YouTube[1]. Co použít jako obraz?

Jedním z prvních nápadů může být právě notový zápis. Opět se však vrací problém nastíněný ve druhém odstavci, neboť sledování statických obrázků not, které se jednou za čas vymění, ocení jenom hudby znalý člověk a laika tedy příliš nezajímají.

Synchronizovaný pohyb partitury by však mohl mít daleko větší šanci na úspěch. Její pohyb je založen na hrající skladbě a i laik si na první pohled může všimnout určité provázanosti mezi notami na obrazovce a znějícími tóny. Náhle dokáže daleko lépe identifikovat určité značky s určitým zvukem (či naopak tichem), snadněji si melodii vizuálně představí a také přesně ví, v jaké části partitury se skladba momentálně nachází.

[1] Webová stránka http://www.youtube.com.

1.2. Cíl práce

Cílem bakalářské práce „Synchronizovaná animace notového zápisu“ bylo vytvoření programu, který ze skladby zapsané v jazyce LilyPond vygeneruje video, jehož obraz zachycuje notovou partituru dané skladby, která se pohybuje na základě zvukového doprovodu.

1.3. Osobní motivace k výběru tématu

Toto téma jsem si pro svoji bakalářskou práci vybral především díky svému osobnímu vztahu k hudbě. Již od útlého věku až po dovršení patnáctých narozenin jsem aktivně hrál na elektronický keyboard a zároveň studoval základní uměleckou školu v Brně-Židenicích.

1.4. Obsah jednotlivých kapitol

Druhá kapitola je věnována detailnějšímu popisu samotného GNU LilyPond. Práce čtenáře seznamuje s tímto nástrojem, rekapituluje ve stručnosti jeho historii a především uvádí základy používané syntaxe společně s krátkými ukázkami. Součástí je také krátká odbočka k textovým editorům, ve kterých lze psát zdrojový kód a k projektu Mutopia, virtuálnímu „archívu“ projektů vytvořených v GNU LilyPond.

Další část na tu předchozí částečně navazuje, neboť hovoří o standardu MIDI (Musical Instrument Digital Interface), jednoho z možných výstupů nástroje LilyPond. Účelem této kapitoly však není do detailu rozebrat vývoj MIDI či objasnit systém jakým funguje, ale spíše popsat jeho návaznost a propojení se samotným GNU LilyPond.

Následující pasáž textu se již plně věnuje hlavnímu výstupu této bakalářské práce – programu `ly2video`. Každá z podkapitol postupně rozebírá všechny důležité součásti samotné aplikace. Od softwaru potřebného ke spuštění, možností synchronizace notového zápisu až po vznik výsledného videa. Poslední část poté identifikuje nejzásadnější nedostatky.

Závěr práce se snaží nastínit možnosti budoucího vývoje a také diskutuje přínos aplikace do problematiky automatizované tvorby videí notových materiálů.

1.5. Přílohy

K textu je přiložena příloha, obsahující seznam veškerých přepínačů použitelných při spouštění programu `ly2video` společně s krátkým vysvětlujícím komentářem.

2. GNU LilyPond

2.1. Úvod

Tak jako existuje spousta jazyků a nástrojů pro tvorbu počítačových programů, existují i jazyky a nástroje pro zápis hudby. Hudba se podobně jako kódy aplikací musí řídit určitým souborem pravidel, na kterých autor může vystavět už takřka cokoliv. Dokonce se dá hovořit o určitém „hudebním programování“, v němž programátor místo psaní podmínek, cyklů a vstupně-výstupních operací píše noty, pomlky a opakování.

Jedním ze zástupců „hudebního programování“ je i GNU LilyPond[2]. Tento nástroj napsaný v jazycích C++ a Scheme [1] pro zápis hudební partitury používá svůj vlastní jazyk, jehož základy jsou popsány v podkapitole 2.3.

Mezi výhody GNU LilyPond bych zařadil především tyto čtyři vlastnosti: jednoduchá syntaxe, generovaný výstup notového zápisu má vysokou grafickou kvalitu, velká svoboda při vytváření zdrojového kódu a široké možnosti v upravování i těch nejmenších detailů (například barva not, číslování taktů, umístění textu, délka nožiček a další).

Poslední dvě zmiňované přednosti jsou však dvousečnou zbraní. Vzhledem k benevolenci GNU LilyPond ke zdrojovému kódu a jeho možným podobám je relativně těžké postihnout všechny potenciální alternativy, což vývoj aplikací pro tento nástroj značně ztěžuje.

Závěrem úvodní části druhé kapitoly je třeba upozornit, že samotné slovo „LilyPond“ má v rámci celého projektu několik odlišných významů. Pokud tedy jeho význam nebude v rámci kontextu daného textu jednoznačný, bude explicitně uvedeno, zda se jedná o programovací jazyk, celý nástroj, či zdrojový kód.

[2] Mezi další patří například jazyk ABC notation (http://www.abcnotation.com), MusiXTeX (sada maker a písem rozšiřující TeX) či nástroje MuseScore (http://www.musescore.org) nebo Rosegarden (http://www.rosegardenmusic.com).

2.2. Stručná historie

Vývoj GNU LilyPond začal v roce 1996 a jeho autory jsou dva tehdejší členové eindhovenského orchestru mládeže Han-Wen Nienhuys a Jan Nieuwenhuizen. Nieuwenhuizen se tehdy svému francouzskému kamarádovi pochlubil novým projektem, na němž tehdy pracoval, preprocesorem pro nástroj MusiXTeX. Han-Wen však nebyl s kvalitami MusiXTeXu příliš spokojen a po ročním váhání se rozhodl vytvořit si svůj vlastní nástroj pro sázení hudby, LilyPond. Zanedlouho se k němu připojil i Jan [2].

První plnohodnotná verze byla zveřejněna v roce 1998. Druhá ji následovala o pět let později a přinesla především zjednodušení používané syntaxe. Poslední stabilní verze má označení 2.14.2 a vyšla v červenci loňského roku [3].

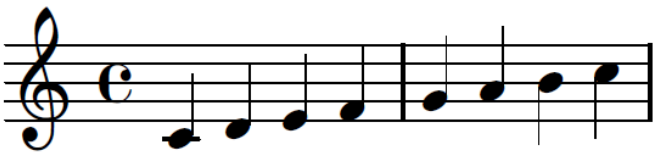
2.3. Nastínění syntaxe

Následující podkapitola se snaží alespoň částečně představit syntaxi programovacího jazyka LilyPond a naznačit, jakým způsobem v něm lze zapisovat noty. LilyPond je v tomto směru velice benevolentní a poskytuje skutečně velikou svobodu při psaní zdrojového kódu. Níže uvedené příklady tak nejsou jediným možným způsobem, jak daný příklad zapsat.

2.3.1. Základy notového zápisu

Zapisování jednotlivých not na notovou osnovu funguje v jazyce LilyPond velice intuitivně. Stačí používat jejich názvy z hudební terminologie – c, d, e, v případě půltónů například gis, ais nebo bes. Základní stupnice C dur vypadá takto:

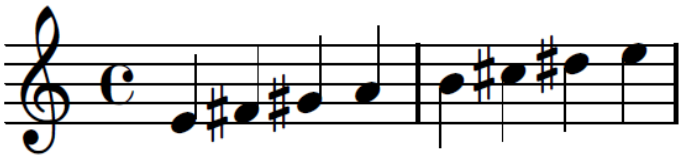
```
{
   c‘ d‘ e‘ f‘
   g‘ a‘ b‘ c‘‘
}
```



Obr. 1: Stupnice C dur

Složené závorky reprezentují oblast notového zápisu, apostrofy u not zase jejich oktávu. Ustavičné psaní „čísla“ oktávy ke každé notě by bylo u rozsáhlejších zápisů poměrně únavné, proto lze před složené závorky napsat příkaz `\relative`, pomocí něhož se explicitně řekne, v jaké oktávě má daný zápis začínat. Výška následující noty se pak odvozuje dle vzdálenosti kvarty té předcházející (Obr. 2). Jednorázové úpravy tohoto pravidla se provádějí pomocí apostrofů či čárek za názvem noty, například `c‘` by se vyložilo jako přeskočení jedné oktávy vzhledem k předcházející notě.

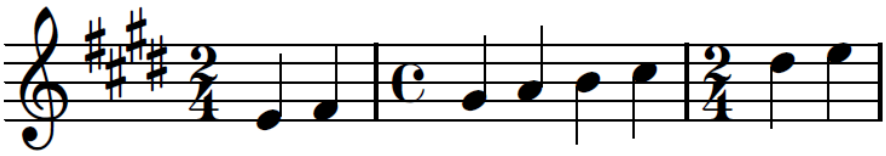
```
\relative c‘ {
   e fis gis a
   b cis dis e
}
```



Obr. 2: Stupnice E dur

Základní čtyřčtvrteční takt[3] se změní pomocí příkazu `\time`, za nímž následuje jeho nová hodnota. Podobně jako u not se i zde používají pojmy známé z notových zápisů, tedy například 3/4 nebo 6/8. Mezi (v tomto případě houslový) klíč a označení taktu lze vložit předznamenání, čehož se dosáhne pomocí direktivy `\key`. Za ní musí následovat vybraná tónina a zda jde o její durovou (`\major`), či mollovou (`\minor`) podobu. Upravená stupnice E dur s využitím všech těchto příkazů může vypadat například takto:

```
\relative c‘ {
   \key e \major
   \time 2/4 e fis
   \time 4/4 gis a b cis
   \time 2/4 dis e
}
```



Obr. 3: Stupnice E dur s využitím příkazů `\time` a `\key`

[3] Značka „c“ je jiným označením čtyřčtvrtečního taktu (4/4).

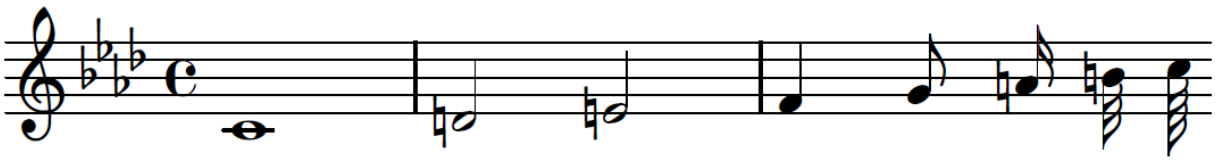
Jak naznačuje Obr. 3, i přes přítomnost předznamenání je nutno stále používat „správné“ názvy jednotlivých not (tedy například fis). Pokud by uživatel vložil pouze f, bude před danou notou odrážka.

Změna klíče na dané notové osnově se provádí příkazem `\clef`, k dispozici jsou klíč houslový (standardní volba i bez použití `\clef`; značka `treble`), basový (`bass`), altový (`alto`) a tenorový (`tenor`), ale i spousta dalších.

2.3.2. Délka not

Délka noty se zapisuje přímo za její název (respektive za název a případný posun oktávy) a to pomocí číslice. Opět zde funguje „klasická“ notace, tudíž například ais2 bude půlová nota, ais8 osminová a tak dále. Při absenci číselného ohodnocení je použita nejbližší předcházející délka, její určování tedy funguje rekurzivním systémem. Standardní nastavení notového zápisu používá čtyřčtvrteční notu, což je vidět i na předchozích obrázcích.

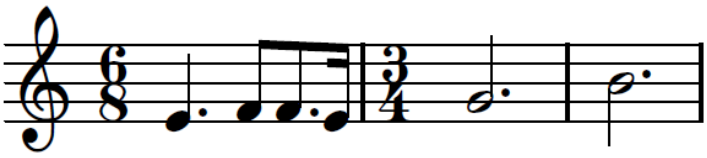
```
\relative c‘ {
   \key f \minor
   c1 d2
   e2 f4
   g8 a16
   b32 c64
}
```



Obr. 4: Příklad standardních délek

Nechybí ani známý operátor tečky, který tón prodlužuje o padesát procent původní délky. Tečka se ve zdrojovém kódu píše na konec noty a pro její použití je nutno explicitně uvést i číselnou délku (rekurze zmiňovaná v předchozím odstavci zde nefunguje).

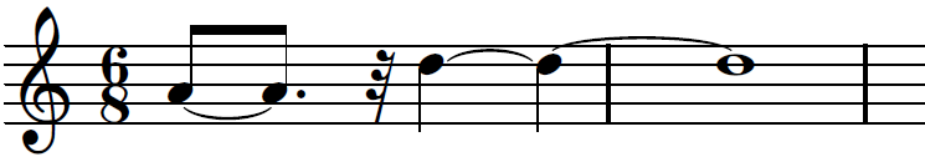
```
\relative c‘ {
   \time 6/8
   e4. f8 f8. e16
   \time 3/4
   g2. b2.
}
```



Obr. 5: Příklad použití tečky

Třetí možností jak ovlivnit délku tónů je ligatura. Ligatura je uskupení několika po sobě jdoucích not a v LilyPondu je značena znakem vlnovky, ~ (Obr. 6). Ve většině případů jsou takto propojeny noty o stejné výšce, z nichž zazní vždy pouze ta první. Ostatní ji pak už pouze prodlužují. Délka ligatury je tak přímo úměrná součtu délek not v jednom propojení.

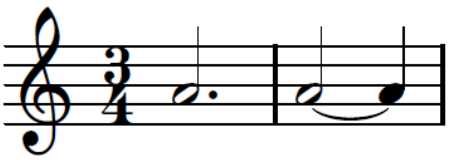
```
\relative c‘ {
   \time 6/8
   a‘8 ~ a8. r32
   d4 ~ d4 ~ d1
}
```



Obr. 6: Příklady možných ligatur

Závěrem je možno dodat, že některé varianty ligatury je možno nahradit tečkovou notací (a obráceně), jak dokazuje obrázek č. 7.

```
\relative c‘ {
   \time 3/4
   a‘2.
   a2 ~ a4
}
```

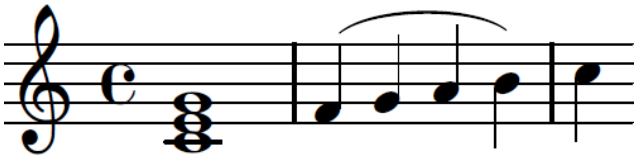


Obr. 7: Tečkový zápis ekvivalentní ligatury

2.3.3. Harmonie a legato

Harmonie (zahrání několika not zároveň) a legato (posloupnost několika not spojených obloučkem se musí hrát s co nejmenším odstupem mezi sebou) se ve zdrojovém kódu zapisují pomocí lomených, respektive hranatých závorek. Každý z těchto typů zápisu má lehce odlišnou syntaxi. Zatímco noty uzavřené v lomených závorkách se chovají jako jedna jediná (a lze tedy jednorázově upravovat i délku harmonie, viz Obr. 8), v legatu jsou na sobě i nadále zcela nezávislé. Jedna nota navíc musí stát ještě před závorkami, aby legato uvozovala.

```
\relative c‘ {
   <c e g>1
   f4(g a b) c
}
```

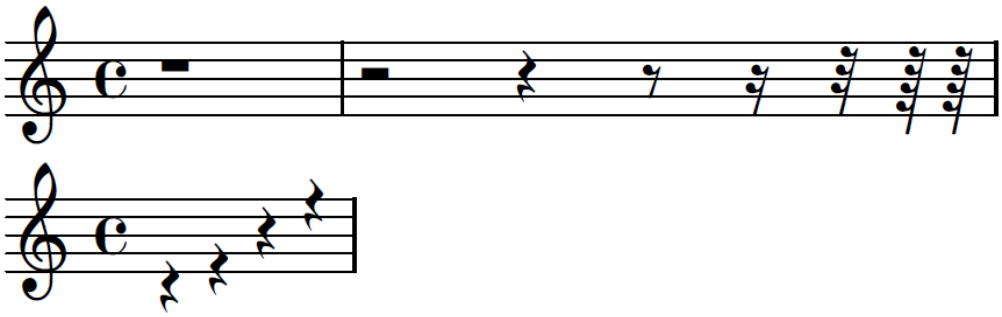


Obr. 8: Lomené a kulaté závorky v praxi

2.3.4. Pomlky

Důležitou součástí hudebního zápisu jsou kromě not také pomlky. LilyPond umožňuje dvojí způsob zapisování pauz: buď jako speciální notu nesoucí název `r` (platí pro ni stejná pravidla jako pro ty obyčejné) nebo pomocí příkazu `\rest`. V druhém případě je nutno před samotný příkaz ještě přiřadit notu, jejíž délka a pozice na notové osnově bude pomlce přiřazena (Obr. 9).

```
\relative c‘ {
   r1 r2 r4 r8 r16 r32 r64
}
\relative c‘ {
   c4 \rest e4 \rest
   b‘4 \rest e4 \rest
}
```



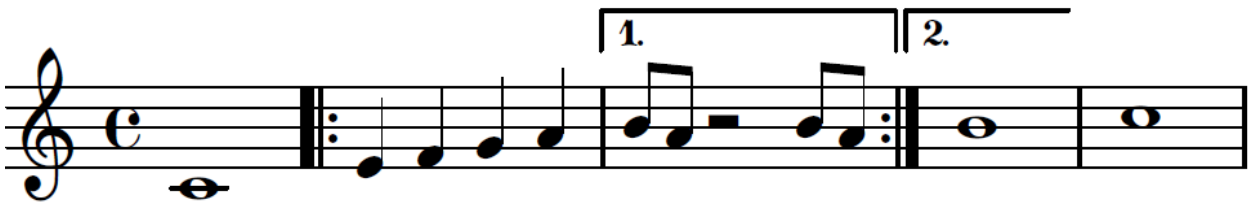
Obr. 9: Příklady syntaxe pomlk

2.3.5. Opakování

Opakování určité části notového zápisu se dá docílit pomocí příkazu `\repeat`. Syntaxe příkazu je následující: `\repeat typ_opakování počet_opakování noty_k_opakování.` Jazyk LilyPond definuje čtyři základní typy – `volta`, `unfold`, `percent` a `tremolo`.

- `volta` (Obr. 10) – reprezentuje standardní pojetí opakování (notový zápis je na začátku a na konci „obalen“ znaménky opakování (tlustá čára, tenká čára a dvojtečka)). Doplňující příkaz `\alternative` umožňuje měnit závěrečné pasáže jednotlivých opakování.

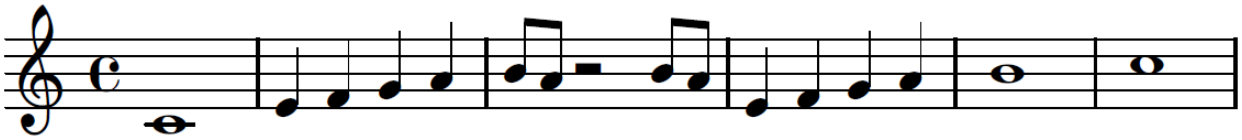
```
\relative c‘ {
   c1
   \repeat volta 2 {
      e4 f g a
   }
   \alternative {
      {b8 a8 r2 b8 a8}
      {b1}
   }
   c1
}
```



Obr. 10: Vzhled `\repeat volta`

- `unfold` (Obr. 11) – podobné jako volta. Jediný rozdíl je v tom, že se zde zápis vypíše celý, tedy v podobě, v jaké by byl ve skutečnosti zahrán.

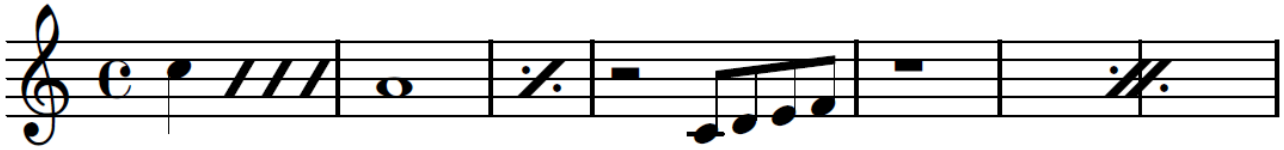
```
\relative c‘ {
   c1
   \repeat unfold 2 {
      e4 f g a
   }
   \alternative {
      {b8 a8 r2 b8 a8}
      {b1}
   }
   c1
}
```



Obr. 11: Vzhled `\repeat volta`

- `percent` (Obr. 12) – „procentuální“ opakování dovoluje opakovat určité noty, či sled not v rámci jednoho, či více taktů. Zápis se vždy poprvé vypíše pomocí not, při druhém (a vícenásobném) opakování se nahradí speciální značkou.

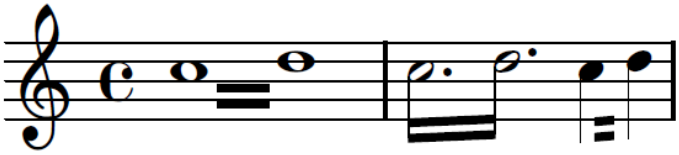
```
\relative c‘‘ {
   \repeat percent 4 {
      c4
   }
   \repeat percent 2 {
      a1
   }
   \repeat percent 2 {
      r2 c,4 d4 e4 f4 r2
   }
}
```



Obr. 12: Vzhled `\repeat percent`

- `tremolo` (Obr. 13) – tato opakování mohou být definována jak mezi dvojicemi not a harmonií, tak na jediné notě. V obou případech je však jejich význam stejný: rychlé střídání daných tónů (respektive rychlé opakování jediného).

```
\relative c‘‘ {
   \repeat tremolo 8 { c16 d }
   \repeat tremolo 6 { c16 d }
   \repeat tremolo 2 { c16 d }
}
```



Obr. 13: Vzhled `\repeat tremolo`[4]

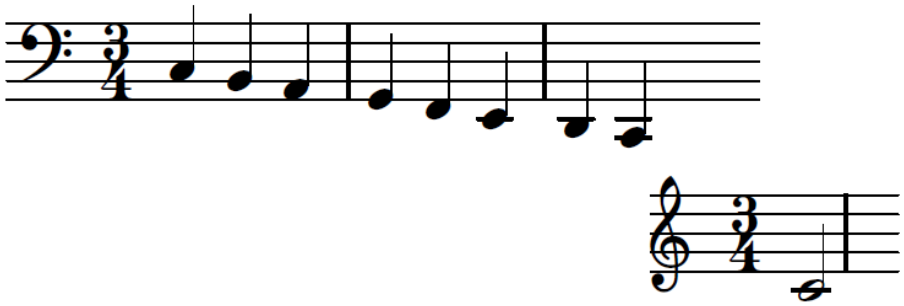
2.3.6. Vytváření proměnných a blok `\score`

Uzavřené celky partitury lze ukládat do proměnných a z nich pak složit celou výslednou skladbu. Při jejich používání je však situace poněkud složitější, neboť je nelze používat zcela nahodile. Ke správnému užití může dojít pouze ve druhé proměnné (kdy je první „vložena“ do té druhé) nebo v rámci bloku `\score`, umožňující spojování více notových zápisů do několika souběžně hrajících hlasů. Detailnější popis těchto možností však už přesahuje rámec základního nastínění syntaxe jazyka LilyPond.

```
promenna = \relative c {
   \clef “bass”
   c4 b a g
   f e d c
}
promenna2 = \relative c’{
   \time 3/4
   \promenna
   c2
}

\score {
   \promenna2
}
```

[4] Příklad převzat z oficiální dokumentace LilyPondu [4].



Obr. 14: Příklad použití proměnných a bloku `\score`

2.3.7. Velikost papíru a hlavička

Programovací jazyk LilyPond dovoluje manipulovat i s velikostí papíru, na němž bude notový zápis později vysázen. Předdefinované velikosti v sobě zahrnují drtivou většinu obvykle používaných formátů, které definuje mezinárodní standard ISO 216. Nechybí však ani německý rozšiřující standard DIN 476, či rozměry používané v Severní Americe [5].

Výběr je uskutečněn buď během překladu pomocí přepínače `-dpaper-size=\"název_velikosti\"` (např. `-dpaper-size=\"A6\"`[5]), či ve speciálním bloku nazvaném `\paper`, do nějž se vloží příkaz `#(set-paper-size "název_velikosti")`.

Blok `paper` taktéž dovoluje definovat své vlastní rozměry papíru a to buď v centimetrech, milimetrech, palcích nebo typografických bodech. LilyPond umožňuje úpravy vlastností samotného papíru (např. velikost, okraje), některých aspektů partitury (např. maximální délka řádku, velikost notové osnovy) či dokonce samotného procesu vysázení (např. maximální počet stránek, maximální počet řádků na jednom papíře, mezery mezi řádky a další).

Položky jako název skladby, její autor, či věnování se zapisují do části `\header` (čili hlavička). Hlavičkou lze opatřit buď celý dokument (vytiskne se pouze na první stránku), či jednotlivé notové zápisy (v tomto případě se hlavičky přidávají přímo nad první řádek dané osnovy). Do bloku hlavička spadají i informace o autorském právu (vlastnost `copyright`).

[5] Přítomnost zpětných lomítek je v tomto případě nutná [6].

2.4. Tvorba a překlad

Zdrojový kód jazyka LilyPond je možné psát několika různými způsoby. LilyPond k tomuto účelu přímo v instalačním balíku dodává aplikaci LilyPad, jejíž funkčnost se nijak zvlášť neliší od Poznámkového bloku a jemu podobných primitivních textových editorů. Jejich použití proto nic nebrání.

Překlad zdrojového kódu probíhá pomocí příkazové řádky. Minimální syntaxe volání překladače je následující: `lilypond zdrojový_kód.ly`, lze ji ovšem doplnit o celou řadu přepínačů a to jak základních, tak rozšiřujících (ty jsou označeny „předponou“ `-d`).

Daleko vhodnější a uživatelsky přívětivější volbou je program Frescobaldi[6]. Je zdarma dostupný, byl již přeložen do několika světových jazyků, poskytuje daleko přehlednější zpracování psaného kódu a některé části dokáže automaticky vygenerovat (například hlavičku). Kromě toho umožňuje generování notového zápisu a případné zvukové stopy přímo v okně aplikace. Uživatel se základními nároky tak nemusí s příkazovou řádkou vůbec přijít do styku.

2.5. Mutopia Project

Mutopia Project vznikl v návaznosti na nástroj LilyPond. Jeho úkolem je shromažďovat a třídit dostupné zdrojové kódy na základě autora daného díla, hudebního nástroje, historického období a dalších kategorií. Jedná se o dobrovolný a zdarma spravovaný virtuální „archiv“, který v době psaní této bakalářské práce obsahoval na sedmnáct set různých skladeb[7]. U každé z nich je k dispozici jak původní zdrojový kód, tak zvuková podoba díla a jeho notový zápis v několika formátech.

[6] Oficiální webové stránky: http://www.frescobaldi.org.

[7] Aktuální hodnota je k dispozici na stránce projektu http://www.mutopiaproject.org.

3. MIDI

3.1. Úvod, popis

MIDI, zkratka názvu Musical Instrument Digital Interface, je mezinárodní standard pro komunikaci mezi hudebními nástroji a dalšími elektronickými zařízeními. První verze vznikla na počátku osmdesátých let minulého století a díky svým kvalitám a univerzálnosti se používá dodnes.

3.2. Struktura a typ dat

Struktura MIDI souboru se dělí do dvou kategorií – hlavička a zvukové kanály. Hlavička musí být vždycky jenom jedna a uchovává důležité informace pro celou skladbu. Kanálů obvykle bývá více a nesou v sobě data o jednotlivých tónech, z nichž se skládá samotná skladba.

Všechno, co je v MIDI uloženo, je událost určitého typu. Název kanálu, určení délky taktu, zapnutí noty, vypnutí noty, aj. Při zpracování MIDI souborů pro potřeby této bakalářské práce byly nejdůležitější tři typy událostí:

- `SetTempoEvent` (česky „Nastav tempo“) – v určitý čas změní tempo na danou hodnotu,
- `NoteOnEvent` („Zapni notu“) – díky ní se ve skladbě ozývají tóny,
- `EndOfTrackEvent` („Konec kanálu“) – udává čas konce posledního tónu na daném kanále.

Poslední nutnou informací k synchronizaci notového zápisu je rozlišení. To je definováno v hlavičce, po celou dobu běhu zůstává stejné a určuje, kolik tiků trvá jedna čtvrťová nota. Čas v MIDI totiž není měřen pomocí sekund, ale tiků, které si lze zjednodušeně představit jako tikání metronomu.

3.3. MIDI a LilyPond

MIDI soubory mají ve zdrojovém kódu LilyPondu definovanou svoji vlastní část, kterou uvozuje příkaz `\midi`. Pokud uživatel chce pro svoji skladbu kromě grafického zápisu vytvořit také zvukový výstup, musí `\midi` vložit do bloku `\score`. Zatímco na unixových systémech vznikne soubor s příponou „mi-

di“, na systémech Windows LilyPond používá příponu „mid“. Tato určitá anomálie může být odstraněna použitím přepínače `-dmidi-extension=midi` při překladu.

Nutno upozornit, že možnosti zvukového výstupu jsou v LilyPondu částečně omezené. Především ne všechno, co uživatel do svého projektu vloží, se později projeví také v MIDI. Nejzásadnější škrty probíhají především v oblasti rytmu. Změny tempa se do MIDI vypropagují pouze v případě, kdy jsou zapsány jako konkrétní číselné hodnoty. Slovní označení jako `\largo` či `\allegro` jsou ignorována.

Převod také zcela ignoruje značky nad notami reprezentující artikulace a ornamenty [7]. Tento určitý nedostatek lze vynahradit použitím speciální externí knihovny `articulate.ly`[8]. Její přítomnost je však pro potřeby aplikace `ly2video` nežádoucí a je ze zdrojového kódu odstraněna (viz. podkapitola 4.4.).

Poslední a spíše drobná omezení jsou kladena na samotnou skladbu a na podobu zdrojového kódu. Každá notová osnova je totiž do MIDI přenesena jako jeden samostatný zvukový kanál, kterých může být maximálně patnáct (první je použit pro systémové informace, tedy hlavičku). Přebývající osnovy jsou poté ignorovány [8]. V kódu, přesněji řečeno v bloku `\score`, se pak musí objevit příkaz `\unfoldRepeats` (podrobněji je popsán v podkapitole 4.4.).

Pokud by jej uživatel nepřidal a jeho vytvořená skladba obsahovala nějaká opakování, v MIDI by tyto části skladby zazněly pouze jednou [9]. Elegantním řešení je vytvoření dvou `\score` bloků – jeden pro grafický výstup a druhý pro zvuk (i s použitím `\unfoldRepeats`).

3.4. Závěr

Popis nástroje LilyPond a formátu MIDI v této a předcházející kapitole byl důležitý především kvůli pochopení jeho základních vlastností. Ty byly využity při tvorbě programu `ly2video`, kterému se věnuje druhá polovina této bakalářské práce.

[8] Oficiální stránky projektu: http://www.nicta.com.au/people/chubbp/articulate.

4. Program `ly2video`

4.1. Úvod, motivace

Možnosti vytvoření animovaného notového zápisu jsou pro uživatele i v dnešní době stále ještě spíše omezené. Nejpropracovanější je z tohoto hlediska aplikace Sibelius[9], v níž lze (podobně jako pomocí LilyPondu) skládat a zapisovat hudbu a která je schopna při přehrávání zrovna otevřené skladby zobrazovat právě znějící tóny.

V tomto směru nelze vynechat ani hudební počítačové hry a výukové programy. Animace notového zápisu sice není v jejich případě to hlavní, přesto některé z nich nabízejí originální řešení (například výukový program MidiIllustrator[10]). Úplným extrémem jsou pak dětské hříčky, jako například Mario Paint[11]. Tato počítačová hra, jež vyšla původně v roce 1992 na konzoli Super Nintendo Entertainment System [10], dovoluje vytváření hudebních skladeb prostřednictvím obrázků postaviček a objektů ze známé skákačky.

Zcela samostatnou kapitolou jsou v tomto směru grafické vizualizace. Tento přístup převádí přehrávanou skladbu do směsice barevných obrazců, které jsou po obrazovce rozmístěny dle výšky tónů[12]. Během přehrávání skladby se pak tyto prvky buď zesvětlují, mizí či vybarvují. Podobně jako v případě „klasického“ zobrazování not existují ve formě přehrávačů (za všechny jmenujme například Music Animation Machine[13], MIDI Player[14] nebo dokonce trojrozměrný MIDITrail[15]), tak i počítačových her (třeba Synthesia[16], grafickým zpracováním velice podobná populární sérii Guitar Hero[17]).

[9] http://www.sibelius.com/home/index_flash.html

[10] http://www.midiillustrator.com

[11] Konverzi pro současné počítače lze nalézt na stránce http://www.unfungames.com/mariopaint/

[12] Příklad podobné animace: http://www.youtube.com/watch?v=Oq4-h20QTOg

[13] http://www.musanim.com

[14] http://www.musanim.com/player/

[15] http://www.sourceforge.jp/projects/miditrail/

[16] http://www.synthesiagame.com

[17] http://www.guitarhero.com

Všechny výše uvedené přístupy mají něco společného: buď se nejedná přímo o animaci notového zápisu (ale hudby znázorněné pomocí grafických prvků), nebo jsou v aplikaci umístěny pouze jako doplňková funkce při komponování hudby a nelze je samostatně použít. Uživatel si tak musí v těchto případech pomoci externími programy, které dovolují nahrávat obrazovku monitoru.

Tvorba programu, který by dokázal rozhýbat notový zápis na základě souběžně přehrávané skladby tak má smysl. Pohled na klasické noty je přirozenější než na různobarevné objekty. Animace nahrávané pomocí externích programů zase mohou trpět sníženou kvalitou obrazu, nehledě na to, že se jejich vzhled může lišit v závislosti na použitém hudebním softwaru. Vyšší náročnost procesu výroby podobného videa je pak už spíše okrajovým problémem.

Výběr nástroje GNU LilyPond je odůvodněn především tím, že je dostupný zcela zdarma (na rozdíl od některých výše jmenovaných programů), má jednoduchou syntaxi (viz podkapitola 2.3.) a vygenerovaný notový zápis vypadá esteticky zřejmě nejlépe, z čehož může výsledné video pouze těžit. „*LilyPond je inspirován tradičními, ručně tvořenými rytinami notových zápisů, které během první poloviny dvacátého století zveřejňovala evropská hudební vydavatelství, včetně společností Bärenreiter, Duhem, Durand, Hofmeister, Peters a Schott.*“ (Autorský tým LilyPondu, 2002, s. 4, přel. Szabó).

Možností, jak výstup z programu `ly2video` využít, je hned několik. Za nejpravděpodobnější a zřejmě nejčastější variantu v tuto chvíli považuji zveřejňování videí animovaného notového zápisu na zahraničním serveru YouTube a jejich prezentaci ostatním uživatelům Internetu. Stejně tak si však umím tuto aplikaci představit jako pomocníka při automatické tvorbě klipů k notovým materiálům. Tyto klipy by pak mohly sloužit jako učební pomůcka, ať už při výuce hudební nauky, historie, či při studiu hry na daný hudební nástroj.

4.2. Technické aspekty programu

Program `ly2video` je naprogramován ve skriptovacím jazyce Python. K jeho nastudování jsem využil publikaci autora Marka Summerfielda, Python 3: Výukový kurz [11]. Aplikace tak měla být původně psaná ve třetí plnohodnotné verzi jazyka Python. Kvůli nutnosti použití několika externích knihoven, z nichž některé ji zatím nepodporují, jsem byl nucen přejít na starší ver-

zi, 2.7. Změny v oblasti jazyka, kterou jsem při tvorbě `ly2video` využíval, jsou však minimální.

Během psaní zdrojového kódu jsem dodržoval následující konvence:

- anglické pojmenování proměnných,
- jména proměnných a funkcí začínají malým písmenem, u víceslovných názvů je každé další slovo uvozeno velkým znakem,
- maximální šířka řádku osmdesát znaků,
- anglicky psané komentáře a informační zprávy pro uživatele.

Pro spuštění a správnou funkčnost `ly2video` jsou kromě jazyka Python (verze 2.7) a LilyPondu zapotřebí také programy TiMidity++ [12] a FFmpeg [13]. Oba dva byly zvoleny především proto, že jsou zdarma dostupné a existují ve verzích pro většinu současných operačních systémů. FFmpeg jsem krom toho vybral také z toho důvodu, že s jeho používáním již mám určité zkušenosti z dřívějška. Podrobnější popis jejich rolí v `ly2video` je v podkapitole 4.7.

TiMidity++ je aplikace schopná přehrávat a zpracovávat zvukové soubory ve formátu MIDI. Jeho autorem je Japonec Masanao Izumo, jenž navázal na projekt Tuukka Toivonena pojmenovaný TiMidity 0.2i z roku 1995. Poslední stabilní verze pochází v října 2004. Hlavní výhodou je schopnost konvertovat MIDI soubory do kódování PCM[18] za pomocí externího softwarového syntetizátoru.

FFmpeg je oproti tomu nástroj pro práci a manipulaci s videem. Pomocí knihovny libavcodec, která v sobě obsahuje spoustu dnes běžně používaných formátů a kódování, dovoluje střih a práci s obrazem, vytváření videa ze sekvence obrázků (funguje i opačný postup), spojování zvuku s obrazem, výběr kódování zpracovávaného souboru a mnoho dalších funkcí.

4.3. Vstupní parametry

Uživatel může prostřednictvím několika definovaných vstupních parametrů, takzvaných přepínačů, ovlivňovat vlastnosti výsledného generovaného videa. Následující podkapitoly se zmiňují pouze o těch nejdůležitějších, kompletní seznam je k dispozici v příloze A.

[18] Pulse-code modulation, česky Pulzně kódová modulace.

4.4. Vstup a jeho úprava

Vstup programu `ly2video` představuje jediný soubor s příponou „.ly“ obsahující projekt vytvořený v programovacím jazyce LilyPond. Uživatel jej předává prostřednictvím přepínače `-i` (či `--input=`). Pokud na dané cestě neexistuje, program na standardní chybový výstup vypíše související chybovou hlášku a skončí s nenulovou návratovou hodnotou.

Jednotlivé verze LilyPondu mezi sebou bohužel nejsou v některých ohledech příliš kompatibilní. Jelikož je program `ly2video` napsán primárně pro poslední zveřejněnou verzi nástroje (tedy 2.14.2), mohou nastat při použití starších projektů problémy při překladu. Určitým řešením je zavolání pomocného skriptu `convert-ly`, který je dodáván přímo v instalačním balíku nástroje GNU LilyPond a který dokáže starší projekty převést do aktuální podoby. Transformace se však v některých případech nemusí podařit.

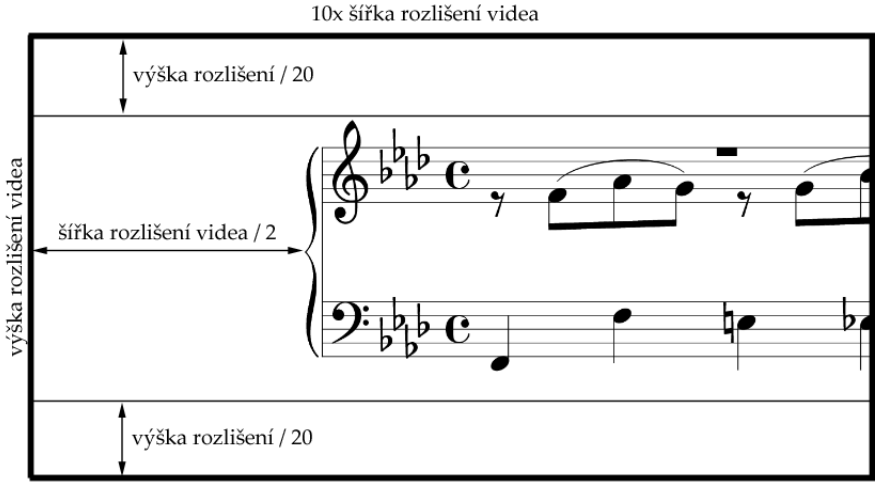
Vzhledem k tomu, že LilyPond dovoluje velice svobodný přístup k vytváření notového zápisu (viz kapitola 2.3.), je nutné obdržený projekt nejprve automaticky projít a některé části upravit, případně odstranit. Veškeré změny popisované v následujících několika odstavcích nejsou přenášeny i do původního projektu uživatele. Veškerý kód je při kontrole ukládán do dočasně vytvořeného souboru a následně načten do operační paměti kvůli rychlejšímu přístupu k jednotlivým řádkům.

Pořadí popisu úprav jednotlivých částí, které je uvedeno v této práci, není jedinečné. V kódu totiž tyto bloky figurují jako samostatné jednotky a lze je tedy libovolně řadit[19] (viz podkapitola 2.3.). Následující posloupnost je však dle mého názoru nejvhodnější a z pohledu člověka nejpřirozenější.

Jako první je nutno nalézt část kódu reprezentující hlavičku (anglicky `header`). Údaje v ní obsažené nejsou pro potřeby generovaného videa nijak důležité, výjimku tvoří pouze název skladby a její autor. Ty jsou využity při případném generování titulní strany (přepínač `--title-at-start`), která je umístěna ještě před samotné noty a má čistě informativní hodnotu. Hlavička je po nalezení potřebných údajů smazána.

[19] Výjimku tvoří proměnné, které musí být před prvním použití již definovány.

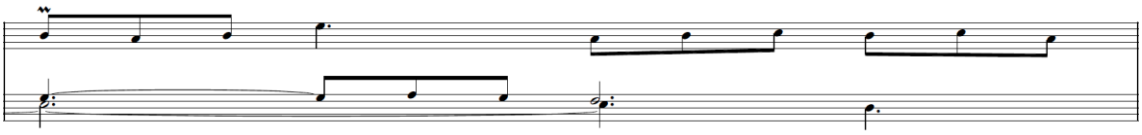
Další podstatnou součástí zdrojového kódu LilyPondu je nastavení rozměrů papíru (`paper`). Podobně jako v případě hlavičky nejsou uvedená data nikterak důležitá a program místo stávajícího nastavení použije své vlastní, uzpůsobené pro potřeby samotného generování (naznačeno na obrázku č. 15). Za blok `paper` je nakonec vložen příkaz `set-global-staff-size`, který určuje velikost notového zápisu a jehož hodnota je přibližně určena z výšky zbývajícího prostoru a počtu notových linek skladby.



Obr. 15: Používané rozměry papíru

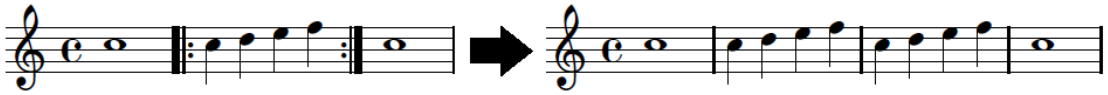
Počet linek je odvozen z náhledu notového zápisu, který LilyPond umožňuje vytvořit (přepínač `-dpreview`). Tato část programu zatím využívá předaný uživatelův projekt a po skončení výpočtu všechny vzniklé soubory maže. Odstraněna je i zvuková stopa v MIDI, která by sice mohla být již teď zachována pro pozdější použití, ovšem vzhledem k potenciální přítomnosti externí knihovny `articulate.ly` to není možné.

Také zápis notové osnovy musí projít určitými úpravami. Největším problémem z hlediska generování videa jsou příkazy, které umožňují manuální zalamování stránek – `\noBreak`, `\break` a `\pageBreak`. LilyPond se totiž v takovém případě snaží zaplnit notami celý dostupný prostor, což u papíru popsaném v předchozí části textu způsobí nepřirozené roztáhnutí (Obr. 16). Všechny výskyty těchto příkazů jsou proto odstraněny.



Obr. 16: Příklad roztáhnutí notového zápisu

Poslední úpravy jsou provedeny nad částí partitura (`score`). Je potřeba ihned za uvození daného bloku vložit příkaz `\unfoldRepeats`, který způsobí, že se všechna možná opakování v notovém zápise vypíšou ve své plné délce (Obr. 17). Jedná se o čistě estetické řešení z důvodu plynulejšího pohybu po notovém zápisu. S menšími obměnami by šlo použít i „klasické" verze s opakováními, na druhou stranu by to však znamenalo spoustu náhlých skoků, které by při sledování videa působily nepřirozeně, až rušivě.



Obr. 17: Příklad rozepsání not po aplikaci příkazu `\unfoldRepeats`

Zdrojový kód LilyPondu také může obsahovat několik příkazů, které přímo nespadají do kteréhokoliv z dříve popsaných bloků, přesto je potřeba na jejich přítomnost správně reagovat. Program se musí vypořádat především s těmito příkazy:

- `\include "articulate.ly"` a s ním související `\articulate` – dovoluje daleko přesnější a uhlazenější přenesení napsané skladby do formátu MIDI. Jeho přítomnost je však nežádoucí, neboť by bylo těžké odlišit nadbytečné události od těch, které reprezentují jednotlivé noty.
- `\pointAndClickOff` – zakazuje vkládání odkazů na zdrojový kód do notového zápisu uloženého ve formátu Adobe PDF (dále PDF). Ty jsou však pro funkčnost programu důležité, neboť je na nich založeno vyhledávání not (viz sekce 4.5.3.).
- (uživatelem určené) `set-global-staff-size` – musí být odstraněno z logického důvodu. Program definuje pro tuto proměnnou svoji vlastní hodnotu a pokud by ve zdrojovém kódu nechal i tuto, nemuselo by být vždy zcela jednoznačné, která se má použít.

- `\bookOutputName "uživatelův název"` – umožňuje upravit standardní chování LilyPondu při pojmenovávání výstupních souborů, při nichž se používá název uživatelova projektu. Vzhledem k tomu, že aplikace `ly2video` takové chování předpokládá, je nutno příkaz smazat.

Po skončení tohoto procesu je celý (upravený) projekt připraven k předání kompilátoru jazyka LilyPond, který zkontroluje jeho syntaxi a následně vygeneruje notový zápis ve formátu PDF, PNG a zvukový soubor ve formátu MIDI. Podoba druhého a posledního volání LilyPondu z příkazové řádky vypadá následovně:

```
lilypond -fpdf --png -dpoint-and-click -dmidi-extension=midi
ly2videoConvert.ly
```

Jednotlivé přepínače pak mají následující význam [14]:

- `-fpdf` – zajistí, aby LilyPond po skončení překladu nesmazal vytvořené PDF (v návaznosti na následující odrážku).
- `--png` – každou stranu dokumentu PDF uloží ve formátu PNG. Právě z těchto obrázků jsou generovány snímky pro výsledné video (viz podkapitola 4.6.).
- `-dpoint-and-click` – PDF bude obsahovat odkazy do zdrojového kódu. Ty se poté využijí pro nalezení pozic not v obrázcích PNG.
- `-dmidi-extension=midi` – vynucuje uvedenou příponu MIDI souboru (viz podkapitola 3.3).

4.5. Vyhledání pozic pro generování

4.5.1. Dva možné přístupy vyhledávání

Jednou z nejdůležitějších součástí procesu vzniku výsledného videa je nalezení pozic v obrázcích notového zápisu, které jsou vygenerovány LilyPondem. V tomto případě lze volit mezi dvěma zcela odlišnými přístupy řešení tohoto problému – vyhledávání prostřednictvím taktových čar, či prostřednictvím not.

4.5.2. Vyhledávání taktů

Metoda vyhledávání taktů pracuje pouze s vygenerovanými obrázky, snaží se nalézt taktové čáry a pomocí nich určit rozměry a pozice jednotlivých taktů. Tento přístup vychází z předpokladu, že délka skladby je přímo úměrná počtu taktů a čas, po nějž trvá každý z nich, je vždy stejný. Při vytváření videa tak již stačí pouze dodržovat tuto vypočtenou délku.

Nevýhod metody je hned několik. Ta nejdůležitější spočívá v tom, že obraz dost často nenabízí přesně to, co divák slyší. Číslo taktu sice bude vždy souhlasit, u jednotlivých not je však situace složitější. Jediná možnost, jak může stoprocentně souhlasit i jejich zobrazování je moment, kdy mají všechny noty stejnou délku a jsou tedy rozloženy rovnoměrně po celé šířce taktu. Daleko častěji dochází ke zcela opačné variantě – některé noty zazněly dříve, než se dostaly do středu videa a naopak. Proto je daleko vhodnější využít vyhledávání not.

4.5.3. Vyhledávání not

Druhým možným řešením je vyhledávání not. Základní myšlenka je velice podobná té předchozí – program vyhledá pozice všech not v zápise a při generování pracuje se vzdáleností jednotlivých dvojic. Již na první pohled je tedy patrné, že vyhledávání not je z hlediska synchronizace zvuku a obrazu daleko přesnější. Daný tón se na středu videa objeví právě v tu chvíli, kdy jej divák také slyší.

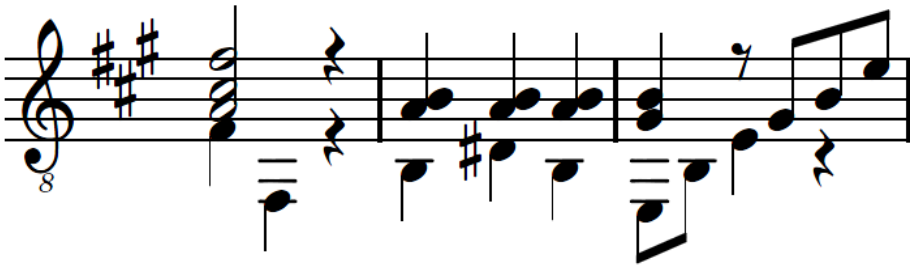
I tento přístup však má pár nevýhod. Pohyb přes jednotlivé noty totiž v některých případech není příliš plynulý a dokonce může docházet i k poměrně razantním skokům. Jejich příčinnou jsou především ozdobné noty, které se hrají velice rychle a tím pádem mohou být zobrazeny pouze po velice krátký časový úsek.

Samotný proces vyhledávání je daleko složitější než v předchozím případě. Samozřejmě, opět by šlo postupovat obdobnou metodou z předchozí podkapitoly, kdy dochází k procházení vytvořených obrázků. Daleko snadnější a efektivnější je ovšem využití služeb, které k tomuto účelu nepřímo poskytuje samotný LilyPond. Notový zápis ve formátu PDF totiž lze prostřednictvím přepínače `-dpoint-and-click` doplnit o odkazy přímo na zdrojový kód skladby, čehož lze elegantně využít.

Přímočaré řešení je tedy nasnadě. Stačí si ze všech odkazů vybrat pouze odkazy na noty a následně určit pozici každé z nich. Výpočet probíhá pomocí souřadnic samotného odkazu na dané stránce. PDF však používá jiné jednotky než obrázky (tedy pixely) a je proto nutno užít malého triku. Ten vychází ze skutečnosti, že pozice daného objektu je na stránce i obrázku stejná. Tím pádem stačí vypočítat poměr mezi umístěním objektu a šířkou strany a následně jej vynásobit délkou souvisejícího obrázku v pixelech.

Nutno dodat, že ne každá nalezená nota může být prohlášena za platnou. Některé z nich totiž mohou být součástí ligatury a ve vygenerovaném zvukovém záznamu nezazní. Proto je potřeba, aby se ještě před samotným vyhledáváním prozkoumal zdrojový kód LilyPondu a byly určeny noty, které se v rámci výše popsaného vztahu vynechají.

Tento na první pohled jednoduchý přístup má však jednu zásadní chybu. Pokud se v zápise objeví několik not, které zazní ve stejnou chvíli a navíc jsou na osnově relativně blízko sebe, LilyPond je v rámci přehlednosti vykreslí po obou stranách nožičky[20] (Obr. 18). Metoda nastíněná v předchozím odstavci ovšem nic takového nepředpokládá a skupinu not identifikuje jako dvě rozdílné. Celý systém v tuto chvíli selže, neboť při synchronizaci bude tato „dvojitá“ nota přebývat.



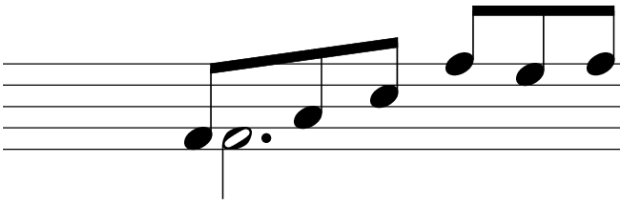
Obr. 18: Příklady vysázení po obou stranách nožičky

Vzhledem k tomu, že každý odkaz má určenu svoji oblast platnosti[21], nabízí se na první pohled relativně jednoduché řešení popsaného problému. Teoreticky by mělo stačit si před konečným akceptováním pozice zkontrolovat, zda se oblast odkazu dané noty nedotýká, či dokonce nepřekrývá s oblastí té předcházející. Tato úprava způsobí, že systém již funguje vcelku spolehlivě, nedokáže ovšem obstát v situaci, kdy je na jednu notovou osnovu vykresleno více hlasů

[20] Stejný problém může nastat při vykreslení více hlasů na jednu notovou osnovu.

[21] Jinak řečeno oblast, která na kliknutí myší reaguje přesměrováním na daný odkaz.

zároveň. Může se totiž stát, že ten stejný tón zazní v obou hlasech ve stejnou chvíli a každý navíc s jinou délkou (Obr. 19).



Obr. 19: Příklad problémového vysázení dvou hlasů na jedné osnově

Příliš nepomůže ani řešení, v němž by se kontrolované okolí noty uměle roztáhlo o několik pixelů doleva a doprava. Problému popsaném v předchozím odstavci by se sice zabránilo, na druhou stranu by však docházelo k nesprávnému ignorování některých jiných not, především pokud by šlo o jejich rychlý sled.

K celému problému vyhledávání je tedy nutno přistoupit o něco složitějším a robustnějším způsobem, který v sobě zahrnuje i souběžné zpracování zvukové stopy ve formátu MIDI. Nejprve je nutno zjistit, kolik událostí reprezentující sepnutí noty se v daný časový okamžik spustí (viz podkapitola 3.2.). Následně se podobné procházení aplikuje při hledání indexů not (tedy kolik se jich vyskytuje na zjištěných pozicích) a začnou se tyto dvě informace porovnávat. Pokud počet událostí v MIDI a not na obrázku v daný časový okamžik sedí, pozice je označena za platnou. Při menším počtu událostí se pak následující index z obrázku vynechává, neboť je zřejmé, že se chybějící noty nacházejí právě zde.

Ani tato metoda však neřeší problém naznačený na obrázku č. 19, neboť se pro danou výseč osnovy vygeneruje šest MIDI událostí, zatímco pozic na obrázku bude sedm. Potenciálním řešením je krom porovnávání počtu not v MIDI a na obrázku také kontrola, zda sedí i jejich názvy.

4.6. Samotná synchronizace

4.6.1. Modelová situace

Přistupme nyní k samotné synchronizaci, v níž se využívá pozic nalezených v souboru MIDI a také na obrázcích notového zápisu. Generování obrazové stopy probíhá v programu `ly2video` pomocí jednoduchého algoritmu, který bude nejprve demonstrován na modelovém příkladě.

Předpokládejme, že máme dvojici pozic not na obrázku, které jsou od sebe vzdáleny šedesát pixelů. Na časové ose je pak dělí přesně jedna sekunda a uživatel nepoužil přepínač `-f` (respektive `--fps=`). Pro tuto pasáž videa je tedy potřeba vygenerovat třicet snímků, což je standardní hodnota snímkové frekvence. Potřebujeme třicet obrázků, vzdálenost obou not je šedesát pixelů. Posun pomyslného okna, které „stříhá" obrázky pro video, jsou dva pixely.

4.6.2. Synchronizace podrobněji

V praxi synchronizace obrazu se zvukem probíhá o něco složitěji. Jako první se zkontroluje, zda uživatel nepožaduje před samotné video vytvoření titulní strany. Pokud tomu tak je, musí se nejprve vygenerovat obrázky obsahující název skladby a jejího autora, které byly předtím získány z hlavičky ve zdrojovém kódu (viz. podkapitola 4.4.). Teprve až poté se může přistoupit k notovému zápisu.

Samotný výpočet se pak příliš neliší od modelové situace nastíněné v předchozí podkapitole. Nejzásadnější rozdíl spočívá v tom, že čas není v MIDI uložen v sekundách, ale takzvaných „ticích" (viz podkapitola 3.2.). Proto je nutno jej nejprve přetransformovat do sekund, což se provede následující rovnicí:

$$\text{délka (sekundy)} = \frac{\text{právě používané tempo}}{\text{rozlišení MIDI}} * \frac{\text{délka (tiky)}}{1\ 000\ 000}$$

Počet obrázků pro video pak dostaneme vynásobením snímkovou frekvencí. Hodnota však jenom výjimečně vychází jako celé číslo, proto je nutno ji vždy zaokrouhlovat.

Vzhledem k zaokrouhlování počtu snímků ke generování je pak ještě nutno se vypořádat se vznikajícími „přebytky". K tomuto účelu je zavedena speciální proměnná, která rozdíly mezi přesnou a zaokrouhlenou hodnotou eviduje a pokud jejich součet přesáhne hodnotu jedné, musí být následující snímek přeskočen. Kdyby tomu tak nebylo, u delších skladeb by postupně docházelo k zpoždění zvuku vůči obrazu.

Jako poslední pak přichází na řadu samotné generování potřebných snímků, přidání čáry zobrazující momentálně znějící noty a následné uložení obrázků s příslušným pořadovým číslem.

4.7. Výstup

`Ly2video` vstupuje do své finální fáze. Na závěr dochází k převedení zvukové stopy ve formátu MIDI do kódování PCM (a tedy formátu WAV[22]), transformování vytvořené sekvence obrázků do podoby videa a jejich sloučení se zvukem do finálního výstupu.

Právě k tomuto účelu jsou v aplikaci přítomny i oba dva externí programy, detailněji popsané v podkapitole 4.2., TiMidity++ a FFmpeg. Jako první je zavoláno TiMidity++, které pomocí příkazu `timidity ly2videoConvert.midi -Ow` provede požadovanou konverzi. Pojmenování souboru `ly2videoConvert` je odvozeno od názvu dočasně vytvořeného projektu, jehož vznik je detailněji rozebrán v podkapitole 4.4. Přepínač `-Ow` zajistí převod do formátu WAV [15].

FFmpeg oproti tomu převádí vygenerovanou sekvenci obrázků do videa a přidává zvukovou stopu, která vznikla pomocí TiMidity++. Syntaxe volání vypadá následovně:

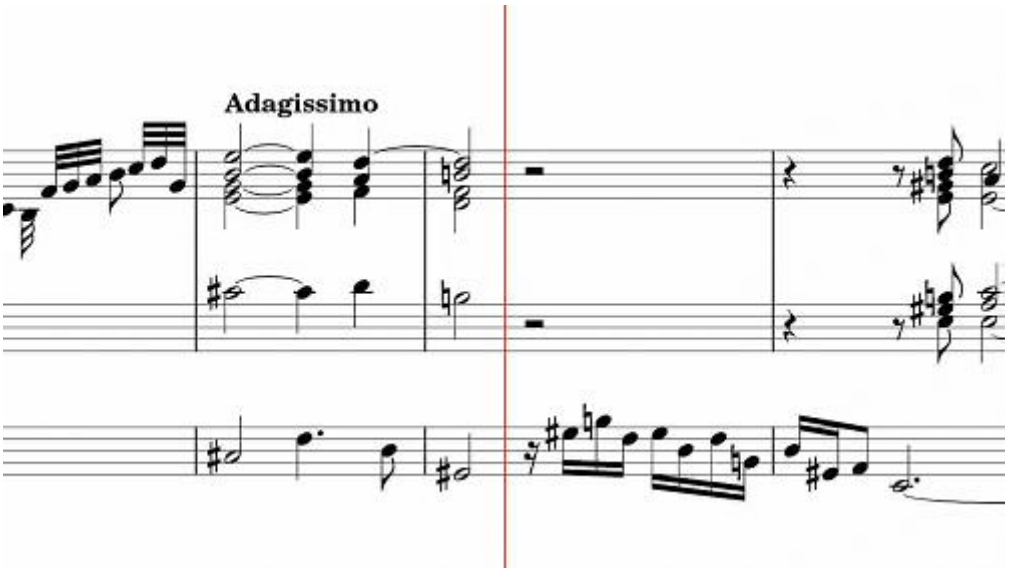
```
ffmpeg -f image2 -r snímková_frekvence -i .\frames\frame%d.png
-i ly2videoConvert.wav název_výstupu
```

Přepínače uvedené v tomto příkazu mají následující význam [16]:

- `-f` – definuje, o jaký způsob transformace jde. Slovo `image2` značí převod posloupnosti obrázků do videa.
- `-r` – určuje požadovanou výslednou snímkovou frekvenci. Hodnota používaná v příkaze je proměnná, neboť ji uživatel může prostřednictvím přepínače `–f` (respektive `--fps=`) při spuštění `ly2video` změnit.
- `-i` – označuje vstupní data, tedy obrázky a následně zvukovou stopu. Značka `%d` v prvním případě reprezentuje pořadové číslo daného snímku.

Závěrem program uživatele informuje, že byl celý převod ukončen a že výstup z programu nalezne v souboru, jehož název si zvolil pomocí přepínače `-o` nebo `--output=`. Pokud tak neučinil, je jméno generovaného videa odvozeno od pojmenování vstupního projektu.

[22] Waveform Audio File Format.



Obr. 20: Snímek výsledného videa[23]

Nutno upozornit, že pokud uživatel požadoval také vygenerování titulní stránky, je převod o něco složitější. Obě části videa se totiž musí vytvořit samostatně (bez ztráty kvality – přepínač `-same_quant`), poté pomocí unixového příkazu `cat` sloučit[24] a následně znovu zavolat FFmpeg. Tento nástroj totiž neumožňuje spojovat více videí dohromady přímo v rámci jednoho volání. Výše uvedený postup pochází přímo z oficiálního zdroje [17].

4.8. Nedostatky

4.8.1. Parciální zápis hudby

Nejzásadnějším problémem jsou projekty, které byly napsány pomocí parciálního zápisu (nebo alespoň některé jejich části). Tento příkaz dovoluje psát určité části kódu ne přes celé partitury, ale přes celé takty (jinak řečeno ne vodorovně, ale svisle).

Podobný zápis způsobuje problém především při vyhledávání pozic not. Vzhledem k tomu, že se program musí vypořádat s notami patřícími do ligatury, nejprve nalezne veškeré noty a tato spojení. Poté při procházení vždy ignoruje následující notu po značce ligatury v kódu (tedy po vlnce).

[23] Několik vygenerovaných videí lze nalézt na adrese http://www.youtube.com/playlist?list=PL444F0513202699C4.

[24] Na operačním systému Windows je k tomuto účelu příkaz `copy`.

Vzhledem k tomu, že ligatura může přesahovat okraj taktu, parciální zápis zapříčiní, že se mohou přeskakovat noty, které leží ve stejném taktu jako ligatura, pouze o notovou osnovu níže. Později tak může dojít k selhání procesu vyhledávání indexů či ke špatné synchronizaci. V nejhorším případě celá aplikace předčasně skončí.

Potenciálním řešením může být úprava vyhledávání pozic ligatur a všech not, které by nepoužívalo zdrojový kód uživatele, ale interní interpretaci zápisu v LilyPondu. Vzhledem k tomu, že nástroj je z valné většiny napsán v jazycích C++ a Scheme [1], vyžadoval by si tento přístup detailnější a dlouhodobější studium.

4.8.2. Přidávání titulní strany

Také přidávání titulní strany do výsledného videa je možná až příliš zbytečně složité. V ideálním případě by se snímky pro titulku a notový zápis generovaly do jediné sekvence a přidávaná zvuková stopa by se o daný počet sekund na začátku zpozdila. Nastíněný postup ovšem selhává u zdržení audia.

První potenciální možností by bylo odložit start zvuku až při transformaci sekvence obrázků do videa pomocí FFmpegu, konkrétněji pomocí příkazu `-itsoffset` [18]. Tento přepínač se vkládá přímo před vstupní soubor, na který se má aplikovat a lze mu předat buď čistě sekundy nebo složitější zápis ve formátu `hodiny:minuty:vteřiny.tisíciny_vteřiny`. Bohužel však dokáže na požadovanou dobu zastavit pouze obraz, zvuk jeho přítomnost ignoruje.

Při nemožnosti využít služeb FFmpeg se nabízí použití TiMidity++. Naneštěstí ani tento program nedokáže pozdržet převáděný zvuk a vložit na začátek několik vteřin trvající ticho.

Nabízí se tedy otázka, proč `ly2video` nevyužívá jiný software, který dokáže soubory MIDI překonvertovat do jiného formátu a navíc podobnou funkci obsahuje. Problém je v tom, že se během vývoje nepodařilo najít jinou aplikaci, jenž by pracovala v příkazové řádce, byla by dostupná na všech hlavních operačních systémech a dokázala by nabídnout stejnou funkcionalitu, jako právě TiMidity++ (viz podkapitola 4.2.).

Poslední možností je manuální posunutí všech událostí v MIDI souboru. Délka titulní strany však musí být ještě předtím převedena ze sekund na tiky, což se provede upravenou verzí rovnice zmíněné v podkapitole 4.6.2. Toto řešení ovšem není vzhledem k nutnosti podobného převodu a následného zaokrouhlování příliš šťastné a je zde uvedeno pouze pro úplnost.

5. Závěr

Možností k dalšímu vývoji se momentálně nabízí několik. Kromě odstranění nedostatků zmiňovaných v podkapitole 4.8. by mohlo být přínosné kontaktovat vývojáře textového editoru Frescobaldi a pokusit se zakomponovat `ly2video` jako jednu z nadstavbových funkcí, kterou jejich program nabízí. S velkou pravděpodobností by tak došlo k masivnějšímu rozšíření `ly2video` i mezi méně zkušené uživatele LilyPondu.

Dalším potenciálním směrem tvorby může být automatické nahrávání vygenerovaného videa na YouTube. Tento nápad vychází z předpokladu popsaném v podkapitole 4.1., podle nějž bude ve většině případů výstup z programu `ly2video` použit právě pro jeho zveřejnění na zmiňovaný zahraniční server.

Samozřejmě by záleželo na tom, jakým způsobem je nastavena vnitřní politika YouTube a zda je něco podobného vůbec uskutečnitelné. Během posledních let totiž server prošel několika velice zásadními proměnami, díky nimž programy s podobnou funkcionalitou přestávaly fungovat.

Troufám si tvrdit, že největší přínos programu `ly2video` je vidět na první pohled. Jak popisuje úvod ke čtvrté kapitole, vůbec poprvé se v oblasti počítačové tvorby notové partitury objevuje zdarma dostupná aplikace, která dokáže z předloženého notového zápisu vygenerovat video, v němž se na základě přehrávaných tónů plynule pohybuje samotná partitura. Možná je lehce nespravedlivé, že bude dostupná pouze lidem používající LilyPond, vzhledem k estetickým vlastnostem nástroje je tato volba logická (viz. kapitola č. 2).

Tato práce popisuje aplikaci `ly2video` ve verzi 1.0. Program byl představen uživatelům GNU LilyPond a zveřejněn k volnému použití. Dá se předpokládat, že bude kladně přijat, soudě podle předchozích požadavků uvnitř komunity[25].

[25] Například až doposud jediný návod pro tvorbu synchronizovaného videa, který je dostupný na adrese http://permalink.gmane.org/gmane.comp.gnu.lilypond.devel/13371.

Seznam literatury

[1] Autor neznámý. GNU LilyPond Music Typesetter. *Ohloh Analysis* [online]. 2012. [cit. 13. května 2012]. Dostupné na: <http://www.ohloh.net/p/lilypond/analyses/latest>.

[2] NIENHUYS, Han-Wen a NIEUWENHUIZEN, Jan. Předmluva. *GNU LilyPond — Learning Manual* [online]. Utrecht/Eindhoven, Nizozemsko, červenec 2002 [cit. 10. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond-learning/Preface#Preface>.

[3] PERCIVAL, Graham. LilyPond 2.14.2 released. *Info-LilyPond* [online]. 25. července 2011 [cit. 10. května 2012]. Dostupné na: <http://lists.gnu.org/archive/html/info-lilypond/2011-07/msg00002.html>.

[4] Vývojářský tým GNU LilyPond. Tremolo repeats. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 14. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond/Short-repeats#Tremolo-repeats>.

[5] Vývojářský tým GNU LilyPond. Paper size. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 12. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond/Paper-size>.

[6] Vývojářský tým GNU LilyPond. Command line options for LilyPond. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 12. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond-program/Command-line-options-for-lilypond>.

[7] Vývojářský tým GNU LilyPond. What goes into the MIDI output?. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 10. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond/What-goes-into-the-MIDI-output_003f>.

[8] Vývojářský tým GNU LilyPond. MIDI output. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 12. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.14/Documentation/notation/midi-output>.

[9] Vývojářský tým GNU LilyPond. Repeats in MIDI. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 9. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.12/Documentation/user/lilypond/Repeats-in-MIDI#Repeats-in-MIDI>.

[10] Autor neznámý. Mario Paint. In Soubor neznámých autorů. *Wikipedia, the free encyclopedia* [online]. 2010. [cit. 16. května 2012]. Dostupné na: <http://en.wikipedia.org/wiki/Mario_Paint_Composer>.

[11] SUMMERFIELD, Mark. *Python 3: Výukový kurz*. Brno: Computer Press, a.s., 2010

[12] IZUMO, Masanao. What is TiMidity++. *TiMidity++ official website* [online]. 2004. [cit. 4. května 2012]. Dostupné na: <http://timidity.sourceforge.net/#info>.

[13] Vývojářský tým FFmpeg. General Documentation. *FFmpeg official website* [online]. 17. května 2012. [cit. 17. května 2012]. Dostupné na: <http://www.ffmpeg.org/general.html#SEC6>.

[14] Vývojářský tým GNU LilyPond. Basic command line options for LilyPond. *GNU LilyPond — Learning Manual* [online]. 2009 [cit. 13. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.15/Documentation/usage/command_002dline-usage#basic-command-line-options-for-lilypond>.

[15] IZUMO, Masanao. TiMidity++ command line option. *TiMidity++* [software]. 2004. [cit. 15. května 2012]. Dostupné na: <http://bluewing.usamimi.info/timidity/index.php?p=optdoc&lang=en>.

[16] Vývojářský tým FFmpeg. ffmpeg Documentation. *FFmpeg official website* [online]. 17. května 2012. [cit. 17. května 2012]. Dostupné na: <http://ffmpeg.org/ffmpeg.html>.

[17] Vývojářský tým FFmpeg. FFmpeg FAQ. *FFmpeg official website* [online]. 17. května 2012. [cit. 17. května 2012]. Dostupné na: <http://ffmpeg.org/faq.html#How-can-I-join-video-files_003f>.

[18] PRITCHETT, Howard. Using ffmpeg to manipulate audio and video files. *LINUX "howto" pages* [online]. 10. října 2006 [cit. 5. května 2012]. Dostupné na: <http://howto-pages.org/ffmpeg/>.

[19] Neznámý autor. How do I change video quality? In Soubor neznámých autorů. *YouTube Help* [online]. 2005. [cit. 2. května 2012]. Dostupné na: <http://support.google.com/youtube/bin/answer.py?hl=en-GB&answer=91449>.

Autorský tým LilyPondu. *Essay on automated music engraving* [online]. 2002. [cit. 17. května 2012]. Dostupné na: <http://lilypond.org/doc/v2.14/Documentation/essay.pdf>.

Příloha A - přepínače programu `ly2video`

`-i SOUBOR, --input=SOUBOR`

Slouží k předání vstupního projektu GNU LilyPond pro vygenerování videa (soubory s příponou *.ly). Lze využít relativní, či absolutní adresování.

`-o SOUBOR, --output=SOUBOR`

Určuje, pod jakým jménem a případně do jakého adresáře má být vygenerované video uloženo. Pokud přepínač není využit, program použije název předávaného projektu a soubor uloží do stejného adresáře.

`-c BARVA, --color=BARVA`

Dovoluje měnit barvu čáry, jenž se pohybuje po notové osnově a zobrazuje právě přehrávané noty. Uživateli je k dispozici několik základních barev, které přepínači předává prostřednictvím jejich anglických názvů. Konkrétně lze použít černou (`black`), červenou (`red`), hnědou (`brown`), modrou (`blue`), zelenou (`green`) a žlutou (`yellow`) barvu. Standardní je červená.

`-f HODNOTA, --fps=HODNOTA`

Zkratka „fps“ vychází z anglického sousloví „Frames per second“ (česky „Počet snímků za sekundu“ či „Snímková frekvence“). Určuje, kolik obrázků bude ve výsledném videu reprezentovat jednu sekundu. Standardní hodnota je stanovena na třicet.

`-r VÝŠKA, --resolution=VÝŠKA`

Určuje výšku obrazu generovaného videa v pixelech (šířka se poté dopočítává za pomoci používaného poměru stran 16:9). Povolené hodnoty vycházejí ze standardu zahraničního serveru YouTube, jenž umožňuje přehrávání uložených videí v třech základních rozlišení – 640x360 (pokrývá jej hodnota `360`), 1280x720 (`720`, standardní hodnota) a 1920x1080 (`1080`) [19].

`-h, --help`

Vypíše nápovědu na standardní výstup. Ta nabízí nejprve správnou syntaxi spuštění programu na příkazové řádce a poté seznam veškerých přepínačů s krátkým slovním popisem.

`--title-at-start`

Na začátek videa umístí prvotní snímek, na němž je uveden název a autor skladby. Čas, po nějž je titulní obrázek zobrazen, lze upravit následujícím přepínačem.

`--title-delay=HODNOTA`

Určuje, jak dlouhou dobu bude zobrazen název skladby a její autor (viz předchozí `--title-at-start`). Uživatel přepínači předává počet požadovaných vteřin, základní verze počítá se třemi.

`--windows-timidity=CESTA`

Umožňuje uživatelům systému Windows předat programu pozici složky, v níž se nachází spustitelný soubor timidity.exe.

`--windows-ffmpeg=CESTA`

Umožňuje uživatelům systému Windows předat programu pozici složky, v níž se nachází spustitelný soubor ffmpeg.exe.