Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Optimalizace práce v aplikačním SW

Jaroslav Šindler

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jaroslav Šindler

Systémové inženýrství a informatika
Informatika

Název práce

Optimalizace práce v aplikačním SW

Název anglicky

Optimalisation of working in application SW

Cíle práce

Bakalářská práce je tematicky zaměřena na optimalizaci a zefektivnění práce v aplikačním SW. Hlavním cílem práce je identifikace a analýza nedostatků, návrh a realizace řešení pro efektivnější práci ve zvoleném aplikačním SW. Dílčí cíle práce jsou:
- charakteristika zvoleného aplikačního SW,
- charakteristika zvolených programovacích jazyků.

Metodika

Metodika řešené problematiky je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v identifikaci a analýze nedostatků zvoleného aplikačního SW s následným návrhem řešení a vlastní realizací pro zefektivnění práce uživatele. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

40 – 50 stran textu.

Klíčová slova

Sibelius, ManuScript, plug-in, notační software, programovací jazyk

---

Doporučené zdroje informací

AVID. Extra plug-ins for Sibelius. Sibelius.com [online]. @2016 [cit. 20. 1. 2018]. Dostupné z: http://www.sibelius.com/download/plugins/index.html.
CLARKE, Tom. Sibelius® 7: Tutorials [online]. [s.l.]: Avid Technology Inc., ©2011. 124 s.
SEBESTA, Robert W. Concepts of Programming Languages. Boston: Addison Wesley, 2009. 795 s.
Sibelius® Reference Guide [online]. [s.l.]: Avid Technology, Inc., ©2017. 849 s.
Sibelius® Software: Using the ManuScript Language [online]. [s.l.]: Avid Technology, Inc., ©2017. 173 s.
ZELINGER, Ivo. Notografie. Praha: Supraphon, 1986. 197 s.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Pavel Šimek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 11. 9. 2018

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 10. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 25. 08. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Optimalizace práce v aplikačním SW“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14. 3. 2021 ……

Poděkování

Rád bych touto cestou poděkoval Ing. Pavlu Šimkovi, PhD. za odborné vedení, cenné rady a připomínky při tvorbě této práce.

Dále Miloši Kudělkovi, Aničce Brdičkové a Filipu Benešovi za odborné i neodborné rady, trpělivost a pomoc s překlady do i z angličtiny. A především Danušce Bimkové za vše zmíněné, a navíc za přečtení a pečlivou korekturu práce a poskytnuté životní útočiště.

V neposlední řadě děkuji své rodině, hlavně rodičům a Lidušce, všem kamarádům a přátelům za podporu na mé studijní i životní cestě.

OPTIMALIZACE PRÁCE V APLIKAČNÍM SW

Abstrakt

Bakalářská práce v první části popisuje teoretická východiska – aplikační software se zaměřením na notační programy a stručně charakterizuje nejrozšířenější zástupce notačních softwarů. Nejpodrobněji se věnuje programu Sibelius, se kterým je pracováno v druhé části práce. Je charakterizován výběr jeho funkcionalit, jeho uživatelské rozhraní, a hlavně tvorba plug-inů pro tento program ve skriptovacím jazyce ManuScript.

V druhé části práce jsou analyzovány nedostatky a neefektivní postupy práce v programu a je navrženo jejich řešení, případně zefektivnění či zautomatizování postupů pomocí tvorby vlastních plug-inů v jazyce ManuScript. Je charakterizován jejich návrh, metody a jednotlivé prvky a následně je na příkladu předvedeno, jak lze plug-in použít v praxi.

Klíčová slova: Sibelius, ManuScript, plug-in, zásuvný modul, notační software, skriptovací jazyk, optimalizace

OPTIMIZATION OF WORKING IN APPLICATION SW

Abstract

First part of the thesis describes the theoretical background — application software with focus on notation programs. It briefly characterizes the most common scorewriters, the main emphasis is on Sibelius. It describes the selection of its functionality, user interface and especially the creation of plug-ins for this program in the scripting language ManuScript.

The second part analyzes the shortcomings and inefficient procedures in the program and proposes their solution — streamlining or automating these procedures by creating plug-ins in ManuScript. Their design, methods and elements are characterized, and then the example shows how the plug-in can be used in practice.

Keywords: Sibelius, ManuScript, plug-in, notation software, scorewriter, scripting language, optimization

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD

Aplikační software (nebo zkráceně aplikace) značí programové vybavení počítače, které zajišťuje přímou práci uživatele s počítačem – plní uživatelovy konkrétní požadavky. Tato práce se zaměřuje především na specializovaný typ aplikací – notační programy. (1–3)

Notační software (v angličtině *music notation software* či *scorewriter*) slouží k vytváření notových partitur a následné práci s ní (přehrávání, exportování, sdílení s jinými uživateli aj.). Jedná se o programy využívané především muzikanty, skladateli, aranžéry, hudebními pedagogy, vydavatelstvími hudební literatury apod. (3–5)

Existuje mnoho notačních programů; v práci bude stručně charakterizováno několik nejpoužívanějších. Největší důraz je kladen na program Sibelius, se kterým je pracováno i v druhé části práce. Důvodem k výběru této aplikace je jeho široká uživatelská základna, a především možnost tvorby vlastních plug-inů. (6, 7)

Plug-in neboli zásuvný modul je program, jenž funguje uvnitř jiného programu a rozšiřuje jeho funkcionalitu. V softwaru Sibelius jsou plug-iny psány ve jazyce ManuScript. Jedná se o jednoduchý hudební programovací jazyk založený na skriptovacím jazyce Simkin vyvinutém v roce 1999. Později byl jeho tvůrcem a dalšími lidmi rozšířen pro program Sibelius a pojmenován ManuScript. (8)

Sibelius je velmi silným nástrojem ve své kategorii, ale (jako ostatně asi žádný software) nemůže pokrýt všechny požadavky uživatelů nebo přinejmenším nedokáže všechny zpracovat efektivní a snadno přístupnou cestou. Řešením takových situací může být plug-in, který daný postup zefektivní či zautomatizuje a ušetří tak uživateli práci i čas. Těch pro tento software byly napsány již stovky, přesto lze nalézt postupy práce, které lze vytvořením vlastních zásuvných modulů optimalizovat či automatizovat. (8, 9)

V práci je používáno několik hudebních termínů, jejichž význam není vždy vysvětlen. Pro plné pochopení obsahu práce je tak předpokládána základní znalost notační a hudební terminologie.

2 CÍL PRÁCE A METODIKA

2.1 CÍL PRÁCE

Bakalářská práce je tematicky zaměřena na optimalizaci a zefektivnění práce v aplikačním SW. Hlavním cílem práce je identifikace a analýza nedostatků, návrh a realizace řešení pro efektivnější práci ve zvoleném aplikačním SW. Dílčí cíle práce jsou:

- charakteristika zvoleného aplikačního SW,
- charakteristika zvolených programovacích jazyků.

2.2 METODIKA

Metodika řešené problematiky je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v identifikaci a analýze nedostatků zvoleného aplikačního SW s následným návrhem řešení a vlastní realizací pro zefektivnění práce uživatele. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

Ve vlastní práci je pracováno s notačním programem Sibelius – jedná se o jeden z nejpoužívanějších a nejprodávanějších softwarů ve své kategorii. Dalším důvodem k výběru tohoto programu je možnost vytváření plug-inů v jazyce ManuScript (tento jazyk byl vytvořen přímo pro tyto účely pro program Sibelius). Metodika problematiky byla z velké části založena na analýze již napsaných plug-inů, které jsou pro program Sibelius dostupné, a studiu oficiální literatury pro tento skriptovací jazyk.

V práci jsou charakterizovány konkrétní nedostatky či neefektivní postupy specializovaných scénářů v tomto softwaru. Následně je navrženo jejich řešení či zefektivnění právě pomocí vytvoření plug-inů, jejichž funkcionalita a jednotlivé prvky jsou v práci vysvětleny.

3 TEORETICKÁ VÝCHODISKA

3.1 APLIKAČNÍ SOFTWARE

Software obecně je hromadné označení pro počítačové programy, které v počítači provádí nějakou činnost; je tedy protikladem hardwaru (fyzickým součástem počítače).

Software lze podle jeho funkčnosti rozdělit do několika skupin:

1. systémový software
   - firmware
   - operační systém
2. aplikační software (zkráceně jen aplikace)

Pojem aplikační software zahrnuje takové programové vybavení počítače, které zajišťuje přímou práci uživatele s počítačem. Cílem aplikací je zpracovat a řešit konkrétní úlohy požadované uživatelem. K interakci mezi uživatelem a aplikací je využíváno grafické nebo textové prostředí.

I aplikační software lze dále dělit dle funkcí, které vykonává, např.: kancelářské aplikace či kancelářské balíky, správci souborů, vývojové nástroje, grafické programy, databázové systémy, internetové prohlížeče, hry a zábavní software a další specializované druhy aplikací. (1, 2)

Jedním ze specializovaných typů aplikačních softwarů jsou tzv. notační programy. (3)

3.2 NOTAČNÍ PROGRAMY

Notační program (v angličtině *music notation software* nebo *scorewriter*) je software sloužící k vytváření, upravování a tisku notových partitur.

Většina dnešních notačních programů má mnoho dalších funkcí – umožňuje zadávání not pomocí MIDI kontrolerů, přehrávání zapsaných partitur (pomocí MIDI nebo virtuálních nástrojů), export partitur do jiných formátů (MIDI, audio soubor, pdf partitury aj.), sdílení partitur s ostatními uživateli apod. (3–5)

3.2.1 Finale

Obrázek 1: logo Finale (10)

Program Finale byl vyvinut Philem Farrandem v roce 1988 pro společnost Coda Music Software. Ta byla později prodána společnosti Net4Music, která dnes nese název MakeMusic. Finale je dostupné pro operační systémy Microsoft Windows a macOS a dodnes zůstává jedním z nejpoužívanějších notačních softwarů na trhu. (10, 11)

Tomuto programu je nejvíce vytýkána strmá křivka učení a nepřívětivé uživatelské prostředí. Naopak mezi výhody patří velmi bohatá funkcionalita, a proto je používán především profesionály a velkými vydavatelskými společnostmi. (12)

V plné verzi je program Finale prodáván za 600 $; jsou k dispozici i zlevněné studentské a pedagogické licence. Společnost MakeMusic nabízí i další levnější varianty programu s omezenou funkcionalitou – PrintMusic a Finale Notepad –, které jsou dostupné pouze pro operační systémy Microsoft Windows. (10, 13)

3.2.2 Dorico

Obrázek 2: logo Dorico (14)

Dorico je software od společnosti Steinberg dostupný pro operační systémy Microsoft Windows a macOS, který byl vydán v říjnu roku 2016. Byl vyvinut týmem z větší části tvořeným vývojáři programu Sibelius, kteří byli propuštěni v roce 2012. (14, 15)

Program je pojmenován po italském tiskaři z 16. století Valeriu Doricovi, který jako první tisknul duchovní hudbu např. Giovanniho Pierluigiho da Palestriny. (16)

Mezi hlavní výhody Dorica patří uživatelská přívětivost a velmi rozšířená funkcionalita. Zároveň tento software řeší automaticky většinu notačně-grafických náležitostí, které by partitura měla obsahovat, a šetří tak práci a čas uživatele. Další výhodou oproti ostatním notačním softwarům na trhu je možnost moderní notace používané převážně od 20. století (mikrotonalita, zápis polymetrické hudby apod.). (17–19)

Plná verze Dorico Pro je prodávána za 559 €, společnost Steinberg nabízí i licence pro pedagogy a studenty za sníženou cenu. Dále je dostupná verze Dorico Elements v ceně 99,99 €, která má oproti verzi Pro omezenou funkcionalitu. (20)

3.2.3 MuseScore

Obrázek 3: logo MuseScore (21)

MuseScore je notační software zdarma pro operační systémy Microsoft Windows, macOS a Linux. Jde o open-source program pod GNU General Public License. Vznikl odtržením od sekvenceru MusE, který původně sám umožňoval zápis not. Byl vyvinut Wernerem Schweerem, který se v roce 2002 rozhodl, že notační funkcionalitu tohoto programu přesune do nového softwaru MuseScore. (21)

V lednu 2021 byla vydána nejnovější verze programu 3.6, která přináší změny v oblasti notografie, nového písma apod., které mají uživateli ulehčit vytváření profesionálně vypadajících not. (22)

Mezi největší výhody programu patří jeho dostupnost, přívětivé uživatelské prostředí a bohatá funkcionalita. Oproti ostatním zmíněným notačním softwarům také dostupnost ve více jazycích (i v češtině). (21, 23)

3.2.4 Sibelius

Obrázek 4: logo Sibelius (6)

Sibelius je celosvětově nejprodávanější notační program dostupný pro operační systémy Microsoft Windows a macOS. Byl vyvinut firmou Sibelius Software Limited, která je v současné době součástí Avid Technology. (6, 7)

3.2.4.1 Historie

Program Sibelius byl vyvinut ve Velké Británii dvěma bratry, dvojčaty Jonathanem a Benem Finnovými, původně pro počítače Acorn Archimedes. S vývojem začali na konci 80. let a první verze programu vyšla v dubnu roku 1993.

V roce 1998 vyšla verze Sibelius 1.0 pro systém Microsoft Windows a o pár měsíců později verze Sibelius 1.2 pro Mac OS. Z původního assembly language byl celý program přepsán v jazyku C++. Vydání programu pro tyto operační systémy vedlo k jeho celosvětovému rozšíření.

V říjnu roku 2006 byla původní firma Sibelius Software Ltd koupena americkou firmou Avid Technology. V červenci 2012 společnost Avid uzavřela kancelář Sibelia v Londýně a propustila původní vývojářský tým. Po protestech uživatelů vytvořila nový tým vývojářů v Kanadě a na Ukrajině. (6, 24)

3.2.4.2 Verze programu Sibelius

Verze programu Sibelius jsou od roku 2018 značeny ve formátu Sibelius *rok.měsíc*, např. v únoru roku 2021 byla vydána nejnovější verze programu – Sibelius 2021.2. (25, 26)

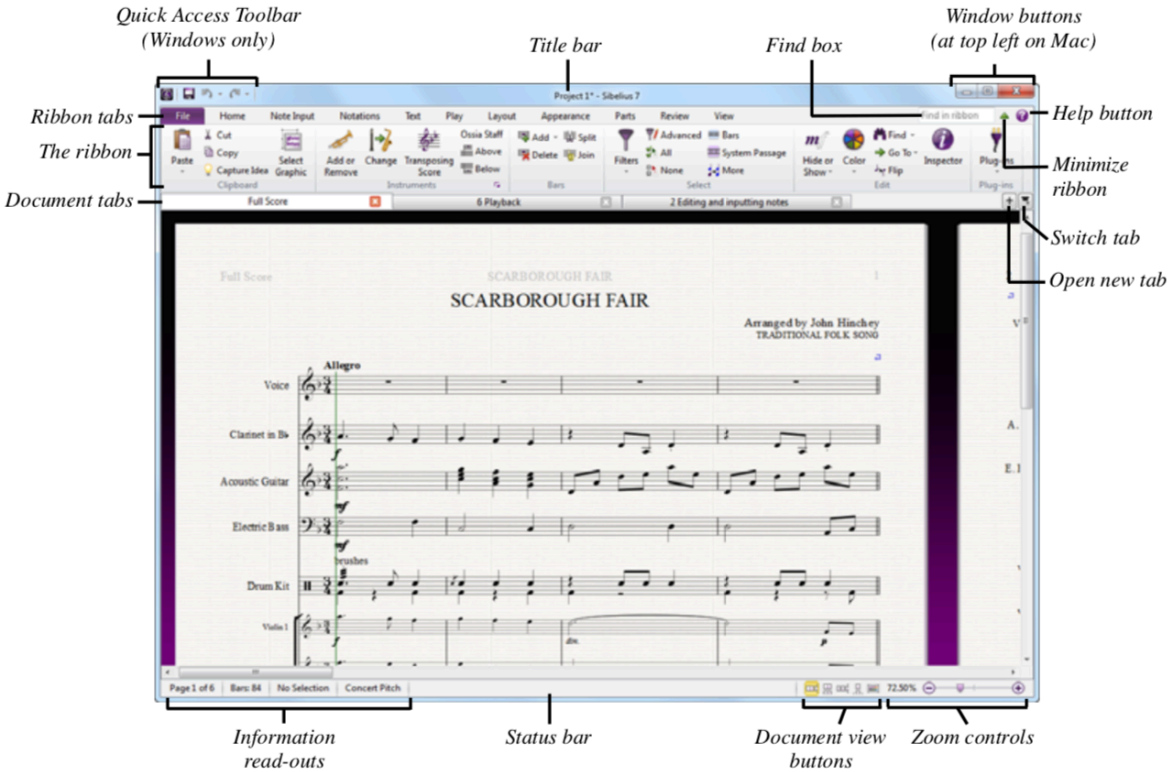
Navíc od roku 2018 jsou dostupné tři varianty programu:

- Sibelius | First – dostupný zdarma, jen základní funkcionalita, maximálně 4 nástroje
- Sibelius (dříve pod názvem Sibelius First) – levnější než plná verze, omezená funkcionalita, maximálně 16 nástrojů
- Sibelius | Ultimate (dříve pod názvem Sibelius) – plná verze programu, prodávána za 599 $ (6, 27)

3.2.4.3 Uživatelské rozhraní

Uživatelské prostředí programu Sibelius prošlo největší změnou při uvedení verze 7 (rok 2011); jako novinka se objevil tzv. pás karet (*the ribbon*) – ovládací prvek patentovaný společností Microsoft, známý především z kancelářského balíku Microsoft Office. (28–30)

Takto vypadá typické okno programu Sibelius (s přidanými popisky):



Obrázek 5: prostředí programu Sibelius (9)

Veškerá funkcionalita programu je rozdělena podle kategorií do jednotlivých karet (*ribbon tabs*): File, Home, Note Input, Notation, Text, Play, Layout, Appearance, Parts (dostupné jen ve verzi Sibelius | Ultimate), Review a View.

Karta File se chová jinak než ostatní zmíněné; po kliknutí na ni je schována partitura a objeví se tzv. pozadí (*background*), speciální zobrazení s další funkcionalitou pro práci s partiturou – vytváření nové partitury, ukládání, otevírání, exportování partitury do jiných formátů, importování partitury z jiných formátů, tisk partitury apod.

Dalších deset karet obsahuje funkcionalitu pro práci v partituře. Jsou seřazeny podle toho, jak vývojáři předpokládali, že skladatel, muzikant či notograf bude postupovat při práci na projektu – bude postupně využívat funkce karet zleva doprava, tj. od karty Home po kartu View:

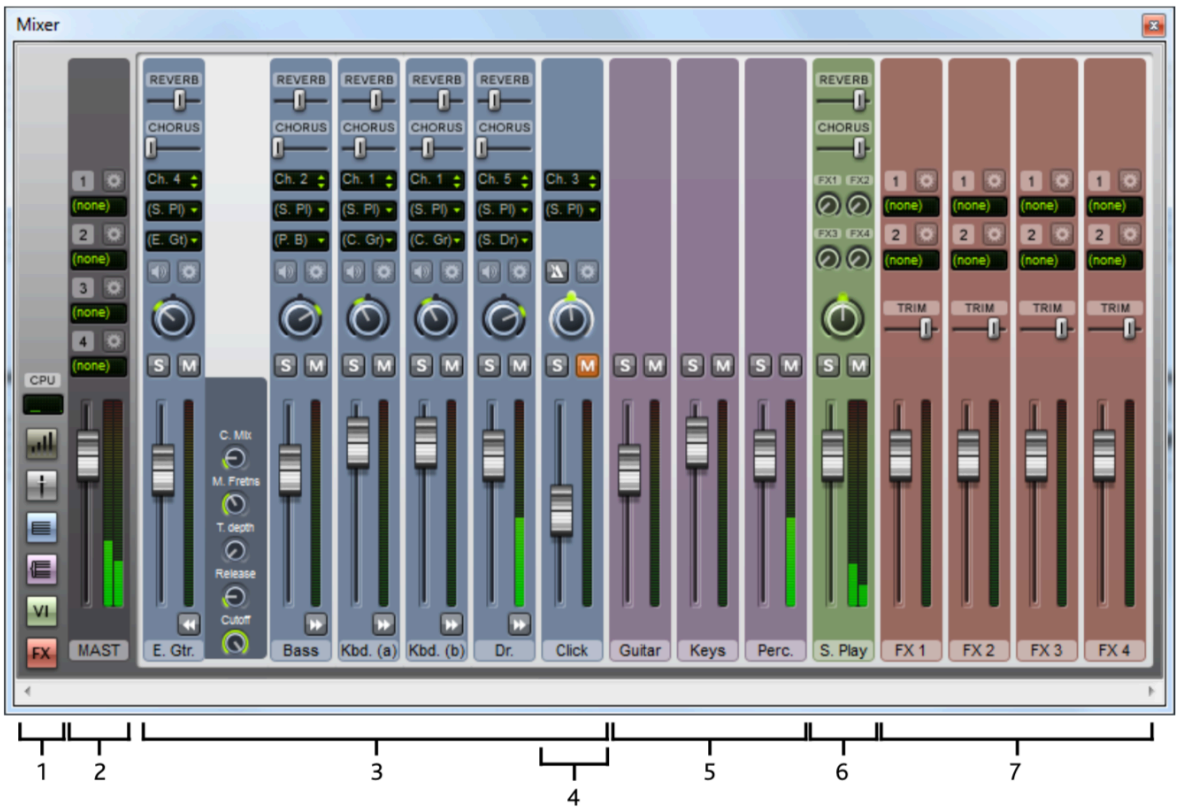
- Home – základní nastavení partitury, přidávání/odstraňování nástrojů a taktů, operace se schránkou (kopírování, vkládání apod.), filtry atd.
- Note Input – vkládání not, operace pro úpravu not, nepravidelné útvary not, enharmonické záměny, transpozice atd.
- Notation – základní prvky partitury (mimo not), tedy klíče, taktové čáry, předznamenání, taktová označení, dále symboly, linkové znaky, typy notových hlaviček atd.
- Text – nastavení fontů, velikostí písma, textových stylů, dále zpěvní text, akordové značky, orientační značky, nastavení číslování stránek a taktů atd.
- Playback – konfigurace přehrávání, mixer (charakterizován v kapitole 3.2.4.4), ovládací prvky přehrávání, nastavení interpretace atd.
- Layout – nastavení vzhledu dokumentu jako například velikost stránek, okraje, rozmístění osnov a systémů, nastavení magnetic layout (blíže vysvětleno v kapitole 3.2.4.4) atd.
- Appearance – nastavení ovlivňující vzhled partitury, např. takzvaný house style, mezery okolo not (*note spacing*), formát názvů nástrojů v partituře atd.
- Parts – nastavení vztahující se k partům nástrojů
- Review – komentáře, práce s verzemi dané partitury atd.
- View – nastavení zobrazení otevřené partitury na obrazovce, otevřených oken programu Sibelius atd.

Každá karta má svoji funkcionalitu navíc roztříděnou do tzv. skupin (*groups*) spojující příbuzné/podobné příkazy. (9)

3.2.4.4 Vybrané funkcionality programu Sibelius

Mixer

Mixer v programu Sibelius umožňuje upravovat nastavení přehrávání jednotlivých nástrojů v partituře. Zobrazí se stisknutím klávesy M (pouze na systému macOS) nebo obecně vybráním Play > Setup > Mixer. Vypadá takto:



Obrázek 6: mixer (9)

V obrázku jsou vyznačené jednotlivé sekce mixeru:

1. ukazatel využití procesoru, zobrazení/skrytí rozšířených zobrazení (do výšky anebo dalších sekcí)
2. celková hlasitost všech virtuálních nástrojů a efektů (tzv. *master volume*)
3. jednotlivé nástroje partitury a jejich nastavení (hlasitost, solo/mute, panorama, MIDI kanál, chorus, reverb atd.)
4. nastavení metronomického kliku při přehrávání (tzv. *click track*)
5. sekce skupin nástrojů – jejich hromadné ovládání
6. virtuální nástroje
7. efekty (9)

MIDI a jeho exportování v programu Sibelius

Zkratka MIDI značí Musical Instrument Digital Interface. Jde o standard pro propojování elektronických hudebních nástrojů mezi sebou a mezi dalšími zařízeními (například počítač). (9, 31)

Pojmem MIDI se zároveň označuje i audio soubor ve standardním formátu SMF (Standard MIDI File), který slouží pro ukládání hudebních dat a jejich přenosu mezi různými zařízeními nebo programy. Téměř veškerý hudební software umožňuje ukládání (exportování) a otevírání (importování) MIDI souborů. Tyto soubory mají příponu „.mid“ a oproti digitálním audio formátům (např. WAV, AIFF, MP3 atd.) neukládají zvuk samotný, ale jsou pouze seznamem událostí popisující kroky, které musí zvuková karta (nebo jiné zařízení) učinit, aby vygenerovala konkrétní zvuky. Díky tomu jsou MIDI soubory několikanásobně menší než digitální zvukové soubory; navíc umožňují zmíněné události všemožně upravovat a tím výsledný zvuk měnit. (31)

Tímto způsobem je možné partituru v programu Sibelius exportovat do jiných audio softwarů. MIDI soubory jsou ale určeny především pro přehrávání zvuků, ne pro sázení not a jejich tisk či export do grafické podoby (neobsahují všechny informace, které lze najít v běžné partituře – artikulační znaménka, legatové obloučky apod.; zároveň nerozlišují enharmonické záměny not – např. rozdíl mezi zápisem noty fis a ges je ignorován). (9)

V programu se partitura do MIDI souboru exportuje pomocí File > Export > MIDI. Zobrazí se stránka s nastavením exportu. Po stisknutí tlačítka „Export“ je zobrazeno dialogové okno, kde uživatel zvolí název a umístění vyexportovaného MIDI souboru partitury. Vzniklý soubor obsahuje i nastavení přehrávání (jako je rubato, espressivo, rytmické frázování apod.) a nastavení mixeru (není ale vyexportován click track, i pokud je v mixeru nastaven a zapnut). (9, 32)

Exportování audio souborů

Podobným způsobem lze vyexportovat partituru do digitálního audio souboru (ve formátu WAV, AIFF či MP3) pomocí File > Export > Audio. Uživatel musí nastavit, jakou konfiguraci přehrávání chce pro export použít – musí být vybrána taková, která používá alespoň jeden virtuální nástroj (Sibelius | Ultimate je dodáván s přibližně 36 GB rozsáhlou zvukovou knihovnou, ale lze používat i virtuální nástroje od jiných firem). (9, 27)

Dále lze nastavit umístění vyexportovaného souboru a jeho název, přenosová rychlost (*bit rate*; u formátu MP3) či bitová hloubka (*bit depth*; u formátů WAV a AIFF), vzorkovací frekvence (*sample rate*) a zda se má partitura vyexportovat celá či až od nějakého místa.

Export se potvrdí tlačítkem „Export“. Stejně jako MIDI soubor bude výsledný audio soubor odpovídat konfiguraci přehrávání v mixeru (avšak i včetně click tracku). (9)

Symboly

Všechny standardní hudební symboly programu Sibelius jsou dostupné na kartě Notation > Symbols v galerii symbolů (*symbol gallery*). Symboly lze na rozdíl od jiných objektů umístit kamkoliv – je například možné vložit do partitury symbol posuvky jinam, než je u posuvek zvykem (tj. před notu). Takto přidaný symbol se nebude chovat jako běžná posuvka – nebude se měnit jeho pozice, pokud bude změněna výška noty, neovlivní přehrávanou výšku tónu apod. (9)

Umístění objektů

Symboly i všechny ostatní objekty v partituře (noty, linkové znaky, text atd.) jsou k partituře přichyceny horizontálně (v ose x) i vertikálně (v ose y), takže zůstávají na správném místě, i když se rozložení partitury změní.

V horizontální ose jsou přichyceny k rytmické pozici v taktu (pokud je nota či pomlka na konkrétní rytmické pozici posunuta doleva či doprava, všechny objekty přichycené k této pozici se posunou společně s notou/pomlkou).

Ve vertikální ose je většina objektů přichycena k notové osnově (k její prostřední lince). Objekty, které takto náleží nějaké osnově, jsou nazývány objekty osnovy (*staff objects*). Některé objekty jsou přichyceny ke všem osnovám daného systému, ne ke konkrétní osnově. Takové objekty se nazývají objekty systému (*system objects*), jedná se například o název partitury, tempová označení, prima- a secondavolty apod.

Každý objekt má v programu Sibelius nastaveny posuny v obou osách, aby byl daný objekt umístěn na správné místo v partituře. Tyto hodnoty může uživatel měnit posunem objektů pomocí myši nebo jejich přepsáním. (8, 9)

Magnetic Layout

Magnetic layout je funkce programu Sibelius pro automatické řešení kolizí objektů v partituře. Uživatel tak nemusí ručně měnit pozice objektů jako je dynamika, symboly, texty, akordické značky, čísla taktů apod. Objekty, které spolu souvisí, jsou navíc zarovnávány společně (například zpěvní text je automaticky zarovnán na stejnou výšku v rámci každé osnovy, stejně tak dynamická znaménka, akordické značky, tempová označení, pedalizace apod.).

Uživatel může deaktivovat funkci magnetic layout pro celou partituru (na kartě Layout > Magnetic Layout > Magnetic Layout), případně jen pro konkrétní objekt v partituře (vybere objekt a nastaví na kartě Layout > Magnetic Layout combo box Object na hodnotu „Off“; druhou možností je kliknout pravým tlačítkem myši na objekt a zvolit Magnetic Layout > Off).

Zmíněný combo box u každého vybraného objektu ukáže, jaké je jeho nastavení magnetic layout:

- hodnota „Default“ značí, že nastavení funkce magnetic layout je stejné jako u celé partitury (tzn. nebylo explicitně změněno),
- hodnota „Off“ značí explicitně vypnutou funkci magnetic layout pro daný objekt,
- hodnota „On“ značí explicitně zapnutou funkci magnetic layout pro daný objekt. (9)

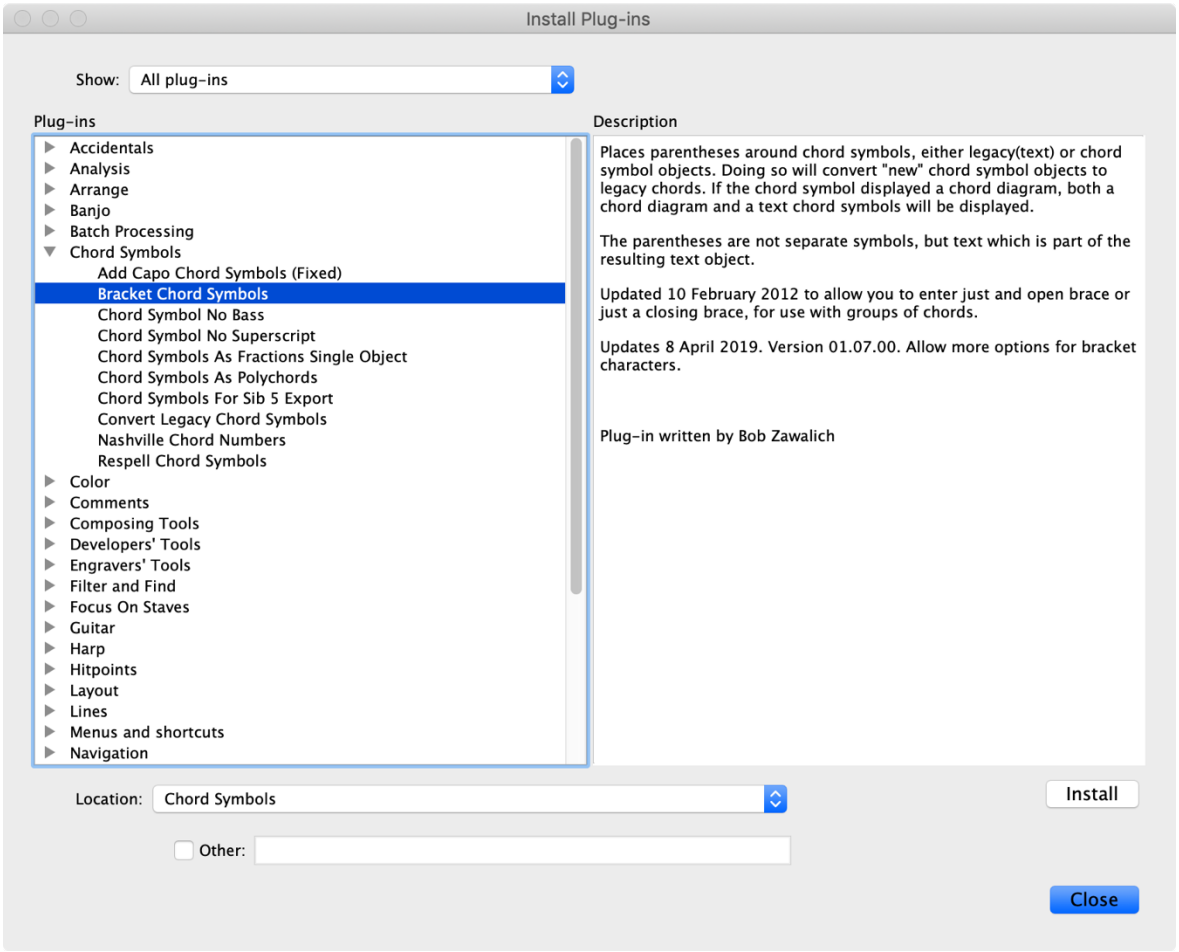
3.3 PLUG-INY V PROGRAMU SIBELIUS

Plug-in neboli zásuvný modul je program, který funguje uvnitř jiného softwaru a rozšiřuje jeho funkcionalitu. V programu Sibelius jsou zásuvné moduly psány ve skriptovacím jazyce ManuScript. (8)

Sibelius je v základu dodáván s cca 150 již nainstalovanými plug-iny a na oficiálních stránkách společnosti Avid je možné zdarma stáhnout dalších přibližně 500 schválených plug-inů. Již dodané plug-iny jsou charakterizovány v manuálu k programu Sibelius (*Sibelius Reference Guide*) a není nutné je instalovat, ostatní musí nainstalovat sám uživatel. (33)

3.3.1 Instalace plug-inů v programu Sibelius

V programu Sibelius je od verze 7 (od roku 2011) dostupný instalátor plug-inů přístupný na File > Plug-ins > Install Plug-ins. Zde je uživatel může jednoduše stahovat a instalovat.



Obrázek 7: dialogové okno Install Plug-ins (zdroj vlastní)

V tomto dialogovém okně lze vybrat plug-in, který chce uživatel nainstalovat. V levé části jsou plug-iny řazeny do kategorií. V pravé části je zobrazen popis vybraného plug-inu. V dolní části si uživatel může zvolit, do jaké podložky na disku se daný plug-in nainstaluje (umístění plug-inů v počítači bude blíže vysvětleno v kapitole 3.3.3). Nainstalování je potvrzeno tlačítkem „Install“. (8, 33, 34)

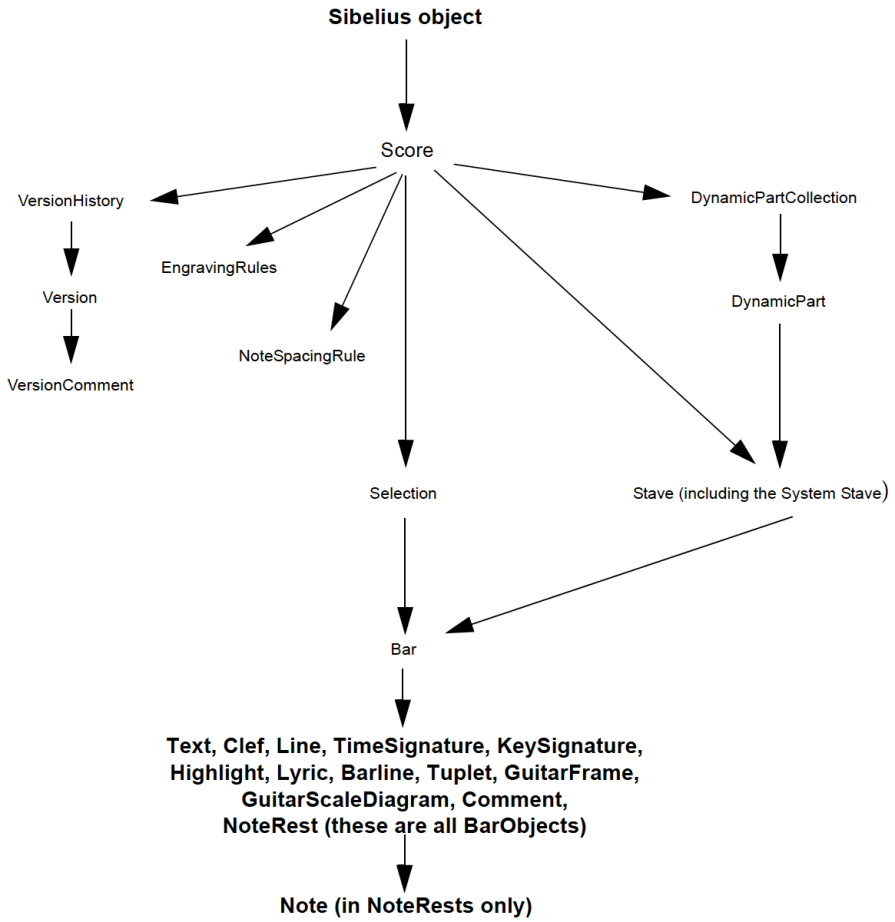
3.3.2 ManuScript

ManuScript je jednoduchý hudební programovací jazyk používaný k psaní plug-inů pro program Sibelius. Je založen na skriptovacím jazyce Simkin, který byl vyvinut Simonem Whitesidem v roce 1999 a později jím a dalšími lidmi rozšířen pro program Sibelius. (8)

3.3.2.1 Reprezentace partitury v jazyce ManuScript

Na partituru (*score*) je v jazyce ManuScript nahlíženo jako na nejvyšší objekt v hierarchické struktuře. Každá partitura obsahuje nezáporné množství osnov (*staff*/*stave*), každá osnova obsahuje alespoň jeden takt (*bar*; všechny osnovy v partituře mají stejný počet taktů), každý takt obsahuje další objekty (*bar objects*) – například text (*text*), klíč (*clef*), taktové označení (*time signature*), pomlku (*note rest*). Objekt NoteRest neznačí jen pomlku jako takovou, ale obecně rytmickou pozici v taktu – tzn. může obsahovat i noty (*note*). (8)

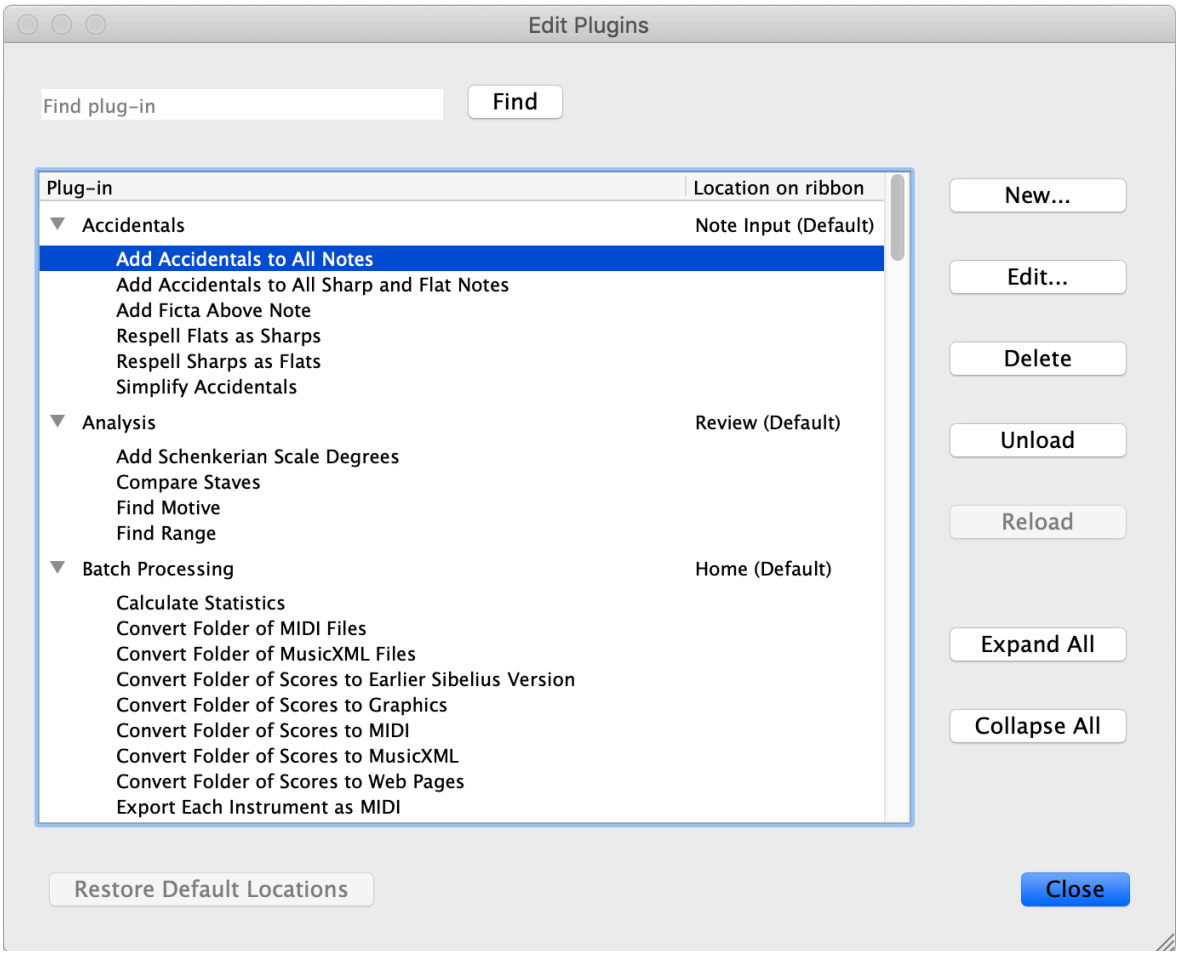
Tato hierarchie objektů je zobrazena na následujícím schématu:



Obrázek 8: hierarchie objektů v jazyce ManuScript (8)

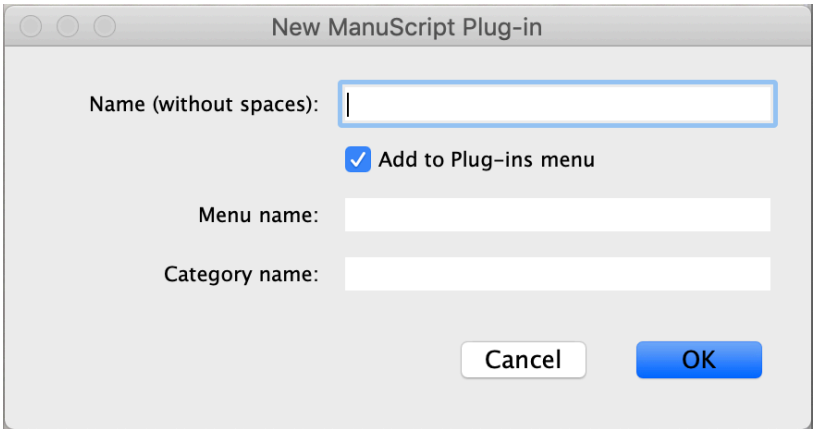
3.3.3 Vytváření plug-inů v programu Sibelius

V otevřené partituře se po kliknutí na kartě File > Plug-ins > Edit Plug-ins zobrazí toto okno:



Obrázek 9: dialogové okno Edit Plugins (zdroj vlastní)

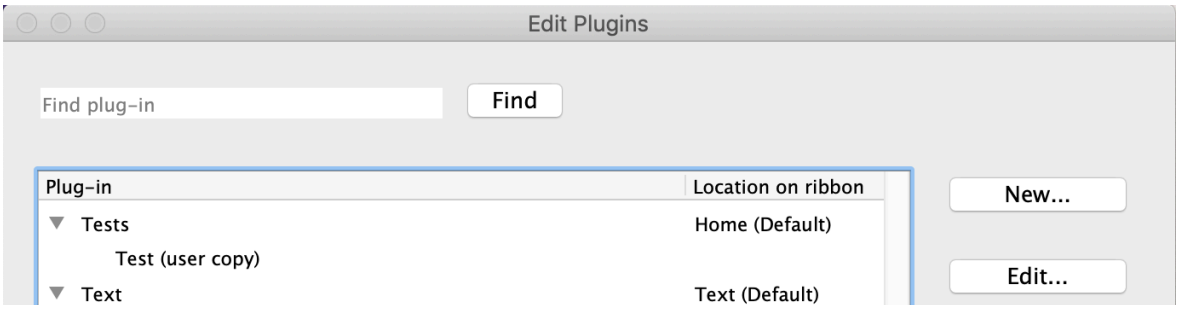
Již existující plug-iny se upravují tlačítkem „Edit…“, nové plug-iny se vytvářejí pomocí tlačítka „New…“:



Obrázek 10: dialogové okno New ManuScript Plug-in (zdroj vlastní)

Uživatel musí vyplnit jméno – *Name (without spaces)* –, pod nímž bude soubor plug-inu uložen (s příponou „.plg“), dále jméno, které se zobrazí v menu plug-inů (*Menu name*), a název kategorie, ve které se bude daný plug-in nacházet (*Category name*).

Po vytvoření plug-inu například s názvem Test, názvem v menu Test a názvem kategorie Tests se objeví daný plug-in v dialogovém okně Edit Plugins:



Obrázek 11: vytvoření plug-inu Test (zdroj vlastní)

Plug-in má za svým názvem „(user copy)“, což značí, že je uložen ve složce s daty uživatelských aplikací, nikoliv v samotné složce programu Sibelius. Standardně se složka nachází:

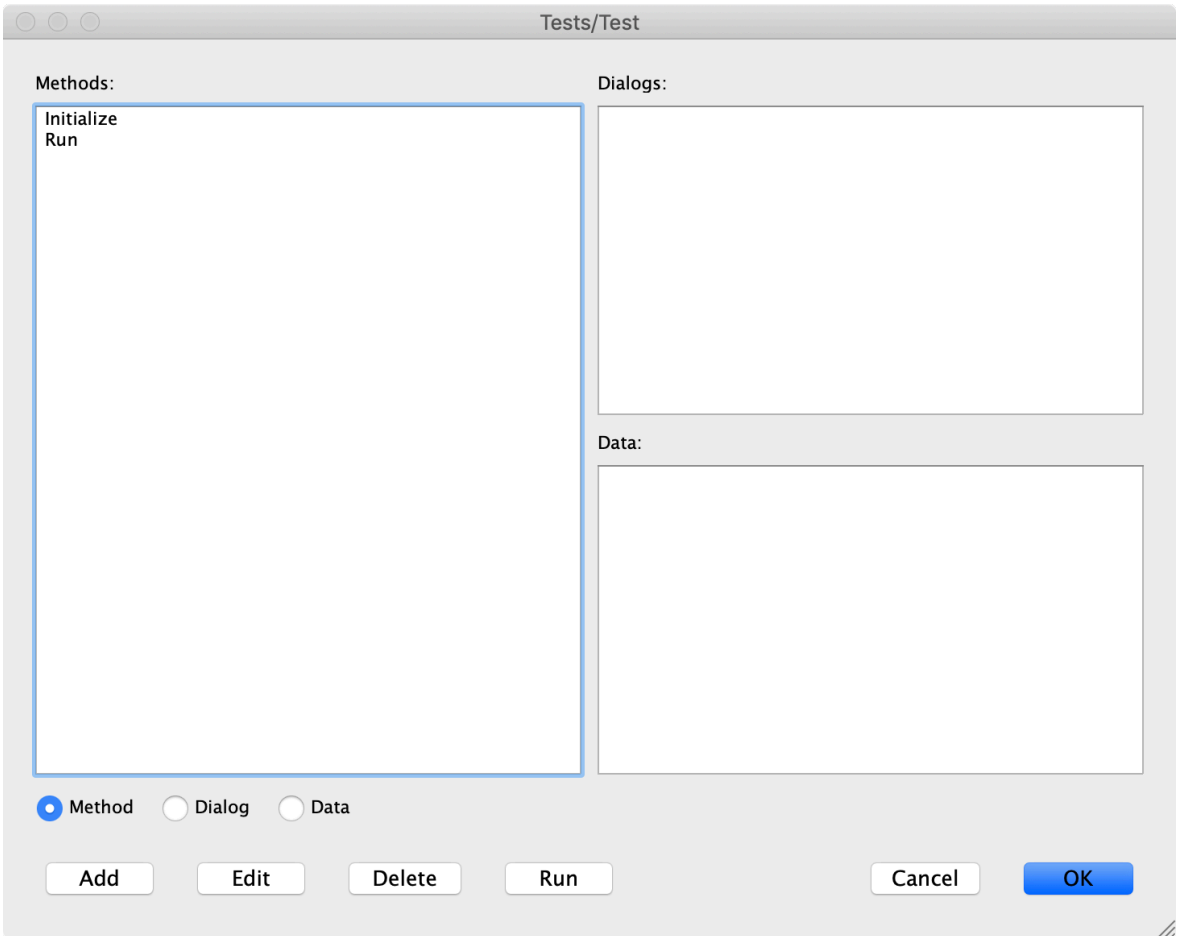
- C:\Users\<uživatelské jméno>\AppData\Roaming\Avid\Sibelius\Plugins (operační systém Microsoft Windows)
- /Users/<uživatelské jméno>/Library/ApplicationSupport/Avid/Sibelius/Plugins (operační systém macOS)

Tam je plug-in uložen v podsložce s názvem shodným se zvoleným názvem kategorie při tvorbě plug-inu.

Sloupec „Location on ribbon“ značí kartu (*ribbon tab*), na které lze v programu Sibelius daný plug-in najít (viz kapitola 3.2.4.3). (8)

3.3.4 Metody, dialogová okna, data

Vytvořený plug-in Test lze upravit pomocí tlačítka „Edit…“ (nebo pomocí dvojitého poklepání na název plug-inu) v dialogovém okně Edit Plugins. Zobrazí se okno:



Obrázek 12: okno úprav plug-inu (zdroj vlastní)

Zobrazuje tři typy informací, které tvoří plug-in v programu Sibelius:

- metody (*methods*)
- dialogová okna (*dialogs*)
- proměnné (*data*) (7, 8)

3.3.4.1 Metody

Plug-in se skládá z metod neboli procedur, funkcí. Jak je vidět na předchozím obrázku, každý plug-in v základu obsahuje dvě metody: Initialize a Run.

Metoda Initialize je volána při každém spuštění programu Sibelius, pro vytvořený plug-in Test vypadá takto:

```
AddToPluginsMenu("Test","Run").
```

Volá metodu AddToPluginsMenu, která přidá položku do menu plug-inů a má dva parametry: *menu text* (název, jak se daná položka bude jmenovat – v tomto případě Test) a *function*

name (jméno metody, která se spustí po výběru dané položky v menu plug-inů – typicky metoda Run). Každý plug-in může volat metodu AddToPluginsMenu právě jednou, tzn. může být přidán do menu plug-inů pouze jednou. (V některých plug-inech se metoda Initialize používá ještě na nastavení defaultních hodnot datových proměnných.)

Metoda Run je volána po spuštění daného plug-inu z menu plug-inů, v případě vytvořeného plug-inu Test na kartě Home > Plug-ins > Tests > Test.

Můžeme vytvářet další metody (vybráním radio buttonu „Method“ a stisknutím tlačítka „Add“). Po výběru metody a po stisknutí tlačítka „Delete“ je vybraná metoda smazána. Tlačítko „Edit“ (nebo dvojité poklepání) otevře dialog ManuScript method, kde lze upravovat kód vybrané metody. A stisknutím tlačítka „Run“ se metoda spustí (je-li stisknuto tlačítko Run bez vybrané metody, spustí se automaticky metoda Run). (7, 8)

3.3.4.2 Dialogy

Zde jsou zobrazena uživatelem vytvořená dialogová okna daného plug-inu. Jazyk ManuScript umožňuje tvořit vlastní dialogová okna pomocí jednoduchého vestavěného editoru.

Aby nějaká metoda mohla zobrazit dialogové okno, musí zavolat metodu

```
Sibelius.ShowDialog(dialogName, Self),
```

kde dialogName značí název dialogu, který se má zobrazit, a Self značí samotný plug-in, který právě běží a který vlastní danou metodu. (8)

Vytváření a úprava dialogů

Dialogy se tvoří, editují a mažou stejně jako metody a datové proměnné. Vytváří se vybráním radio buttonu „Method“ a stisknutím tlačítka „Add“. Upravují nebo mažou se vybráním konkrétního dialogového okna ze seznamu a stisknutím tlačítka „Edit“, respektive „Delete“.

Při vytvoření nebo úpravě se zobrazí dané okno spolu s paletou dostupných kontrolních prvků:

Obrázek 13: paleta kontrolních prvků (8)

Nový kontrolní prvek je vytvořen přetažením daného prvku z této palety do dialogového okna. Vlastnosti daného kontrolního prvku mohou být změněny v jeho dialogu Properties, který se otevře dvojitým poklepáním na prvek (nebo pomocí pravého tlačítka myši a vybráním Properties nebo vybráním prvku levou myší a stisknutím Command + Enter na macOS, respektive Control + Enter na systému Microsoft Windows). Okno s vlastnostmi se liší podle typu kontrolního prvku, ale velká část je pro všechny společná:

- *ID* (řetězec, který jednoznačně identifikuje prvek)
- *Text* (text zobrazující se v kontrolním prvku / u kontrolního prvku)
- *Position* (umístění prvku v dialogovém okně)
- *Size* (velikost kontrolního prvku)
- *Variable storing control's value* (název datové proměnné, která odpovídá kontrolnímu prvku při běhu plug-inu)
- *Method called when clicked* (metoda, která je zavolána, pokud uživatel klikne na prvek)
- *Click closes dialog, returning True/False* (po kliknutí na tento prvek se dialogové okno zavře; zvolené True/False je návratová hodnota funkce Sibelius.ShowDialog, kterou bylo dialogové okno otevřeno)

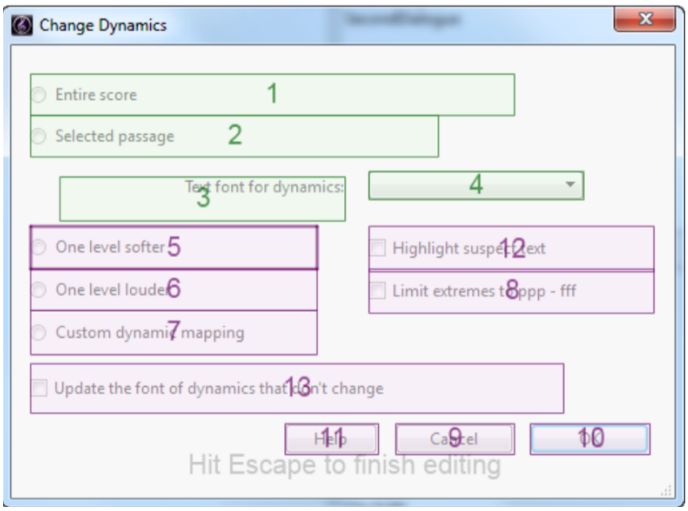
Vlastnosti dialogového okna lze nastavit dvojitým kliknutím na prázdné místo dialogu či kliknutím pravým tlačítkem myši a vybráním Properties. Otevře se Dialog Properties, kde lze nastavit:

- *Name* (název, pod kterým se dialog zobrazí v seznamu dialogů plug-inu)
- *Title* (název, který se zobrazí v horní liště dialogu)
- *Size* (velikost dialogu) a *Position* (pozice, na které se dialogové okno bude otevírat)

Pokud byly provedeny změny, při zavírání dialogového okna je uživatel dotázán, zda chce změny uložit či nikoliv. (7, 8)

Creation order

Pořadí, ve kterém v daném dialogovém okně byly vytvořeny kontrolní prvky (*creation order)*, určuje mimo jiné takzvaný tab order (pořadí, ve kterém je přepínáno mezi jednotlivými prvky klávesou Tab; týká se pouze prvků, na které lze kliknout – radio button, list box, textové pole apod.). Creation order lze změnit kliknutím pravým tlačítkem myši na prázdné místo v dialogu a výběrem Set Creation Order z nabídky. Otevře se okno:



Obrázek 14: dialogové okno s ukázkou creation order (8)

Creation order se nastavuje postupným klikáním na kontrolní prvky. Změněné prvky se zvýrazní zeleně. Po skončení se režim změny creation order ukončí klávesou Esc. (8)

3.3.4.3 Data

V sekci Data jsou vypsány proměnné, jejichž hodnoty jsou uloženy mezi běhy plug-inu. Tyto proměnné mohou být pouze typu String, proto jsou vhodné například pro texty, které se mají vypsat uživateli v dialogovém okně apod. Naopak je nelze použít jako složitější globální proměnné. (8)

4 VLASTNÍ PRÁCE

Notační program Sibelius je velmi silným nástrojem používaným pro vytváření hudebních materiálů a pro další práci s nimi. Ale i přes svou bohatou funkcionalitu nedokáže obsáhnout a řešit všechny požadavky uživatelů. Některé specializované situace a problémy tak lze řešit pouze neefektivními či neoptimálními postupy, které jsou pro uživatele pracné a časově náročné.

Tři takovéto konkrétní situace jsou v této části práce charakterizovány a je navrženo jejich řešení pomocí tvorby plug-inů v jazyce ManuScript.

Autor pracoval v softwaru Sibelius ve verzi Ultimate 2018.6 na operačním systému macOS Mojave.

4.1 EXPORTOVÁNÍ AUDIO SOUBORŮ

Pokud chce uživatel vyexportovat audio (nebo MIDI) soubor otevřené partitury, musí tak učinit přes File > Export > Audio (případně File > Export > MIDI), zvolit příslušné nastavení, kliknout na tlačítko „Export“ a program příslušný soubor vyexportuje. Poměry hlasitostí, zvolené zvuky jednotlivých osnov/nástrojů apod. se vyexportují podle nastavení v mixeru.

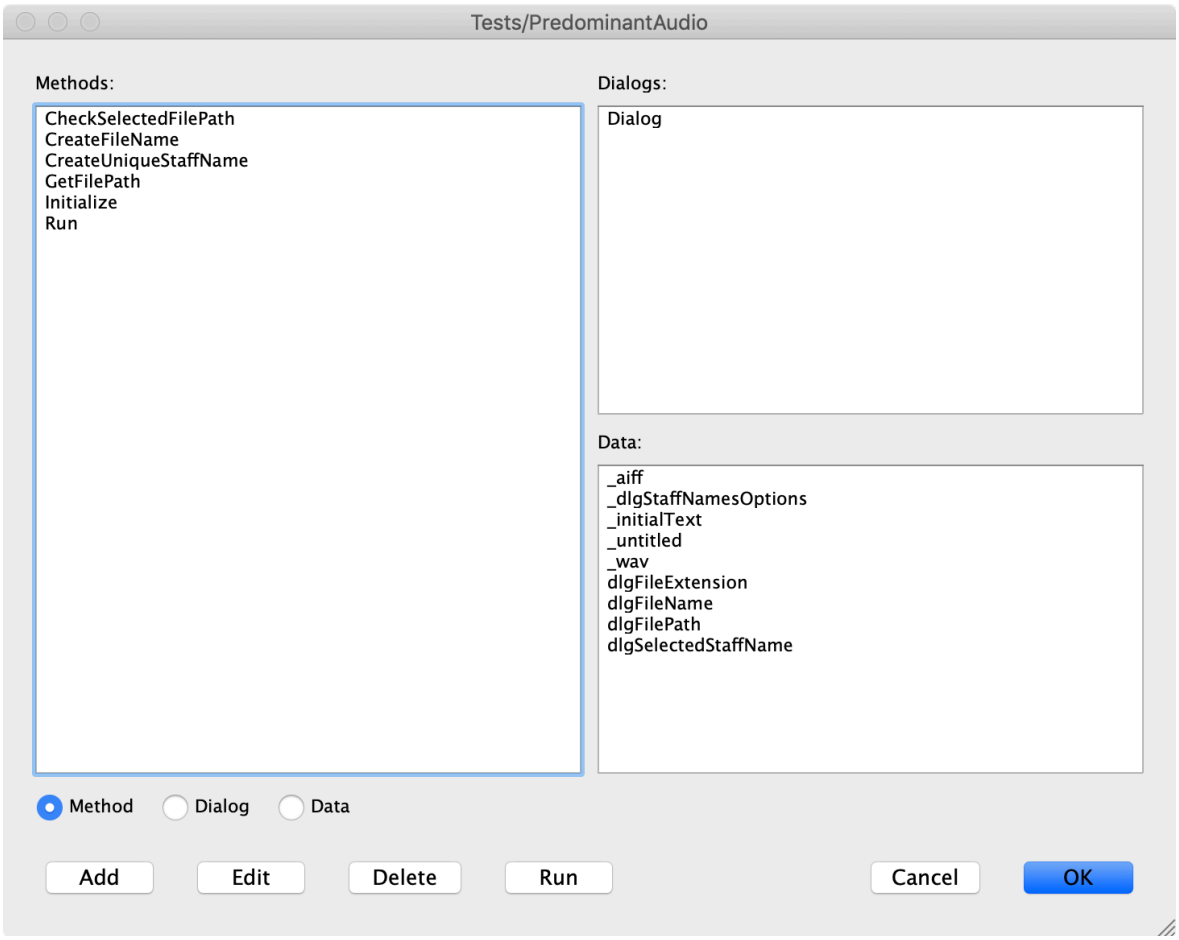
Pro nácvikovou praxi hudebníků je výhodné vyexportovat zvlášť audio soubor každé osnovy s danou osnovou zvýrazněnou a s ostatními zeslabenými. To by znamenalo pro každou osnovu změnit nastavení v mixeru a vyexportovat nový soubor s novým názvem, což i pro středně velké partitury znamená poměrně složitou práci.

Možným řešením by v tomto případě bylo vytvořit plug-in, který by tuto činnost (přenastavení hlasitostí v mixeru a vyexportování audio souboru) zautomatizoval.

4.1.1 Plug-in 1 (Export Individual Staves as Predominant Instrument in Mix)

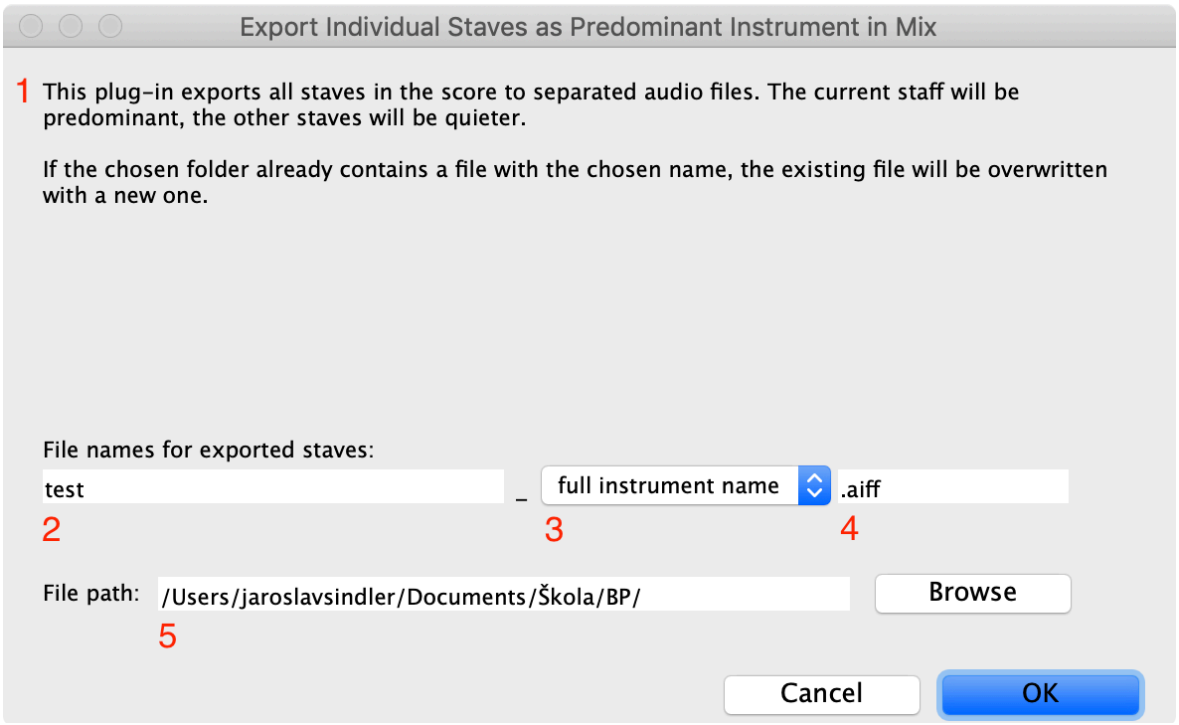
Plug-in „Export Individual Staves as Predominant Instrument in Mix“ vyexportuje pro každou osnovu partitury zvukový soubor s danou osnovou zesílenou a s ostatními osnovami s potlačenou hlasitostí.

4.1.1.1 Okno úprav plug-inu



Obrázek 15: metody, dialogová okna a data plug-inu 1

4.1.1.2 Dialogové okno plug-inu



Obrázek 16: dialogové okno plug-inu 1

1. informace o plug-inu
2. první část názvu vyexportovaných audio souborů
3. druhá část názvu vyexportovaných audio souborů – specifikace konkrétní exportované osnovy
4. přípona vyexportovaných audio souborů
5. umístění vyexportovaných souborů

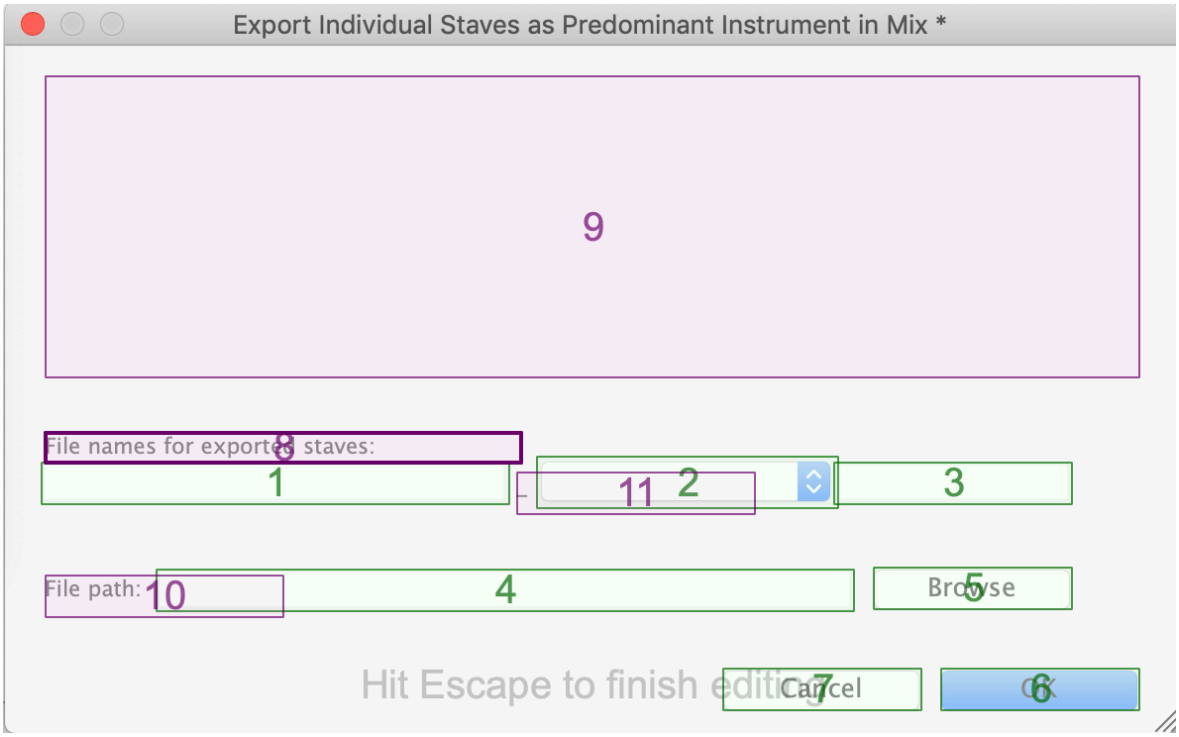
Uživatel zvolí název souboru, který se skládá ze dvou částí – první část je automaticky navržená podle názvu partitury, se kterou se v danou chvíli v programu Sibelius pracuje. Jako druhou část uživatel vybere, chce-li vyexportované zvukové soubory jednotlivých osnov specifikovat jejich zkráceným (*short instrument name)* či plným názvem (*full instrument name*).

Přípona vyexportovaného souboru se doplní automaticky podle operačního systému („.wav“ na systému Microsoft Windows a „.aiff“ na macOS).

Dále uživatel zvolí složku, kam chce audio soubory vyexportovat; automaticky je navržena složka, kde se nachází i právě používaná partitura. Uživatel může cílovou složku změnit pomocí tlačítka „Browse“ – otevře se klasický dialog pro výběr složky.

Creation order

Takto je nastaveno creation order (tab order) dialogového okna:



Obrázek 17: creation order dialogového okna plug-inu 1

4.1.1.3 Data (proměnné)

Proměnné začínající podtržítkem značí konstanty:

- _aiff – obsahuje příponu audio souboru „.aiff“ (standardně pro macOS)
- _dlgStaffNamesOptions – hodnoty, které se objeví na výběr v list boxu (viz Obrázek 16 – 3) pro specifikaci názvu dané osnovy:
  - „short instrument name“ – zkrácený název
  - „full instrument name“ – plný, celý název
- _initialText – obsahuje informace o daném plug-inu, které se zobrazí v jeho dialogovém okně (viz Obrázek 16 – 1)
- _untitled – proměnná, ve které je uložen název nepojmenovaného souboru partitury pro export jeho osnov („untitled“)
- _wav – obsahuje příponu audio souboru „.wav“ (standardně pro systémy Windows)
- dlgFileExtension – hodnota, která se zobrazuje v textovém poli v dialogovém okně (viz Obrázek 16 – 4), obsahující příponu exportovaných souborů

- dlgFileName – hodnota, která se zobrazuje v textovém poli v dialogovém okně (viz Obrázek 16 – 2), obsahující první část názvu vyexportovaného souboru
- dlgFilePath – hodnota, která se zobrazuje v textovém poli v dialogovém okně (viz Obrázek 16 – 5), obsahující cestu, kam se audio soubory vyexportují
- dlgSelectedStaffName – vybraná hodnota z list boxu (viz Obrázek 16 – 3) obsahující druhou část názvu vyexportovaného souboru

4.1.1.4 Metody

Kódy všech metod jsou uvedeny v příloze.

Run

Hlavní metoda Run nejprve zkontroluje, zdali je otevřena nějaká partitura, a pokud ano, zda partitura obsahuje osnovy, nástroje. Dále nastaví příponu vyexportovaných audio souborů podle operačního systému („.aiff“ na macOS, „.wav“ na systému Windows). Nastaví i ostatní proměnné, které se zobrazují v dialogovém oknu a které může uživatel případně měnit. Také uloží (do pole) současné nastavení hodnot hlasitostí jednotlivých osnov v mixeru, aby je po dokončení exportování mohl vrátit do původního stavu. Dále vytvoří pole arrUsedNames, do něhož budou uloženy již použité názvy vyexportovaných souborů (daného běhu plug-inu), aby nedošlo k jejich přepsání, pokud by se některé osnovy jmenovaly stejně.

Při potvrzení plug-inu v dialogovém okně stisknutím tlačítka „OK“ metoda Run vyexportuje pro každou osnovu jeden audio soubor se zvoleným názvem do vybrané složky. Hlasitost dané osnovy je vždy nastavena na 100, ostatní jsou sníženy na 40.

Nakonec vrátí hodnoty hlasitostí jednotlivých osnov do původního stavu.

GetFilePath

Tato metoda je zavolána po stisknutí tlačítka „Browse“ v dialogovém okně plug-inu. Nejprve zavolá vnitřní metodu programu Sibelius SelectFolder(), která otevře dialogové okno pro výběr složky. Pokud je vybraná položka opravdu složkou, změní v proměnné dlgFilePath cestu, do které se budou ukládat vyexportované audio soubory. Nakonec obnoví dialogové okno, aby se změna této proměnné projevila.

CreateFileName

Tato metoda má parametry „staff“ (osnovu, která se bude exportovat) a „arrUsedNames“ (pole názvů již vytvořených souborů). Vytváří a vrací jméno, pod kterým má být audio soubor dané osnovy uložen.

Zkontroluje, jaká varianta druhé části názvu souboru byla v dialogovém okně vybrána (proměnná dlgSelectedStaffName), a podle toho zvolí buď krátké, nebo plné jméno osnovy.

Poté ještě zavolá metodu CreateUniqueStaffName, která zajistí, že zvolené jméno souboru je pro tento běh plug-inu unikátní, a toto unikátní jméno uloží do pole arrUsedNames.

Nakonec zavolá metodu CheckSelectedFilePath (viz dále) a poté sestaví a vrátí metodě Run celý název souboru (plnou cestu).

CreateUniqueStaffName

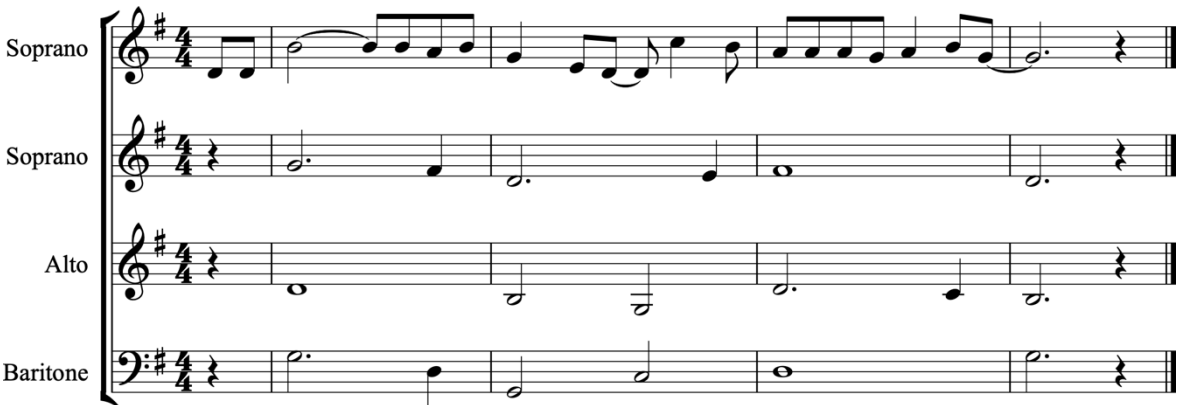
Tato metoda má parametry „originalName“ (jméno osnovy, jejíž unikátnost je potřeba zajistit) a arrUsedNames (pole názvů již vytvořených souborů). Zajistí, že dané jméno osnovy nebylo během tohoto běhu plug-inu použito (v takovém případě by byl starší soubor se stejným názvem přepsán souborem novějším). Takový problém by mohl nastat, pokud by jména dvou (nebo více) osnov byla stejná, např. „Piano“. Tato metoda vrátí jméno druhé osnovy pozměněno na „Piano(2)“, třetí na „Piano(3)“ atd. Pokud je již původní název („originalName“) osnovy unikátní, vrátí ho v nezměněné podobě.

CheckSelectedFilePath

Tato metoda má parametr „path“ (cesta, kam se mají vyexportované audio soubory uložit) a pouze kontroluje, zda je poslední znak zadané cesty lomítko („\“ na systému Windows, „/“ na macOS), případně ho doplní. Nakonec hodnotu path opět vrátí.

4.1.1.5 Příklad použití

Uživatel například vytvoří takovouto partituru, kterou uloží jako soubor pod názvem „Test“:



Obrázek 18: notový zápis pro příklad použití plug-inu 1

Spustí-li plug-in „Export Individual Staves as Predominant Instrument in Mix“, zobrazí se dialogové okno plug-inu (viz kapitola 4.1.1.2), kde uživatel zvolí požadované nastavení – první část názvu ponechá na navržené „Test“ (podle názvu souboru partitury), jako druhou část zvolí „full instrument name“, tedy celé jméno nástrojů (v tomto případě Soprano, Soprano, Alto, Baritone). Umístění nastaví např. na /Users/<uživatelské jméno>/Documents/. Na operačním systému macOS budou mít vyexportované soubory příponu „.aiff“.

Po stisknutí tlačítka „OK“ plug-in pro každou osnovu vyexportuje audio soubor se zvýrazněným daným nástrojem. V tomto případě budou do složky /Users/<uživatelské jméno>/Documents/ vyexportovány 4 soubory:

- test_Soprano.aiff
- test_Soprano(2).aiff
- test_Alto.aiff
- test_Baritone.aiff

(Číslo 2 v závorce u druhého souboru bylo přidáno plug-inem, aby bylo rozlišeno mezi dvěma stejně pojmenovanými osnovami v partituře.)

4.2 EXPORTOVÁNÍ MIDI SOUBORŮ

Exportování MIDI souborů se provádí přes File > Export > MIDI, kde si uživatel zvolí příslušné nastavení a vyexportuje partituru stisknutím tlačítka „Export“.

Problém nastává při exportu partitury do MIDI i s metronomickým klikem, tzv. click trackem – jedná se o velmi běžnou praxi a účelnou pomůcku například při nácviku skladeb muzikanty. Takový export program Sibelius neumožňuje, a i na oficiálním internetovém fóru pro tento software je jako jediné navrhované řešení uváděno vložit do partitury nový perkusní nástroj, vyplnit ho notami simulující metronomický klik a pak tuto partituru vyexportovat jako MIDI soubor.

Dalším praktickým nástrojem při exportování MIDI souborů je tzv. count-in neboli odpočet několika taktů před začátkem samotné partitury. Toho lze v programu Sibelius docílit jen skutečně vytvořením požadovaného počtu prázdných taktů před prvním taktem partitury. Ty je nutné následně také vyplnit notami simulující metronomický klik. Může nastat problém s předtaktím (neúplným taktem na začátku skladby), který je v takovém případě nutné doplnit na délku standardního taktu.

Oba tyto postupy jsou poměrně náročným úkolem, který ale může být snadno řešen pluginem.

4.2.1 Plug-in 2 (Export MIDI with Click)

Plug-in „Export MIDI with Click“ vyexportuje partituru do MIDI souboru společně s metronomickým klikem (tzv. click track), který bude přidán naspod partitury jako samostatný perkusní nástroj.

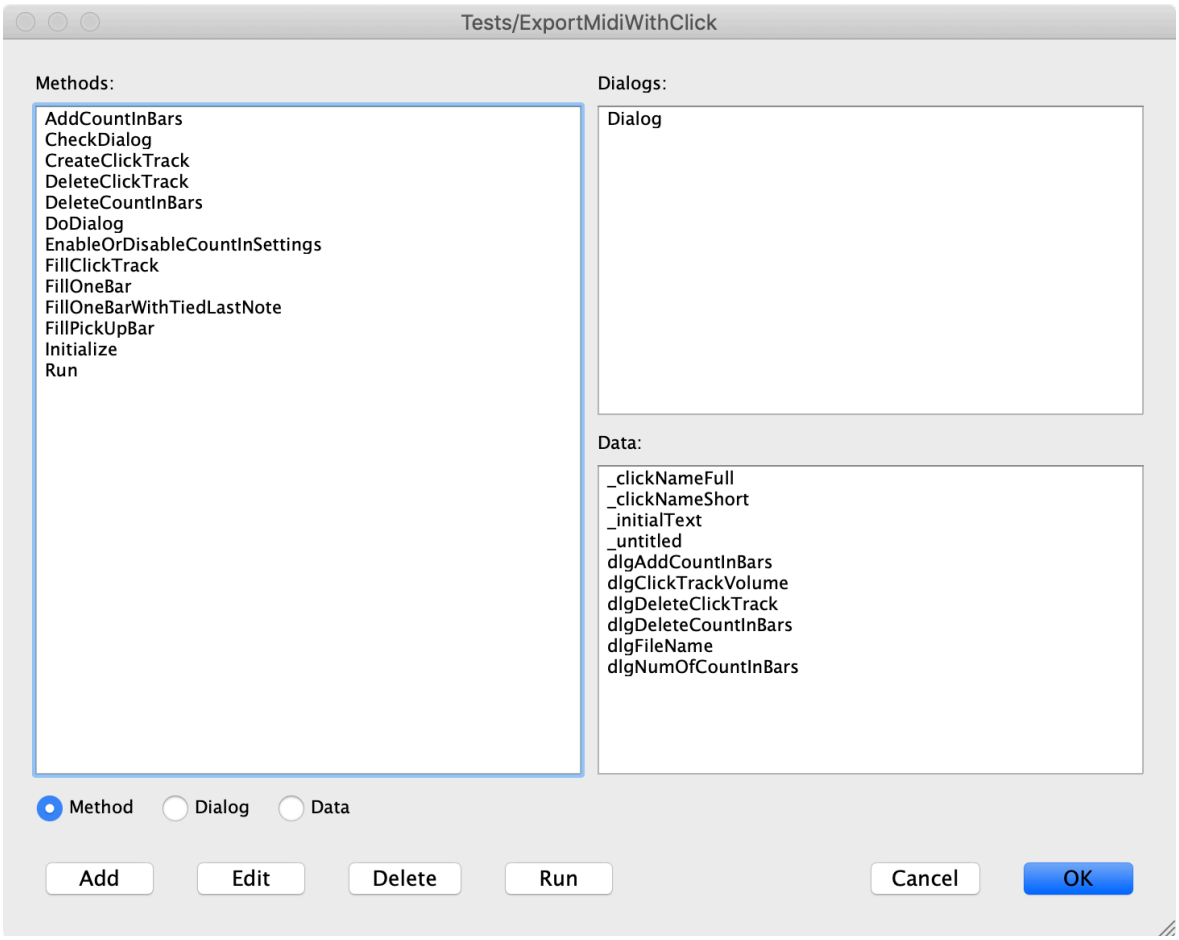
Tento soubor bude vytvořen ve stejné složce, ve které se nachází partitura, a bude mít jméno shodné s názvem souboru partitury, jen místo koncovky „.sib“ bude „.mid“ (označující MIDI soubor). Pokud v této složce již existuje soubor se stejným názvem, bude přesán souborem nově vzniklým.

Uživatel navíc může zvolit vytvoření několika count-in taktů – takty před samotnou partiturou obsahující jen metronomický klik – a jestli tyto přidané takty a samotný click track chce po exportu MIDI souboru v partituře ponechat či nikoliv.

Plug-in si poradí s nepravidelnými takty, měnícími se taktovými označeními i případným předtaktím.

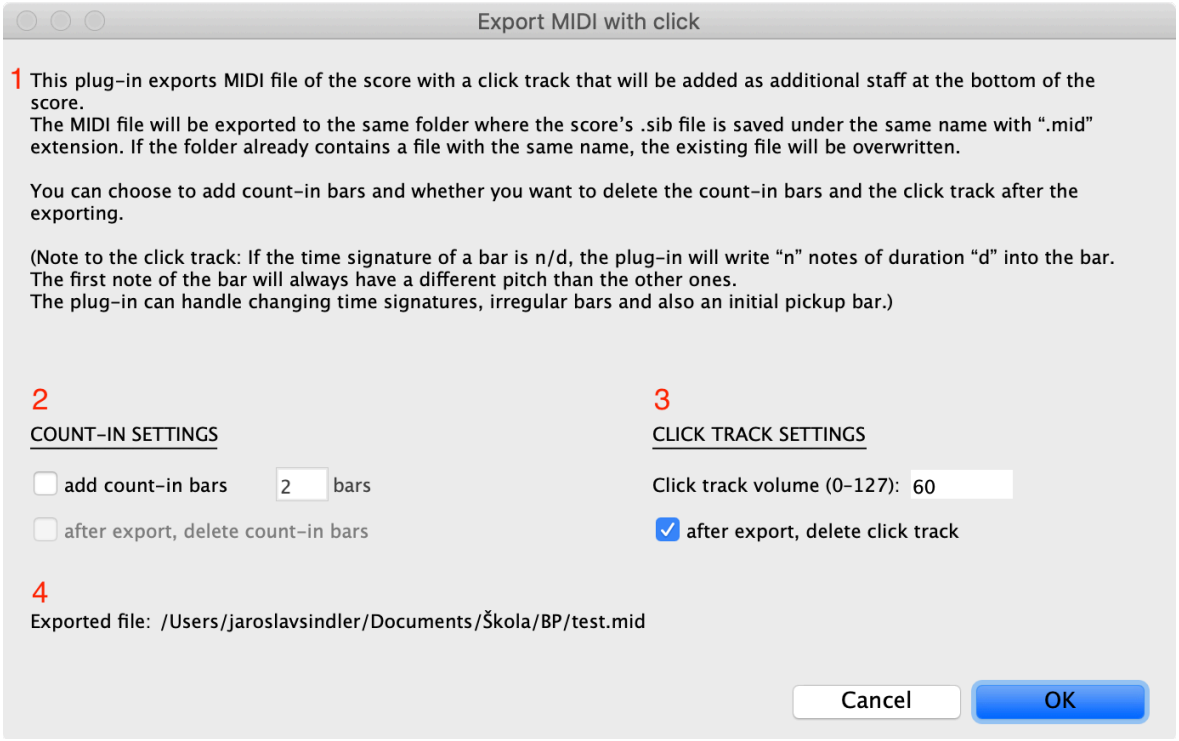
(Poznámka k click tracku: Obecně pokud je taktové označení n/d, plug-in vytvoří click track tak, že daný takt vyplní „n“ notami o délce „d“ – například 6/8 takt bude vyplněn 6 osminovými notami. První nota taktu bude vždy jiné výšky než ostatní, aby byla první doba taktu zvýrazněna.)

4.2.1.1 Okno úprav plug-inu



Obrázek 19: metody, dialogová okna a data plug-inu 2

4.2.1.2 Dialogové okno plug-inu



Obrázek 20: dialogové okno plug-inu 2

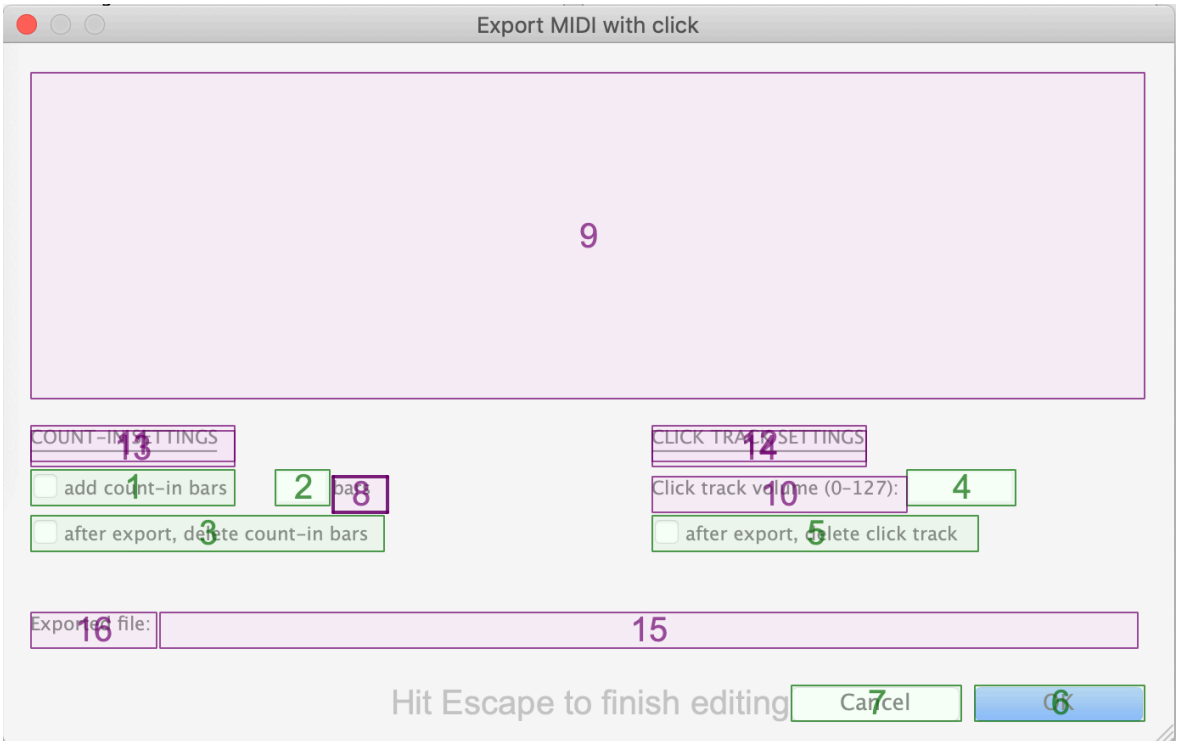
1. informace o plug-inu
2. nastavení count-in taktů (jestli je vytvářet, případně kolik a zda po vyexportování MIDI souboru tyto takty opět smazat nebo ponechat)
3. nastavení click tracku (nastavení jeho hlasitosti a jestli po exportování MIDI souboru click track vymazat nebo ponechat)
4. název a umístění vyexportovaného souboru

Uživatel zvolí, zda chce vyexportovat count-in takty. Pokud ano, může vyplnit jejich počet a zvolit, jestli se mají tyto takty po vyexportování MIDI souboru z partitury opět smazat. Také zvolí hlasitost click tracku a jestli se má po vyexportování MIDI souboru z partitury odstranit.

Tlačítkem „Cancel“ plug-in zruší, tlačítkem „OK“ plug-in spustí a ten vyexportuje MIDI soubor s příslušným nastavením.

Creation order

Takto je nastaveno creation order (tab order) dialogového okna:



Obrázek 21: creation order dialogového okna plug-inu 2

4.2.1.3 Data (proměnné)

Proměnné začínající podtržítkem značí konstanty:

- _clickNameFull – obsahuje plný název vytvořeného nástroje, click tracku („Click track“)
- _clickNameShort – obsahuje zkrácený název vytvořeného nástroje, click tracku („click“)
- _initialText – obsahuje informace o daném plug-inu, které se zobrazí v dialogovém okně (viz Obrázek 20 – 1)
- _untitled – proměnná, ve které je uložen název nepojmenovaného souboru partitury („untitled“)
- dlgAddCountInBars – proměnná true/false spojená s prvním checkboxem v dialogovém okně v sekci nastavení count-in taktů (viz Obrázek 20 – 2) uchovávající, jestli je daný checkbox zaškrtnut či nikoliv – tzn. jestli se mají vytvořit count-in takty nebo ne

- dlgNumOfCountInBars – počet count-in taktů, které se mají vytvořit (standardně jsou uživateli nabídnuty 2 takty, ale tento počet může být změněn)
- dlgDeleteCountInBars – proměnná true/false spojená s druhým checkboxem v dialogovém okně v sekci nastavení count-in taktů (viz Obrázek 20 – 2) uchovávající, jestli je daný checkbox zaškrtnut či nikoliv – tzn. jestli se mají vytvořené count-in takty po exportu MIDI souboru opět smazat nebo ne
- dlgClickTrackVolume – hodnota určující hlasitost click tracku (viz Obrázek 20 – 3), výchozí hodnota je nastavena na 60
- dlgDeleteClickTrack – proměnná true/false spojená s checkboxem v dialogovém okně v sekci nastavení click tracku (viz Obrázek 20 – 3) uchovávající, jestli je daný checkbox zaškrtnut či nikoliv – tzn. jestli se má vytvořený click track po exportu MIDI souboru opět smazat nebo ne
- dlgFileName – proměnná uchovávající jméno souboru, který bude vyexportován (viz Obrázek 20 – 4)

4.2.1.4 Metody

Kódy všech metod jsou uvedeny v příloze.

Run

Hlavní metoda Run nejprve zkontroluje, zdali je otevřena nějaká partitura, a pokud ano, zda partitura obsahuje osnovy, nástroje.

Vytvoří název vyexportovaného souboru, který zobrazí uživateli v dialogovém okně plug-inu (viz Obrázek 20 – 4). Poradí si, i pokud daná partitura ještě nebyla uložena a nemá tedy jméno.

Dále zobrazí dialogové okno plug-inu a pokud uživatel stiskne tlačítko „OK“, provede se zbytek plug-inu s nastavením zvoleným v dialogu.

Následně metoda Run, pokud tato možnost byla zvolena, zavolá metodu AddCountInBars, která vytvoří zvolený počet count-in taktů, a metodu CreateClickTrack, která vytvoří nový nástroj – click track.

Poté zavolá metodu FillClickTrack, která click track vyplní notami simulující metronomický klik, a výslednou partituru vyexportuje jako MIDI soubor pod vytvořeným názvem.

Pokud jsou tyto možnosti zvoleny, následně metoda Run zavolá metody DeleteClickTrack a DeleteCountInBars, které vytvořené count-in takty a click track opět vymažou a tím navrátí partituru do původního stavu.

DoDialog

Metoda DoDialog má parametr „dialog“ (odkaz na dialog, který má vytvořit). Nastaví hodnoty proměnných ovládající kontrolní prvky na výchozí:

- checkbox „add count-in bars“ nastaví na false a deaktivuje odpovídající kolonku na zadávání počtu taktů count-inu,
- checkbox „after export, delete count-in bars“ nastaví na false a deaktivuje ho,
- textové pole ovládající hlasitost click tracku nastaví na 60,
- checkbox „after export, delete click track“ nastaví na true.

Dále zobrazí uživateli dialog s nastavenými hodnotami a po stisknutí tlačítka „OK“ zavolá metodu CheckDialog, která zkontroluje správnost zadaných hodnot. To provádí, dokud nejsou hodnoty správně zadané (po úspěšné kontrole touto metodou vrátí DoDialog hodnotu true) nebo dokud uživatel nestiskne tlačítko „Cancel“ (v takovém případě metoda DoDialog vrátí false).

CheckDialog

Tato metoda ověří správnost zadaných hodnot v dialogovém okně plug-inu. Nejdříve zkontroluje, zda je zadaná hlasitost click tracku a zda je tato hodnota číslo mezi 0 a 127 (včetně). Pokud ne, vrátí false.

Dále zkontroluje, zda zadaný počet count-in taktů je kladné celé číslo. Pokud ne, vrátí false.

Pokud žádná chyba nenastane, vrátí hodnotu true.

EnableOrDisableCountInSettings

Tato metoda aktivuje či deaktivuje kontrolní prvky dialogového okna plug-inu podle akcí uživatele. Pokud uživatel zaškrtne checkbox „add count-in bars“, tato metoda aktivuje kolonku určující počet přidaných taktů a aktivuje také checkbox „after export, delete count-in bars“ a nastaví ho na true.

Naopak pokud uživatel checkbox „add count-in bars“ vyškrtne, metoda provede pravý opak (deaktivuje kolonku určující počet přidaných taktů a zároveň deaktivuje checkbox „after export, delete count-in bars“ a nastaví ho na false).

CreateClickTrack

Tato metoda má parametr „score“ (odkaz na partituru). V této partituře vytvoří nový nástroj s názvem „Click track“, zkráceně „click“ (tyto názvy jsou uloženy v globálních proměnných _clickNameFull a _clickNameShort). Tento nástroj bude perkusní bicí nástroj *woodblock* (v češtině dřívka).

Hlasitost vytvořeného nástroje nastaví dle uživatelem zadané hodnoty v dialogovém okně plug-inu.

Nakonec vrátí odkaz na vytvořený nástroj (click track).

AddCountInBars

Tato metoda má parametry „score“ (odkaz na partituru) a „numOfBars“ (počet taktů). V partituře vytvoří před prvním taktem nové takty (jejich počet je určen právě parametrem „numOfBars“). Pokud je první takt kratší, než by podle taktového označení měl být, jedná se o tzv. předtaktí. V takovém případě před tento takt nevloží standardně dlouhý takt, ale jeho délka bude kratší o délku předtaktí, aby spolu dohromady tvořily jeden standardně dlouhý takt.

FillClickTrack

Tato metoda má parametr „staff“ (odkaz na osnovu). Tuto osnovu vyplní notami, které budou simulovat metronomický klik. Na každý takt zavolá metodu FillOneBar, která tento takt vyplní.

Mohou nastat dvě výjimky – na speciální případ posledního taktu count-inu volá metodu FillOneBarWithTiedLastNote (viz dále). Druhý speciální případ nastane, pokud je daný takt předtaktím. V takovém případě zavolá metodu FillPickUpBar (viz dále), která řeší vyplnění předtaktí.

FillOneBar

Tato metoda má parametry „staff“ (odkaz na osnovu) a „bar“ (odkaz na takt). Vyplní tento takt osnovy metronomickým klikem. Poradí si s nepravidelnými takty, tj. kratšími nebo delšími, než je standardní takt daného taktového označení.

Postupuje od začátku taktu. Na první dobu přidá notu s jinou výškou tónu, aby byla první doba odlišená od ostatních (viz popis plug-inu v kapitole 4.2.1). Zbytek taktu vyplní notami stejné výšky.

Postupně vyplňuje takt, dokud se do něj vejdou noty té délky, která odpovídá jeho taktovému označení. Pokud už se další nota nevejde, zkontroluje, zda nějaké místo v taktu zbylo a případně toto místo vyplní notou odpovídající délky.

FillOneBarWithTiedLastNote

Tato metoda má parametry „staff“ (odkaz na osnovu) a „bar“ (odkaz na takt). Řeší speciální případ, kdy je daný takt posledním taktem count-inu a partitura zároveň obsahuje předtaktí, které není celočíselně dělitelné počtem dob (podle taktového označení). V takovém případě je třeba, aby poslední nota tohoto taktu měla ligaturu a navazovala tak na první notu následujícího taktu (který je předtaktím), aby se v daném místě neozvaly dva kliky metronomu místo jednoho.

FillPickUpBar

Tato metoda má parametry „staff“ (odkaz na osnovu) a „bar“ (odkaz na takt). K tomuto taktu se chová jako k předtaktí a vyplní ho metronomickým klikem. Rozdíl oproti normálnímu taktu je ten, že předtaktí se musí vyplňovat odzadu, nikoliv odpředu.

Metoda tedy postupně přidává noty od konce taktu. Pokud by ale poslední přidaná nota měla být v rámci první doby a uživatel zároveň požaduje vytvoření count-in taktů, bude mít jinou výšku tónu než předchozí noty (musí zvýraznit první dobu – viz popis plug-inu v kapitole 4.2.1).

DeleteClickTrack

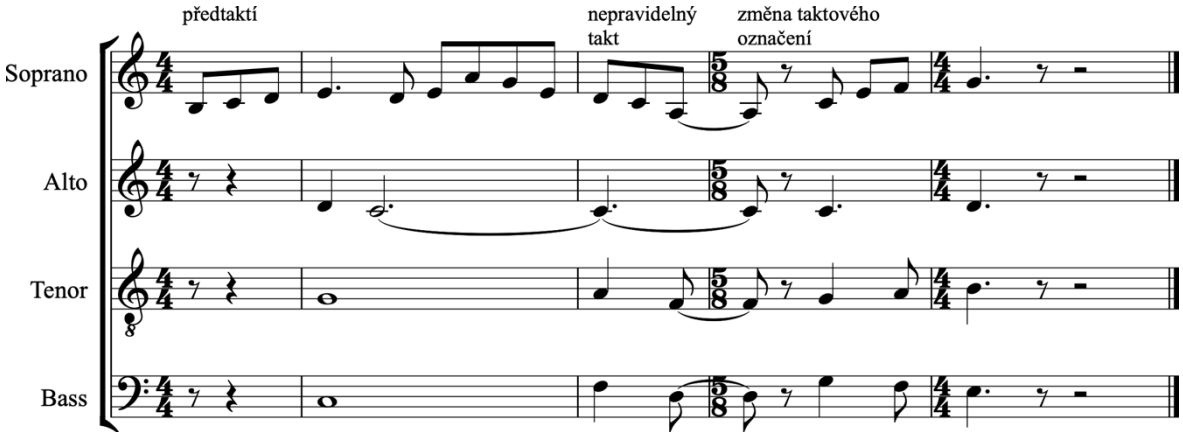
Tato metoda má parametr „staff“ (odkaz na osnovu). Tuto osnovu smaže.

DeleteCountInBars

Tato metoda má parametr „score“ (odkaz na partituru). V této partituře smaže tolik taktů na začátku, kolik uživatel zvolil v dialogovém okně plug-inu. Tím se smažou vytvořené count-in takty.

4.2.1.5 Příklad použití

Uživatel například vytvoří takovouto partituru, kterou uloží jako soubor pod názvem „Test“:



Obrázek 22: notový zápis pro příklad použití plug-inu 2

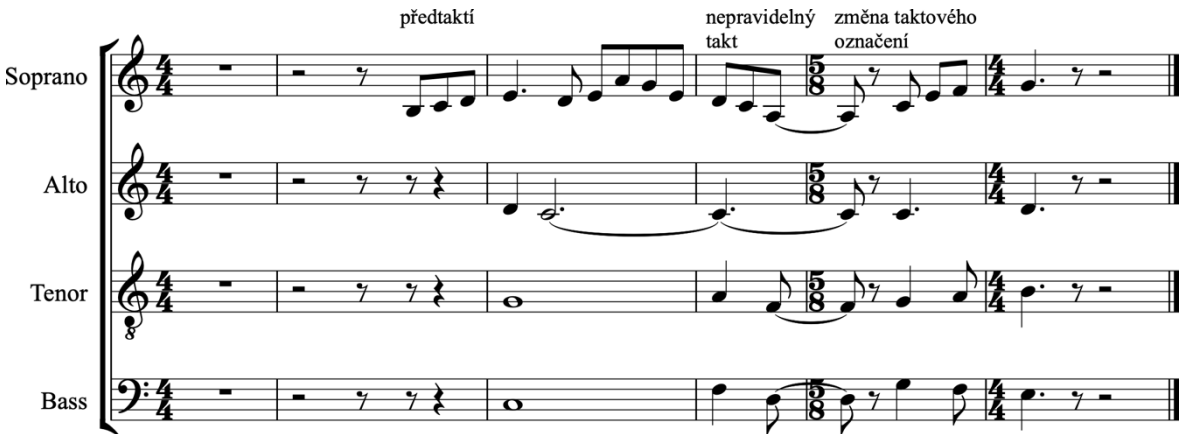
Na obrázku jsou vidět i speciální případy, které plug-in dokáže řešit – předtaktí, nepravidelně dlouhý takt a změna metra (taktového označení).

Pokud spustí plug-in „Export MIDI with Click“, zobrazí se dialogové okno (viz kapitola 4.2.1.2), kde uživatel zvolí požadované nastavení – například že chce přidat 2 count-in takty, které chce po vyexportování MIDI souboru vymazat, a že chce po vyexportování vymazat i vytvořený click track.

Po stisknutí tlačítka „OK“ plug-in vyexportuje MIDI soubor s názvem „test.mid“ do stejné složky, ve které je umístěna partitura. Obsahovala-li složka již soubor „test.mid“, je tímto novým souborem přepsán.

Jelikož uživatel zvolil, že chce count-in takty i click track po vyexportování MIDI souboru smazat, jednotlivé kroky, které plug-in provádí, neuvidí. Zde jsou ale pro přehlednost uvedeny.

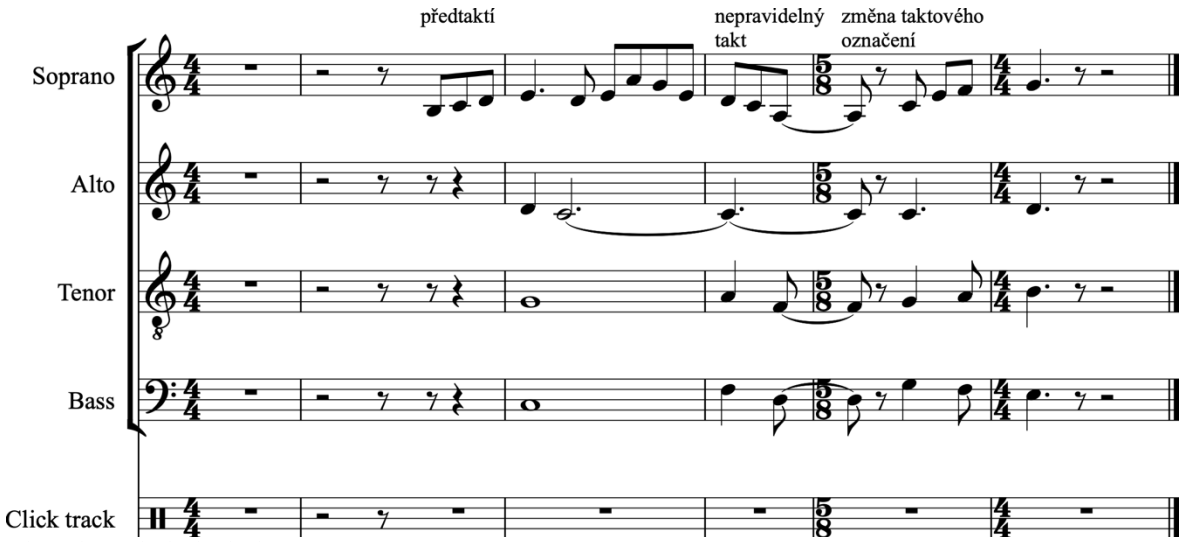
Nejprve plug-in vytvoří 2 count-in takty:



Obrázek 23: první krok plug-inu 2

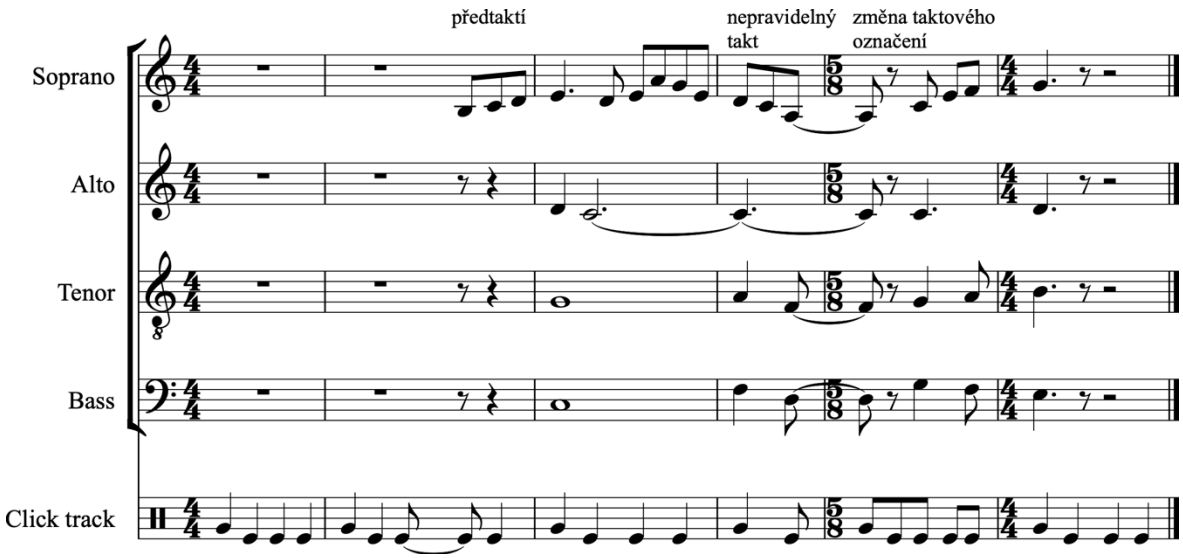
Jak je vidět na obrázku, plug-in si poradí s předtaktím, které doplní do délky standardního taktu podle taktového označení (v tomto případě 4/4) – předtaktí tedy bude začínat již v rámci druhého taktu count-inu.

Poté vytvoří nový perkusní nástroj nazvaný „Click track“:



Obrázek 24: druhý krok plug-inu 2

Ten následně vyplní notami, které budou simulovat metronomický klik:



Obrázek 25: třetí krok plug-inu 2

Jak je na obrázku vidět, vyplnění not funguje i v nepravidelných taktech i při změně taktového označení.

Nakonec (podle uživatelské volby) plug-in vyexportuje daný MIDI soubor i s přidaným click trackem a count-in takty a oboje smaže, čímž vrátí partituru do původní podoby.

4.3 MUSICA FICTA

Při notaci staré hudby (především renesanční a barokní) je někdy zvykem psát posuvky ne před noty (což je standardní způsob moderní notace), ale nad noty. Jedná se o přidané posuvky, tzv. musica ficta, které se v původních rukopisech nezapisovaly, jelikož vycházely z kontextu skladby a z praxe své doby. (35)

V programu Sibelius je dostupný plug-in „Add Ficta Above Note“, který k vybraným notám přidá zvolenou posuvku. Je ale velmi jednoduchý – neumožňuje například přidávat posuvky v závorkách, nastavit jejich umístění nad notou apod. Zároveň při psaní skladby většího rozsahu je nepraktické pro každou potřebnou notu spouštět tento plug-in znovu. Plug-in umí pracovat jen s individuálně vybranými notami, ne výběrem více taktů či osnov najednou. (9)

Komplexnějším řešením této problematiky by bylo vytvořit plug-in, který by ve vybrané pasáži změnil všechny standardně napsané posuvky na symboly nad notou (musica ficta), případně uživateli umožnil další nastavení těchto přidaných symbolů.

4.3.1 Plug-in 3 (Change Accidentals to Ficta)

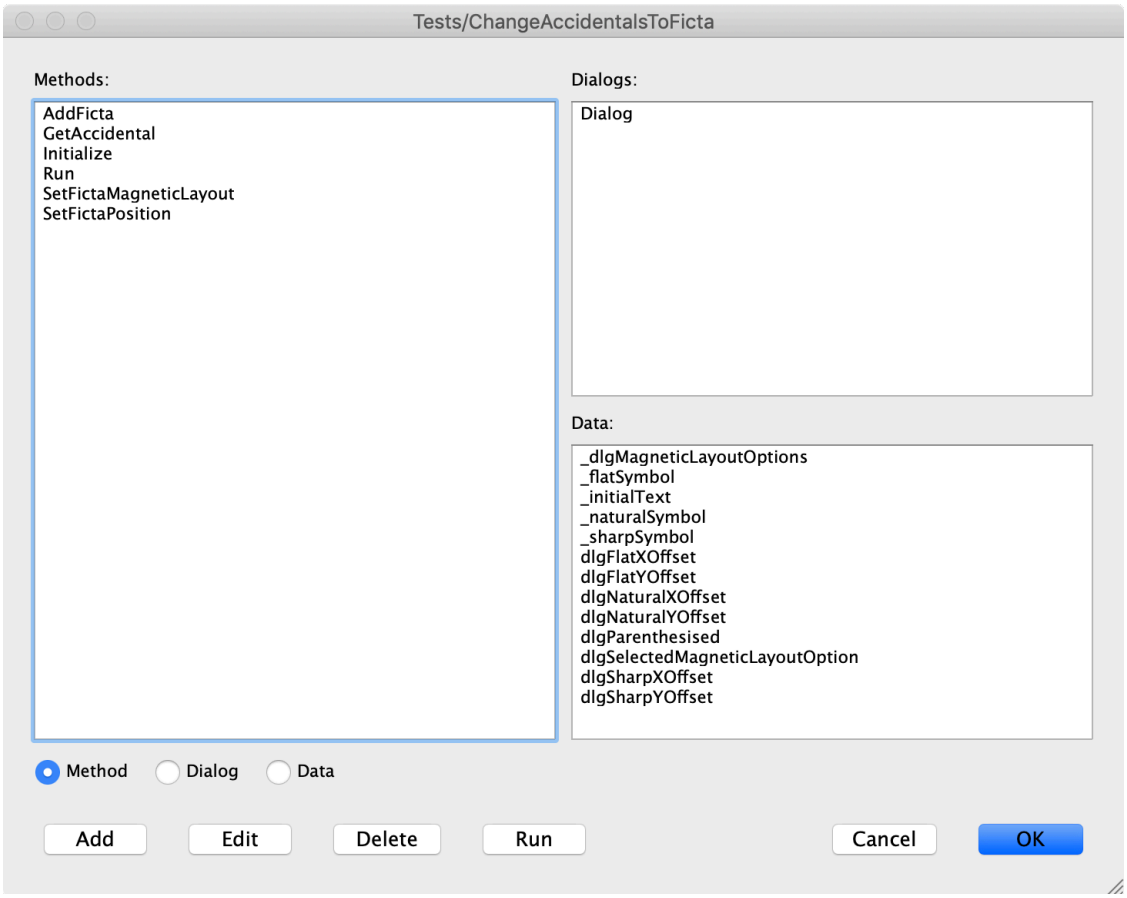
Plug-in ve výběru v partituře zamění standardní posuvky u not za „fictu“ (symbol posuvky nad notou). Uživatel může vybrat jednotlivé noty nebo udělat výběr několika taktů a osnov najednou – každá viditelná posuvka v tomto výběru bude nahrazena.

Původní posuvky budou skryty (ne smazány), takže výška tónů během přehrávání zůstane nezměněna.

Uživatel může nastavit posun daných symbolů v ose x a y a zároveň nastavit jejich magnetic layout. Dále může zvolit, zda tyto symboly mají být v závorkách či nikoliv.

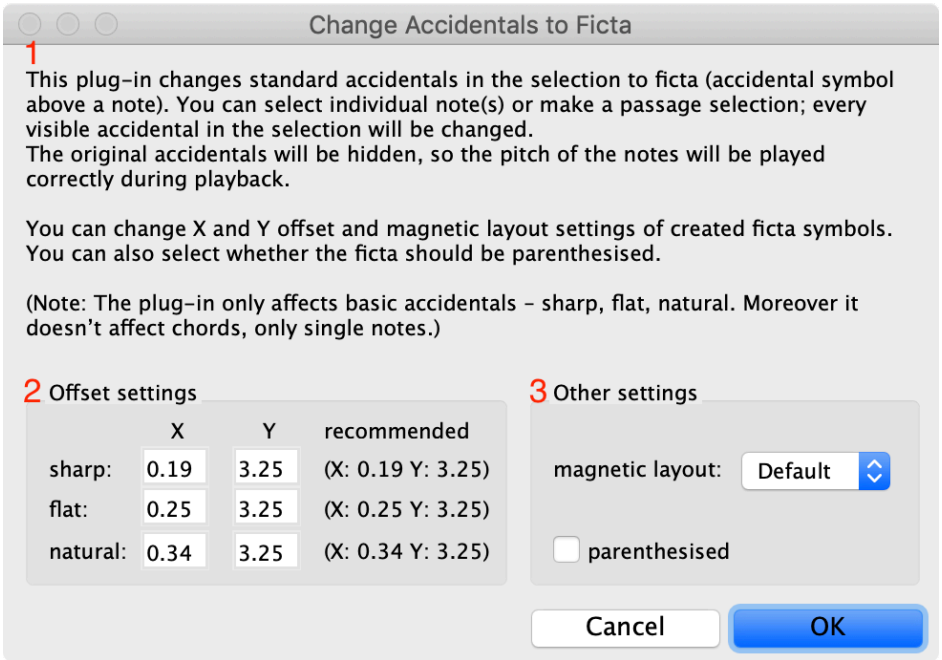
Plug-in nahradí jen základní typy posuvek – kříže, béčka a odrážky –, nezabývá se jinými typy (například mikrotonálními a dvojitými posuvkami). Dále se nezabývá souzvuky tónů (u nich by umístění posuvek nad osnovu bylo nepřehledné a nepraktické), ale jen jednotlivými notami.

4.3.1.1 Okno úprav plug-inu



Obrázek 26: metody, dialogová okna a data plug-inu 3

4.3.1.2 Dialogové okno plug-inu



Obrázek 27: dialogové okno plug-inu 3

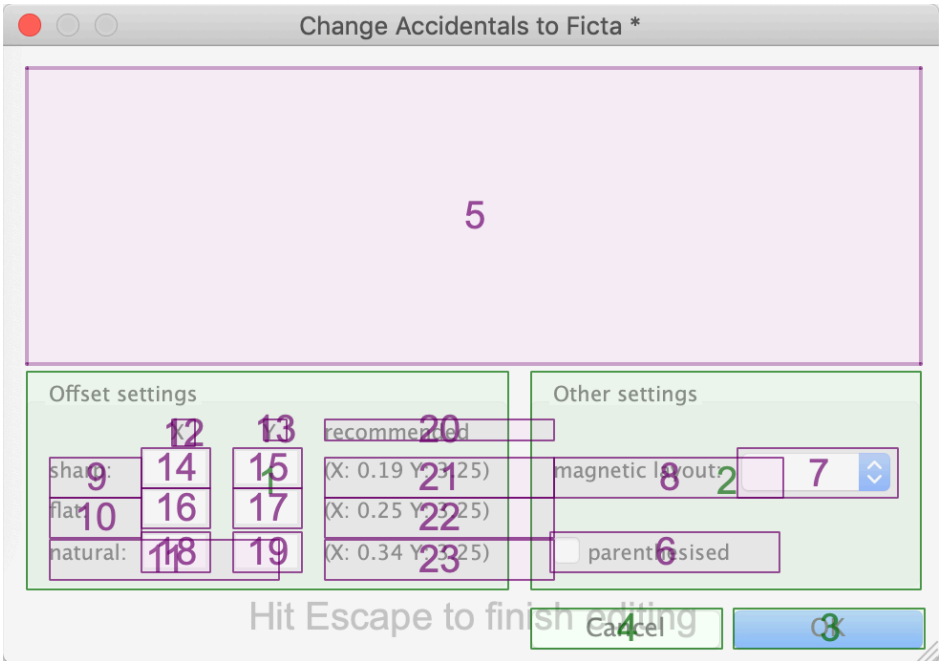
1. informace o plug-inu
2. nastavení posunu jednotlivých symbolů posuvek (křížku, béčka a odrážky) v osách x a y
3. další nastavení
    - magnetic layout – má-li být ponechán jako ve zbytku partitury (*default*), či vypnut (*off*) nebo zapnut (*on*)
    - checkbox „parenthesised“ určující, zda vytvořené symboly mají být v závorce či nikoliv

Uživatel zvolí posuny symbolů (jsou mu navržené výchozí hodnoty, které posuvky umístí ve vhodné vzdálenosti nad osnovu doprostřed notové hlavičky). Dále zvolí nastavení magnetic layout vytvořených symbolů a zda mají být tyto symboly v závorkách.

Tlačítkem „Cancel“ plug-in zruší, tlačítkem „OK“ plug-in spustí a ten nahradí posuvky za symboly s příslušným nastavením.

Creation order

Takto je nastaveno creation order (tab order) dialogového okna:



Obrázek 28: creation order dialogového okna plug-inu 3

Jelikož se některé prvky dialogového okna překrývají, nebylo možné nastavit tab order tak, aby začínal od umístění jednotlivých posuvek v oblasti „Offset settings“.

4.3.1.3 Data (proměnné)

Proměnné začínající podtržítkem značí konstanty:

- _dlgMagneticLayoutOptions – hodnoty, které se objeví na výběr v list boxu v oblasti „Other settings“ (viz Obrázek 27 – 3) pro nastavení magnetic layout:
  - „Default“ – ponechá nastavení stejně, jako je ve zbytku partitury (na kartě Layout > Magnetic Layout > Magnetic Layout)
  - „On“ – vytvořené symboly posuvek budou mít zapnutý magnetic layout (bez ohledu na nastavení zbytku partitury)
  - „Off“ – vytvořené symboly posuvek budou mít vypnutý magnetic layout (bez ohledu na nastavení zbytku partitury)
- _flatSymbol – číselná hodnotu symbolu béčko v programu Sibelius (258)
- _initialText – obsahuje informace o daném plug-inu, které se zobrazí v dialogovém okně (viz Obrázek 27 – 1)
- _naturalSymbol – číselná hodnotu symbolu odrážka v programu Sibelius (260)
- _sharpSymbol – číselná hodnotu symbolu křížek v programu Sibelius (262)
- dlgFlatXOffset – posun symbolu béčko v horizontální ose x (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 0,19, aby byl symbol zarovnán na střed notové hlavičky
- dlgFlatYOffset – posun symbolu béčko ve vertikální ose y (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 3,25
- dlgNaturalXOffset – posun symbolu odrážka v horizontální ose x (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 0,34, aby byl symbol zarovnán na střed notové hlavičky
- dlgNaturalYOffset – posun symbolu odrážka ve vertikální ose y (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 3,25
- dlgParenthesised – proměnná true/false spojená s checkboxem v oblasti „Other settings“ (viz Obrázek 27 – 3) uchovávající, jestli je daný checkbox zaškrtnut či nikoliv, tzn. jestli mají být vloženy symboly posuvek v závorkách nebo ne
- dlgSelectedMagneticLayoutOption – vybraná hodnota z list boxu v oblasti „Other settings“ (viz Obrázek 27 – 3) obsahující požadovaného nastavení magnetic layout pro přidané symboly posuvek

- dlgSharpXOffset – posun symbolu křížek v horizontální ose x (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 0,25, aby byl symbol zarovnán na střed notové hlavičky
- dlgSharpYOffset – posun symbolu křížek ve vertikální ose y (viz „Offset settings“ – Obrázek 27 – 2); uživateli je doporučena hodnota 3,25

4.3.1.4 Metody

Kódy všech metod jsou uvedeny v příloze.

Run

Hlavní metoda Run nejprve zkontroluje, zdali je otevřena nějaká partitura, a pokud ano, zda partitura obsahuje osnovy, nástroje. Dále ověří, zda je v partituře něco vybráno. Pokud ne, zobrazí uživateli dialog (message box) s otázkou, zda chce plug-in spustit na celou partituru či nikoliv (viz kapitola 4.3.1.5). Pokud uživatel stiskne tlačítko „No“, plug-in nic nevykoná a skončí; pokud stiskne tlačítko „Yes“, plug-in vybere celou partituru a na ní bude probíhat zbytek příkazů.

Dále metoda Run zobrazí uživateli dialogové okno plug-inu, kde zvolí příslušné nastavení a stiskne-li tlačítko „OK“, u všech not ve výběru provede náhradu jejich standardní posuvky (pokud nějakou mají) na symbol musica ficta (symbol dané posuvky nad notou – podle uživatelovy volby bude tento symbol v závorce nebo ne) a nastaví její pozici a magnetic layout. Původní posuvku skryje.

Pokud je na nějakém místě souzvuk více tónů, plug-in takovýto útvar přeskočí. Zároveň přeskočí mikrotonální a dvojité posuvky.

GetAccidental

Tato metoda má parametr „note“ (odkaz na notu). Podle posuvky, která je u dané noty, vrátí hodnotu (číslo) příslušného symbolu (ty jsou uloženy v globálních proměnných _sharpSymbol, _naturalSymbol a _flatSymbol – viz kapitola 4.3.1.3).

AddFicta

Tato metoda má parametry „noteRest“ (odkaz na konkrétní pozici v taktu) a „type“ (číselné označení symbolu, který má být nad notu přidán – křížek, béčko či odrážka).

Metoda si z parametru „noteRest“ zjistí pozici v daném taktu a na ni přidá symbol podle parametru „type“ (pokud uživatel zaškrtl checkbox „parenthesised“, je typ symbolu změněn na stejný, jen v závorce, jehož číslo je u všech symbolů posuvek v tabulce symbolů větší o 16). Následně změní velikost přidaného symbolu na „CueSize“, tj. o něco menší, než je standardní velikost (jedná se čistě o estetickou úpravu).

Na konci vrátí odkaz na právě přidaný symbol.

SetFictaMagneticLayout

Tato metoda má parametr „ficta“ (odkaz na symbol posuvky nad notou). Této posuvce nastaví magnetic layout dle uživatelova výběru v dialogovém okně na jednu ze tří možností:

- „On“ – symbolu zapne magnetic layout (bez ohledu na nastavení zbytku partitury – na kartě Layout > Magnetic Layout > Magnetic Layout)
- „Off“ – symbolu vypne magnetic layout (bez ohledu na nastavení zbytku partitury)
- „Default“ – ponechá nastavení symbolu stejné, jako je ve zbytku partitury

SetFictaPosition

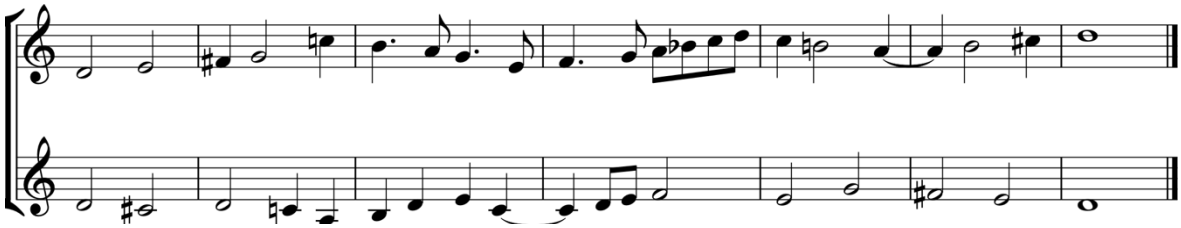
Tato metoda má parametr „ficta“ (odkaz na symbol posuvky nad notou). Nastaví této posuvce pozici podle zadaných hodnot v dialogovém okně plug-inu.

Podle typu posuvky nastaví její posun (*offset*). Zároveň si poradí s potřebným posunem v horizontální ose x, pokud jsou dané posuvky v závorce.

(Hodnoty posunu musí být vynásobeny 32, jelikož pozice je v jazyce ManuScript zadávána v hodnotách 1/32 skutečné pozice.)

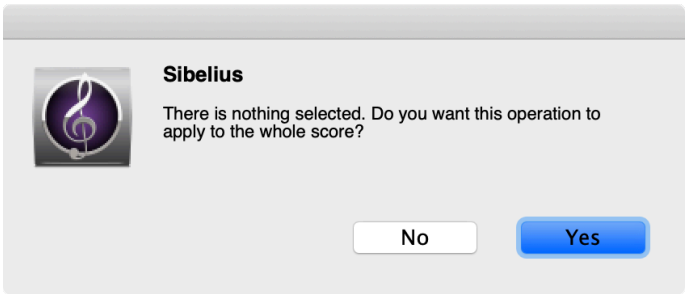
4.3.1.5 Příklad použití

Uživatel například vytvoří takovýto notový zápis:



Obrázek 29: příklad použití plug-inu 3 – před spuštěním plug-inu

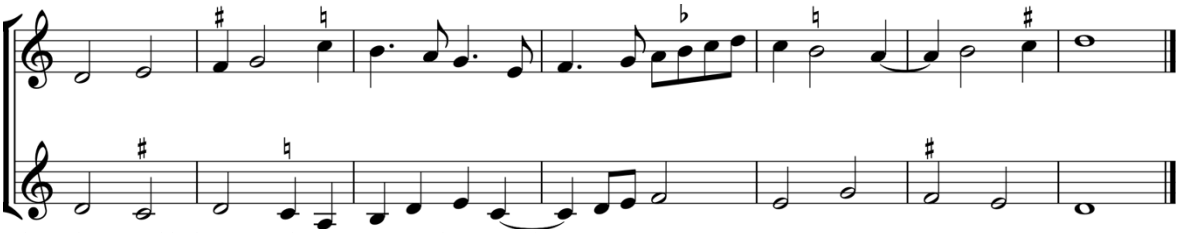
Pokud uživatel v partituře nic nevybere a spustí plug-in „Change Accidentals to Ficta“, zobrazí se tento dialog:



Obrázek 30: zobrazený dialog, pokud uživatel neučinil výběr

Pokud klikne na tlačítko „No“, plug-in bude zrušen. Pokud stiskne tlačítko „Yes“, zobrazí se dialogové okno plug-inu (viz kapitola 4.3.1.2), kde nastaví vše potřebné – např. offset posuvek nechá nepozměněné, magnetic layout ponechá „Default“ a checkbox „parenthesised“ nechá nezaškrtnutý.

Po stisknutí tlačítka „OK“ plug-in nahradí všechny posuvky v partituře příslušnými symboly nad osnovou:



Obrázek 31: příklad použití plug-inu 3 – po skončení plug-inu

5 VÝSLEDKY A DISKUSE

5.1 PLUG-IN 1

Plug-in 1 (Export Individual Staves as Predominant Instrument in Mix) navrhuje řešení při exportu audio souborů v programu Sibelius, pokud je požadováno vyexportovat pro každý nástroj vlastní soubor, kde bude daný nástroj zvýrazněn v mixu (bude zesílen a ostatní nástroje zeslabeny). Plug-in tento postup automatizuje a šetří tak práci uživatele.

5.1.1 Možnosti rozšíření

Plug-in by bylo možné rozšířit o další související funkcionalitu, např.:

- uživatel by si mohl zvolit, jestli má plug-in vypsat výsledek exportování
- možnost nastavení dalších parametrů mixu (panorama, poměry hlasitostí zvýrazněných a potlačených nástrojů, změnit zvuk nástroje atd.)
- možnost výběru osnov, pro které se má audio soubor vyexportovat
- zda se má vyexportovat i metronomický klik
- možnost vyexportovat i MIDI soubory, nejen audio soubory
- kontrolovat, zda je ve zvolené složce již soubor se stejným názvem (a upozornit uživatele, že takový soubor bude přepsán)

5.2 PLUG-IN 2

Plug-in 2 (Export MIDI with Click) automatizuje postup při exportování MIDI souboru s metronomickým klikem z dané partitury; tento druh exportu není v programu Sibelius možný a musí se obcházet poměrně složitým postupem. Plug-in přesně podle tohoto postupu funguje a tuto činnost tak automatizuje.

5.2.1 Možnosti rozšíření

Bylo by možné funkcionalitu tohoto plug-inu propojit s plug-inem předchozím (Export Individual Staves as Predominant Instrument in Mix). Uživatel by mohl vybrat, chce-li pro

každou osnovu vyexportovat zvláštní audio nebo MIDI soubor se zvýrazněným daným nástrojem v mixu. A zároveň jestli chce přidat metronomický klik a count-in takty.

Dále by bylo možné rozšířit plug-in o nastavení jiných druhů metronomických kliků (jiné výšky tónů, jiné akcenty či rytmické patterny, jiný zvuk kliku jako takového) či možnost volby umístění vyexportovaného souboru.

Také by uživatel mohl zvolit, jestli se do vokálních osnov má do count-in taktů přidat i zadání tónů pro zpěváky, aby při poslechu daného MIDI souboru věděli, na jakém tónu jejich linka začíná. (Nebylo by to potřeba u instrumentálních partů.)

5.3 PLUG-IN 3

Plug-in 3 (Change Accidentals to Ficta) řeší specializovaný problém notace staré hudby, kdy se některé posuvky psaly nad notu, a ne před notu, jak je tomu v notaci modernější. U not, které uživatel vybere, nahradí standardní posuvku příslušným symbolem nad notou, aniž by změnil přehrávanou výšku tónu. Tím automatizuje tuto činnost za uživatele.

Tuto problematiku řeší jinak než již v programu Sibelius dostupný plug-in „Add Ficta Above Note“, který nad jednotlivě vybrané noty přidá zvolenou posuvku. Tento plug-in má ale jen velmi omezenou funkcionalitu – neumožňuje přidávat příslušné posuvky v závorkách, nastavit jejich umístění nad notou (posun v osách x a y). Zároveň může být nepraktické tento plug-in spouštět znovu při každém přidávání posuvky k notě. Vytvořený plug-in „Change Accidentals to Ficta“ lze spustit až po napsání not se standardními posuvkami a změnit je tak všechny najednou.

5.3.1 Možnosti rozšíření

Tento plug-in by bylo možné rozšířit o další související funkcionalitu, např.:

- možnost zvolit posuvky v hranatých závorkách (ty se v praxi používají pro ediční úpravy a poznámky)
- nahrazování i jiných posuvek než základních (dvojité posuvky, čtvrttónové posuvky apod.) – toto rozšíření je ale sporné, jelikož jiné než základní posuvky se v „dobové“

hudbě nevyskytovaly, tak se nabízí otázka, zda by se pro takovou funkcionalitu našlo uplatnění

- možnost volby jiných velikostí symbolů, které budou nad notu přidány

K tomuto plug-inu by bylo možné napsat ještě jeden související, který by řešil transpozici hudby se zachováním daných symbolů nad notou. Po této funkcionalitě se ptají i uživatelé programu Sibelius, kteří se takovéto notaci věnují. (36)

5.4 ÚSKALÍ TVORBY PLUG-INŮ

Program Sibelius je velmi specializovanou aplikací, a proto si i tvorba vlastních plug-inů v jazyce ManuScript žádá nejen základní znalost programování, ale také znalost hudební terminologie, principů atd. Pro člověka hudby neznalého by taková práce mohla být velmi náročnou činností.

Dalším problémem je nepřenositelnost vytvořených plug-inů do jiných hudebních softwarů, jelikož skriptovací jazyk ManuScript byl vyvinut speciálně pro potřeby programu Sibelius. Principy a postupy, které plug-iny používají, se však lze inspirovat pro potřeby jiných programů – ať už pro přímou práci v aplikaci či pro tvorbu plug-inů nebo maker v jiných notačních programech.

Proměnné typu string

V kódech metod plug-inů (které jsou uvedeny v přílohách práce) si lze povšimnout netradičního dosazování proměnných typu string, např. v metodě CreateUniqueStaffName prvního plug-inu (viz příloha 8.1.1.6):

```
uniqueName = "" & originalName;
```

To jest způsobeno programátorskou chybou neboli „bugem“ v jazyce ManuScript při porovnávání dvou proměnných typu string, kdy první z nich je pouze jednoznaková (jedná se o interní typ proměnné character, který však není blíže definován v oficiálním manuálu jazyka ManuScript).

```
a = "A";
b = "Abc123456";
if (a = b)
{
  …
}
```

V takovémto kódu je při porovnání proměnných v podmínce příkazu *if* druhá proměnná převedena také na typ character (je zkrácena pouze na první znak „A“) a podmínka je tedy splněna – jazyk ManuScript považuje obě proměnné za shodné.

Tento problém lze řešit umístěním delší proměnné na první místo v podmínce, v tomto případě:

```
if (b = a)
```

Pak ManuScript porovná proměnné správně a podmínka splněna není.

Obecnějším řešením je již uvedené připojení prázdného řetězce ("") k proměnné. Poté vždy ManuScript interpretuje tuto proměnnou jako typ string, a ne character; ve zmíněném příkladu:

```
a = "" & a;
```

Tento problém vysvětluje již v roce 2014 Bob Zawalich na oficiálním fóru skriptovacího jazyka ManuScript a dodnes zůstává nevyřešen. (37)

6 ZÁVĚR

V první části práce byl připraven teoretický podklad pro vlastní práci v části druhé. Byl stručně charakterizován aplikační software obecně, a především specializovaný typ aplikací – notační programy – včetně příkladů nejpoužívanějších zástupců. Důraz byl kladen na program Sibelius, jeho funkcionalitu a součásti (potřebné pro pochopení problematiky v druhé části práce) a na skriptovací jazyk ManuScript, pomocí něhož jsou pro tuto aplikaci vytvářeny plug-iny rozšiřující jeho funkcionalitu.

Ve druhé části práce byly analyzovány tři konkrétní specializované neefektivní postupy v tomto programu:

- exportování audio souborů pro každý nástroj partitury s danou osnovou ve výsledném mixu zesílenou a s ostatními osnovami s potlačenou hlasitostí,
- exportování partitury do MIDI souboru včetně metronomického kliku,
- nahrazování standardních posuvek před notou za symbol příslušné posuvky nad notou (tzv. musica ficta).

Následně bylo navrženo řešení pomocí tvorby vlastních plug-inů ve skriptovacím jazyce ManuScript, které zmíněné postupy zefektivnily či zautomatizovaly. V práci je charakterizováno jejich navržení, funkcionalita a jednotlivé komponenty (jejich dialogová okna, metody a používané globální proměnné). Činnost metod je vysvětlena slovně a jejich kódy jsou navíc uvedeny v přílohách této bakalářské práce. Nakonec je vždy na jednoduchém příkladu ukázáno, jak plug-in funguje v praxi.

Plug-iny nebylo možné vytvářet obecně i pro jiné notační programy, jelikož jazyk ManuScript je speciálním jazykem vytvořeným přímo pro užití v softwaru Sibelius. Z kódů jejich metod a z postupů, jakými dané plug-iny pracují, se ale lze při práci v jiných notačních softwarech inspirovat – dají se tak do určité míry zobecnit.

7 SEZNAM POUŽITÝCH ZDROJŮ

1. MIKLÁŠ, Michal. Software I. Informatika a grafika [online]. [vid. 2021-03-01]. Dostupné z: https://www.gjszlin.cz/ivt/esf/ostatni-sin/software-1.php
2. ČÍŽEK, David. Aplikační software [online]. B.m.: Střední škola informačních technologií a sociální péče, Brno, Purkyňova 97. 2010 [vid. 2021-03-01]. Dostupné z: https://moodle.sspbrno.cz/pluginfile.php/2059/mod_resource/content/0/ApS/Aplikacni_sw.pdf
3. Scorewriter [online]. 2020 [vid. 2020-10-02]. Dostupné z: https://en.wikipedia.org/wiki/Scorewriter
4. MURPHY, Caleb J. 7 Best Music Writing Software Programs for DIY Musicians. Careers In Music | Music Schools & Colleges [online]. 19. říjen 2019 [vid. 2021-01-04]. Dostupné z: https://www.careersinmusic.com/best-music-writing-software/
5. FENDER, Millie. Best music notation software 2021. TopTenReviews [online]. 4. únor 2021 [vid. 2021-02-10]. Dostupné z: https://www.toptenreviews.com/best-music-notation-software
6. Sibelius (scorewriter) [online]. 2020 [vid. 2020-08-25]. Dostupné z: https://en.wikipedia.org/wiki/Sibelius_(scorewriter)
7. Simkin – Scripting Language. Simkin Solutions [online]. [vid. 2021-02-09]. Dostupné z: http://www.simkin.co.uk/Simkin_Sibelius.shtml
8. AVID TECHNOLOGY, INC. ManuScript Language Guide for Sibelius | Ultimate. B.m.: Avid Technology, Inc. 2018
9. AVID TECHNOLOGY, INC. Sibelius Reference Guide (version 2018.6). B.m.: Avid Technology, Inc. 2018
10. Finale (software) [online]. 2021 [vid. 2021-02-10]. Dostupné z: https://en.wikipedia.org/wiki/Finale_(software)
11. REYNA, Rosendo. Finale Music Software. https://musicprintinghistory.org/ [online]. [vid. 2021-02-10]. Dostupné z: https://musicprintinghistory.org/finale/
12. HOBIN, Todd, Elissa MURPHY a Brian SWEENEY. Notation Software, Who Needs It? Making Music Magazine [online]. 29. září 2014 [vid. 2021-02-12]. Dostupné z: https://makingmusicmag.com/notation-software/
13. The Finale Family of Music Notation Software. Finale [online]. [vid. 2021-02-12]. Dostupné z: https://www.finalemusic.com/products/
14. Dorico [online]. 2021 [vid. 2021-02-12]. Dostupné z: https://en.wikipedia.org/wiki/Dorico
15. Sibelius Core Team Now at Steinberg, Building New Notation Tool. CDM Create Digital Music [online]. 20. únor 2013 [vid. 2021-02-12]. Dostupné z: https://cdm.link/2013/02/sibelius-core-team-now-at-steinberg-building-new-notation-tool/

16. WRIGHT, Katy. Steinberg announces new scoring software. Rhinegold [online]. 17. květen 2016 [vid. 2021-02-12]. Dostupné z: https://www.rhinegold.co.uk/music_teacher/steinberg-announces-new-scoring-software/

17. What is Dorico: Discover all the features. Steinberg [online]. [vid. 2021-02-12]. Dostupné z: https://new.steinberg.net/dorico/features/

18. SPREADBURY, Daniel. Dorico is available now, first update coming November. Dorico [online]. 1. listopad 2016 [vid. 2021-02-12]. Dostupné z: https://blog.dorico.com/2016/11/dorico-is-available-now-first-update-coming-november/

19. LIS, Dan. Best Music Notation Software of 2020. Composer's Toolbox [online]. 22. červenec 2020 [vid. 2021-02-12]. Dostupné z: https://composerstoolbox.com/2020/06/22/best-music-notation-software-of-2020/

20. Compare the versions of Dorico: Elements and Pro. Steinberg [online]. [vid. 2021-02-12]. Dostupné z: https://new.steinberg.net/dorico/compare-editions/

21. MuseScore [online]. 2021 [vid. 2021-02-24]. Dostupné z: https://en.wikipedia.org/wiki/MuseScore

22. MUSESCORE. MuseScore 3.6 – A Massive Engraving Overhaul! [online]. 15. leden 2021 [vid. 2021-02-12]. Dostupné z: https://www.youtube.com/watch?v=qLR40BGNy68&ab_channel=Musescore

23. LIS, Dan. MuseScore 3.6 – A massive step forward. Composer's Toolbox [online]. 9. únor 2021 [vid. 2021-02-24]. Dostupné z: https://composerstoolbox.com/2021/02/09/musescore-3-6-a-massive-step-forward/

24. BIELAWA, Herbert. Sibelius 7 Music Notation Software. Computer Music Journal [online]. 1997, 21(2), 99–105. ISSN 0148-9267. Dostupné z: doi:10.2307/3681118

25. BUTLER, Sam. Sibelius 2018 Now Available—What's New. Avid Blogs [online]. 25. leden 2018 [vid. 2021-02-28]. Dostupné z: http://www.avidblogs.com/sibelius-20181-now-available/

26. BUTLER, Sam. What's New in Sibelius—February 2021. Avid Blogs [online]. 25. únor 2021 [vid. 2021-02-28]. Dostupné z: https://www.avid.com/resource-center/whats-new-in-sibelius-february-2021

27. Sibelius Comparison. Avid [online]. [vid. 2021-02-28]. Dostupné z: https://www.avid.com/sibelius/comparison

28. Ribbon (computing) [online]. 2021 [vid. 2021-02-10]. Dostupné z: https://en.wikipedia.org/wiki/Ribbon_(computing)

29. SPREADBURY, Daniel. Reflecting on the ribbon. Scoring Notes [online]. 16. březen 2012 [vid. 2021-02-10]. Dostupné z: http://www.scoringnotes.com/opinion/reflecting-on-the-ribbon/

30. SPREADBURY, Daniel. Sibelius 7 is here! Scoring Notes [online]. 27. červenec 2011 [vid. 2021-02-08]. Dostupné z: https://www.scoringnotes.com/news/sibelius-7-is-here/

31. Standard MIDI Files (SMF) Specification. MIDI Association [online]. [vid. 2021-02-10]. Dostupné z: https://www.midi.org/specifications-old/item/standard-midi-files-smf

32. Exporting midifiles with click? In: The Sibelius Forum [online]. 2. prosinec 2010 [vid. 2021-02-25]. Dostupné z: https://www.sibeliusforum.com/viewtopic.php?t=4943

33. ZAWALICH, Bob. Using plugins in Sibelius [online]. srpen 2018 [vid. 2021-02-10]. Dostupné z: http://www.bobzawalich.com/wp-content/uploads/2018/07/Using-plugins-In-Sibelius-Ultimate.pdf

34. PUFF, Robert. Installing Plug-ins in Sibelius. Of Note [online]. 14. prosinec 2012 [vid. 2021-02-08]. Dostupné z: https://www.rpmseattle.com/of_note/installing-plug-ins-in-sibelius/

35. DUFFIN, Ross W. Un-notated Accidentals (A.K.A. Musica Ficta) | Dr. Ross W. Duffin. College of Arts and Sciences [online]. [vid. 2021-01-29]. Dostupné z: https://casfaculty.case.edu/ross-duffin/muhi-3441-multimedia/muhi-3441-medren-accidentals/un-notated-accidentals-a-k-a-musica-ficta/

36. FREESTONE, Nicholas. Musica Ficta. Sibelius Feedback Community [online]. 6. duben 2016 [vid. 2021-02-28]. Dostupné z: https://sibelius.ideascale.com/a/dtd/Musica-Ficta/231677-22221

37. ZAWALICH, Bob. Sibelius ManuScript plug-in developers - bug? In: Sibelius ManuScript plug-in developers [online]. 29. srpen 2014 [vid. 2021-03-06]. Dostupné z: http://sibelius-manuscript-plug-in-developers.3224780.n2.nabble.com/bug-td7572898.html

8 PŘÍLOHY

8.1 METODY PLUG-INŮ

8.1.1 Plug-in 1 (Export Individual Staves as Predominant Instrument in Mix)

8.1.1.1 Initialize

```
AddToPluginsMenu("Export Individual Staves as Predominant Instrument in
Mix","Run");
```

8.1.1.2 Run

```
if (Sibelius.ScoreCount = 0)
{
      Sibelius.MessageBox("Error! There is no score.");
      return False;
}

score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
      Sibelius.MessageBox("Error! There are no staves.");
      return False;
}

if (utils.IsMac())
{
      extension = _aiff;
}
else
{
      extension = _wav;
}

scoreName = score.FileName; //celá cesta k souboru
file = Sibelius.GetFile(scoreName);

//DIALOG STRINGS
dlgFileExtension = extension; //koncovka zobrazená v dialogu
dlgFilePath = file.Path; //jen cesta
dlgFileName = file.NameNoPath; //název souboru
if(dlgFileName = "")
{
      dlgFileName = _untitled;
}

//ULOŽENÍ PŮVODNÍHO NASTAVENÍ
```

```
originalStavesVolumes = CreateArray(); //uložení původní hlasitosti
jednotlivých osnov
for i = 0 to score.StaffCount
{
      originalStavesVolumes[i] = score.NthStaff(i+1).Volume;
}

arrUsedNames = CreateSparseArray(); //pole pro uložení již použitých
názvů exportovaných souborů

if (Sibelius.ShowDialog(Dialog, Self) = true)
{
      for each Staff s in score
      {
            s.Volume = 40;
      }
      for each Staff s in score
      {
            createdFileName = CreateFileName(s, arrUsedNames);
            s.Volume = 100;

            exported = score.SaveAsAudio(createdFileName);
            if (exported = false)
            {
                  Sibelius.MessageBox("Error. File " & createdFileName &
                  " couldn't be created.");
            }

            s.Volume = 40;
      }
}

//NÁVRAT DO PŮVODNÍHO STAVU
for i = 0 to score.StaffCount
{
      score.NthStaff(i+1).Volume = originalStavesVolumes[i];
}
```

8.1.1.3 CheckSelectedFilePath (path)

```
lastCharacter = CharAt(path, (Length(path) - 1));

if (lastCharacter != Sibelius.PathSeparator)
{
      path = path & Sibelius.PathSeparator;
}
return path;
```

8.1.1.4 GetFilePath

```
folder = Sibelius.SelectFolder();

if (IsObject(folder))
{
      dlgFilePath = folder.Name;
}

Sibelius.RefreshDialog();
```

8.1.1.5 CreateFileName (staff, arrUsedNames)

```
//_dlgStaffNamesOptions
//{
// "short instrument name"
// "full instrument name"
//}

i = utils.GetArrayIndex(_dlgStaffNamesOptions, dlgSelectedStaffName);

if (i = -1) //nemělo by se nikdy stát
{
      Sibelius.MessageBox("unexpected name error");
      return "";
}

switch (i)
{
      case(0) //vybrán krátký název
      {
            staffName = staff.ShortInstrumentName;
            if (CharAt(staffName,Length(staffName)-1) = ".") //odstranění
            případné tečky za krátkým jménem
            {
                  staffName = Substring(staffName,0,Length(staffName)-1);
            }
      }
      case(1)
      {
            staffName = staff.FullInstrumentName;
      }
}

staffName = CreateUniqueStaffName(staffName, arrUsedNames);
arrUsedNames[arrUsedNames.Length] = staffName;
utils.SortArray(arrUsedNames,false); //setřídění pole, aby se nemuselo
procházet víckrát

dlgFilePath = CheckSelectedFilePath(dlgFilePath);

fileName = dlgFilePath & dlgFileName & "_" & staffName &
dlgFileExtension;

return fileName;
```

8.1.1.6 CreateUniqueStaffName (originalName, arrUsedNames)

```
uniqueName = "" & originalName; //"" přidáno kvůli bugu v jazyce
ManuScript
i = 2;

for each name in arrUsedNames
{
      if (uniqueName = name)
      {
            trace("měním, protože " & uniqueName & " = " & name);
            uniqueName = originalName & "(" & i & ")";
            i = i + 1;
```

```
	}
}

return uniqueName;
```

8.1.2 Plug-in 2 (Export MIDI with Click)

8.1.2.1 Initialize

```
AddToPluginsMenu('Export MIDI with Click','Run');
```

8.1.2.2 Run

```
if (Sibelius.ScoreCount = 0)
{
	Sibelius.MessageBox('Error! There is no score.');
	return False;
}

score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
	Sibelius.MessageBox('Error! There are no staves.');
	return False;
}

extension = '.mid';
scoreName = score.FileName;
file = Sibelius.GetFile(scoreName);
filePath = file.Path;
fileName = file.NameNoPath;
if(fileName = '')
{
	fileName = '' & _untitled;
}

dlgFileName = filePath & fileName & extension;

dialogOK = DoDialog(Dialog);
if (dialogOK = false)
{
	return false;
}

clickTrack = CreateClickTrack(score);
if (clickTrack = null)
{
	return false;
}
//trace('Click track vytvořen.');

if (dlgAddCountInBars = true)
{
	AddCountInBars(score, dlgNumOfCountInBars);
}
```

```
FillClickTrack(clickTrack);

score.SaveAs(dlgFileName,'Midi');

if (dlgDeleteClickTrack = true)
{
      DeleteClickTrack(clickTrack);
      //trace('ClickTrack odstraněn.');
}

if (dlgDeleteCountInBars = true)
{
      DeleteCountInBars(score);
}
```

8.1.2.3 DoDialog (dialog)

```
//INITIAL SETTINGS
Sibelius.EnableControlById(Self, dialog, "IDC_editNumOfCountInBars",
false);
Sibelius.EnableControlById(Self, dialog, "IDC_staticBars", false);
Sibelius.EnableControlById(Self, dialog, "IDC_checkDeleteCountInBars",
false);

dlgAddCountInBars = false;
dlgDeleteCountInBars = false;
dlgNumOfCountInBars = 2;
dlgClickTrackVolume = 60;
dlgDeleteClickTrack = true;


while (Sibelius.ShowDialog(dialog,Self) = True)
{
      if (CheckDialog() = True)
      {
            return True;
      }
}

return False;
```

8.1.2.4 CheckDialog

```
//CHECK CLICK TRACK VOLUME
if (dlgClickTrackVolume = '')
{
      Sibelius.MessageBox('Error! Click track volume is empty.');
      return false;
}

clickTrackVolume = 0 + dlgClickTrackVolume; //převedení na číslo
if ((utils.IsNumeric(clickTrackVolume, True) = False) or
(clickTrackVolume < 0) or (clickTrackVolume > 127))
{
      Sibelius.MessageBox('Error! Click track volume must be an integer
between 0 and 127 (inclusive).');
      return false;
```

```
}

//CHECK NUMBER OF COUNT-IN BARS
numOfCountInBars = 0 + dlgNumOfCountInBars; //převedení na číslo
if ((dlgAddCountInBars = true) and ((utils.IsNumeric(numOfCountInBars,
True) = False) or (numOfCountInBars <= 0)))
{
      Sibelius.MessageBox('Error! Number of count-in bars must be an
integer larger than 0.');
      return false;
}

return true;
```

8.1.2.5 EnableOrDisableCountInSettings

```
if (dlgAddCountInBars = true)
{
      Sibelius.EnableControlById(Self, Dialog,
'IDC_editNumOfCountInBars', true);
      Sibelius.EnableControlById(Self, Dialog, 'IDC_staticBars', true);

      Sibelius.EnableControlById(Self, Dialog,
'IDC_checkDeleteCountInBars', true);
      dlgDeleteCountInBars = true;
}
else
{
      Sibelius.EnableControlById(Self, Dialog,
'IDC_editNumOfCountInBars', false);
      Sibelius.EnableControlById(Self, Dialog, 'IDC_staticBars', false);

      Sibelius.EnableControlById(Self, Dialog,
'IDC_checkDeleteCountInBars', false);
      dlgDeleteCountInBars = false;
}

Sibelius.RefreshDialog();
```

8.1.2.6 CreateClickTrack (score)

```
clickTrack =
score.CreateInstrumentAtBottomReturnStave('instrument.unpitched.woodblock
.5lines',True,'' & _clickNameFull,'' & _clickNameShort);

if (clickTrack = null)
{
      Sibelius.MessageBox('Error! A staff for the click track could not
be created.');
      return null;
}

clickTrack.Volume = dlgClickTrackVolume;

return clickTrack;
```

8.1.2.7 AddCountInBars (score, numOfBars)

```
timeSignature = score.SystemStaff.CurrentTimeSignature(1); //taktové
označení daného taktu

numberOfNotesPerBar = timeSignature.Numerator;
durationOfNote = Whole / timeSignature.Denominator; //délka jedné noty
podle taktového označení
durationOfStandardBar = durationOfNote * numberOfNotesPerBar;

durationOfFirstBar = score.SystemStaff[1].Length;

//trace(durationOfFirstBar);

if (durationOfFirstBar < durationOfStandardBar) //pokud je to předtaktí
{
      score.InsertBars(1, 1, durationOfStandardBar - durationOfFirstBar);
      //před předtaktí dám takt, který délkou doplní předtaktí na
standardní takt
      numOfBars = numOfBars - 1;
      score.SystemStaff[1].AddSpecialBarline(SpecialBarlineInvisible);
//'spojí' tento takt pomocí neviditelné taktové čáry
}

score.InsertBars(numOfBars, 1); //před takt 1 vloží numOfBars taktů
standardní délky
```

8.1.2.8 FillClickTrack (staff)

```
timeSignature = staff.ParentScore.SystemStaff.CurrentTimeSignature(1);

numberOfNotesPerBar = timeSignature.Numerator;
durationOfNote = Whole / timeSignature.Denominator; //délka jedné noty
podle taktového označení
durationOfStandardBar = durationOfNote * numberOfNotesPerBar;

for i = 1 to (dlgNumOfCountInBars + 1)
{
      bar = staff.NthBar(i);
      if ((i = dlgNumOfCountInBars) and (bar.Length % durationOfNote >
0))
      {
            FillOneBarWithTiedLastNote(staff, bar);
      }
      else
      {
            FillOneBar(staff, bar);
      }
}

startIndex = dlgNumOfCountInBars + 1;
bar = staff.NthBar(startIndex); //standarní první takt

if (bar.Length < durationOfStandardBar) //předtaktí
{
      FillPickUpBar(staff, bar);
      startIndex = startIndex + 1;
}
```

```
if (startIndex <= staff.BarCount)
{
      for i = startIndex to (staff.BarCount + 1)
      {
            bar = staff.NthBar(i);
            FillOneBar(staff, bar);
      }
}
```

8.1.2.9 FillOneBar (staff, bar)

```
//vyplní daný takt dané osnovy metronomickým klikem
//poradí si i nepravidelnými takty (kratšími i delšími, než je standardní
takt daného taktového označení

timeSignature =
staff.ParentScore.SystemStaff.CurrentTimeSignature(bar.BarNumber);
//taktové označení daného taktu

numberOfNotesPerBar = timeSignature.Numerator;
durationOfNote = Whole / timeSignature.Denominator; //délka jedné noty
podle taktového označení
durationOfStandardBar = durationOfNote * numberOfNotesPerBar;

position = 0;
pitchOfFirstBeat = 67; //67 = G
pitchOfOtherBeats = 64; //64 = E
pitch = pitchOfFirstBeat;

while((position + durationOfNote) <= bar.Length)
{
      bar.AddNote(position, pitch, durationOfNote);

      position = position + durationOfNote;

      if (position != 0)
      {
            pitch = pitchOfOtherBeats;
      }
}

durationRemaining = bar.Length - position;

if (durationRemaining > 0) //nepravidelný takt, který není dělitelný
délkou základní noty
{
      bar.AddNote(position, pitch, durationRemaining); //pitch bude G,
pokud to bude první doba, jinak E (změní se ve whilu)
}
```

8.1.2.10 FillOneBarWithTiedLastNote (staff, bar)

```
//tato funkce slouží k vyplnění nedělitelných taktů count-inu - poslední
nota musí být navázána do první noty předtaktí
```

```
timeSignature =
staff.ParentScore.SystemStaff.CurrentTimeSignature(bar.BarNumber);
//taktové označení daného taktu

numberOfNotesPerBar = timeSignature.Numerator;
durationOfNote = Whole / timeSignature.Denominator; //délka jedné noty
podle taktového označení
durationOfStandardBar = durationOfNote * numberOfNotesPerBar;

position = 0;
pitchOfFirstBeat = 67; //67 = G
pitchOfOtherBeats = 64; //64 = E
pitch = pitchOfFirstBeat;

while((position + durationOfNote) <= bar.Length)
{
      bar.AddNote(position, pitch, durationOfNote);

      position = position + durationOfNote;

      if (position != 0)
      {
            pitch = pitchOfOtherBeats;
      }
}

bar.AddNote(position, pitch, bar.Length - position, true);
```

8.1.2.11 FillPickUpBar (staff, bar)

```
//vyplní předtaktí dané osnovy metronomickým klikem, oproti normálnímu
taktu předtaktí bude vyplňováno odzadu

timeSignature =
staff.ParentScore.SystemStaff.CurrentTimeSignature(bar.BarNumber);
//taktové označení daného taktu

numberOfNotesPerBar = timeSignature.Numerator;
durationOfNote = Whole / timeSignature.Denominator; //délka jedné noty
podle taktového označení
durationOfStandardBar = durationOfNote * numberOfNotesPerBar;

position = bar.Length - durationOfNote;
pitch = 64; //64 = nota E; první doba (G) nebude (jen při count-inu v
první době)

n = 0; //number of added notes
while(position >= 0)
{
      bar.AddNote(position, pitch, durationOfNote);
      n = n + 1;
      position = position - durationOfNote;
}

durationRemaining = durationOfNote + position;
if (durationRemaining > 0)
{
      if ((n = (numberOfNotesPerBar - 1)) and (dlgCountIn = true))
//první nota je v rámci první doby + se bude dělat count-in
```

```
	{
		pitch = 67; //67 = nota G
		bar.AddNote(0, pitch, durationRemaining);
	}
	else //něco zbude, ale nebude to v rámci první doby
	{
		bar.AddNote(0, pitch, durationRemaining);
	}
}
```

8.1.2.12 DeleteClickTrack (staff)

```
score = staff.ParentScore;
staffNum = staff.StaffNum;
numOfBars = staff.BarCount;

score.Selection.SelectPassage(1, numOfBars, staffNum, staffNum);
score.Selection.Delete(True);
```

8.1.2.13 DeleteCountInBars (score)

```
score.Selection.SelectSystemPassage(1,dlgNumOfCountInBars);
score.Selection.Delete(False);
```

8.1.3 Plugin 3 (Change Accidentals to Ficta)

8.1.3.1 Initialize

```
AddToPluginsMenu("Change Accidentals to Ficta","Run");
```

8.1.3.2 Run

```
if (Sibelius.ScoreCount = 0)
{
	Sibelius.MessageBox("Error! There is no score.");
	return False;
}

score = Sibelius.ActiveScore;

if (score.StaffCount = 0)
{
	Sibelius.MessageBox("Error! There are no staves.");
	return False;
}

selection = score.Selection;

n = 0;
for each i in selection
{
	n = n + 1;
}
```

```
if (n = 0)
{
      if (Sibelius.YesNoMessageBox("There is nothing selected. Do you
want this operation to apply to the whole score?") = true)
      {
            score.Selection.SelectPassage(1,score.SystemStaff.BarCount);
//vybrat vše
            selection = score.Selection;
      }
      else
      {
            return false;
      }
}

if (Sibelius.ShowDialog(Dialog, Self) = false) //stisknutí tlačítka
Cancel v Dialogu
{
      return false;
}

for each NoteRest nr in selection
{
      if (nr.NoteCount = 1) //pouze jednotlivé noty (ne pomlky nebo
souzvuky)
      {
            note = nr.Highest;

            if ((CharAt(note.Name, Length(note.Name) - 1) != "+") and
(CharAt(note.Name, Length(note.Name) - 1) != "-"))
            //noty s mikrotonálními posuvkami (mají poslední znak + nebo
-) se ignorují
            {
                  if ((note.IsAccidentalVisible = true) and
((note.Accidental = Sharp) or (note.Accidental = Natural) or
(note.Accidental = Flat)))
                  //není to akord, má viditelnou posuvku a je to #, b
nebo §
                  {
                        accidentalSymbol = GetAccidental(note);
                        ficta = AddFicta(nr, accidentalSymbol);
                        SetFictaPosition(ficta);
                        SetFictaMagneticLayout(ficta);
                        note.AccidentalStyle = 1; //hidden
                  }
            }
      }
}
```

8.1.3.3 AddFicta (noteRest, type)

```
position = noteRest.Position;

if (dlgParenthesised = true)
{
      type = type + 16; //stejné symboly, jen v závorce
}

ficta = noteRest.ParentBar.AddSymbol(position, type);
```

```
ficta.Size = CueSize;
//ficta.ResetPosition();

return ficta;
```

8.1.3.4 GetAccidental (note)

```
accidental = note.Accidental;

if(accidental = 1)
{
      return _sharpSymbol;
}
if(accidental = 0)
{
      return _naturalSymbol;
}
if(accidental = -1)
{
      return _flatSymbol;
}

//nemělo by nastat
return -1;
```

8.1.3.5 SetFictaMagneticLayout (ficta)

```
//_dlgMagneticLayoutOptions
//{
//     "Default"
//     "On"
//     "Off"
//}


i = utils.GetArrayIndex(_dlgMagneticLayoutOptions,
dlgSelectedMagneticLayoutOption);

if (i = -1) //nemělo by se nikdy stát
{
      Sibelius.MessageBox("Unexpected magnetic layout error.");
      return i;
}

switch (i)
{
      case(1) //On
      {
            ficta.UsesMagneticLayout = AlwaysDodge;

      }
      case(2) //Off
      {
            ficta.UsesMagneticLayout = SuppressDodge;
      }
      default //Default
      {
```

```
            ficta.UsesMagneticLayout = DefaultDodge;
      }
}
```

8.1.3.6 SetFictaPosition (ficta)

```
fictaSymbol = ficta.Index;
if (dlgParenthesised = true)
{
      fictaSymbol = fictaSymbol - 16; //bez závorky
}

fictaXOffset = 0;
fictaYOffset = 0;

switch(fictaSymbol)
{
      case(("" & _sharpSymbol))
      {
            fictaXOffset = dlgSharpXOffset * 32; //přepočítání na správné
hodnoty
            fictaYOffset = dlgSharpYOffset * 32;
            if (dlgParenthesised = true)
            {
                  fictaXOffset = fictaXOffset - (0.44 * 32); //posunutí
kvůli závorce
            }
      }
      case(("" & _flatSymbol))
      {
            fictaXOffset = dlgFlatXOffset * 32;
            fictaYOffset = dlgFlatYOffset * 32;
            if (dlgParenthesised = true)
            {
                  fictaXOffset = fictaXOffset - (0.59 * 32);
            }
      }
      case(("" & _naturalSymbol))
      {
            fictaXOffset = dlgNaturalXOffset * 32;
            fictaYOffset = dlgNaturalYOffset * 32;
            if (dlgParenthesised = true)
            {
                  fictaXOffset = fictaXOffset - (0.53 * 32);
            }
      }
}

ficta.Dx = fictaXOffset;
ficta.Dy = fictaYOffset;
```