

MIDI to PDF

Autor: Gheoace Mircea, 333AA

1. Introducere, prezentare a temei, prezentare a obiectivelor

Acest proiect are ca scop soluționarea unei dificultăți cu care s-au confruntat compozitorii de-a lungul timpului. Aceea de a putea crea fără a fi întrerupți. De multe ori chiar dacă reușeau să-și creeze un mediu izolat în care putea fi doar cu sursa inspirației tot trebuiau să se oprească pentru a scrie notele. Acest impediment poate fi astăzi remediat prin conectarea pianului electronic sau instrumentului muzical la un PC folosind un cablu MIDI to USB. Aceasta lucrare va trata fișierele MIDI ce reprezintă o piesă pentru pian. Scopul va fi de a transforma un fișier cu extensia „.mid” într-o partitură salvată într-un fișier PDF.

Midi reprezintă un format de salvare a informației sonore obținută de la unul sau mai multe instrumente până la maxim 16. Important de reținut este că MIDI nu salvează sunetul în sine ci informații ca: numele notelor, durata, numărul instrumentelor, numele acestora și alte astfel de informații. Structura unui fișier MIDI:

timp mesaj ***timp*** mesaj ***timp*** mesaj ...

Un fișier este format din evenimente, fiecare eveniment conținând o informație despre timp și mesajul în sine. Evenimentele conțin octeți de date și octeți de comenzi diferența fiind data de primul bit din octet. Dacă cel mai semnificativ bit este 1 este o comandă altfel este un octet de date. Primul octet este comanda fiind urmat de un număr variabil de biți de date. Sunt două moduri principale de a cânta o piesă: în monofonie sau polifonie. Practic acest lucru prevede dacă se pot sau nu cânta două note simultan.

Sa descriem acum un mesaj la nivel de biți. El este format din o comanda si mai mulți biți de date. Astfel comanda este la rândul ei formata din doua părți mici numite in engleza “nibble”. Pentru ușurătatea exprimării si lipsa in general a unei traduceri satisfăcătoare vom folosi in continuare termenul nibble.

Cele mai importante doua comenzi in MIDI sunt NOTE ON si NOTE OFF. Astfel se specifica când a fost cantata nota si când s-a terminat. In modelul monofonic se folosește doar NOTE ON înțelegând-se ca atunci când este apăsată o nota noua anterioara a fost eliberata.

	NOTE ON	NOTE OFF	Explicații
Octet comanda	1001 cccc	1000 cccc	cccc – reprezintă canalul. Pot fi valori in intervalul [0,127]
Octet date 1	0ppp pppp	0ppp pppp	ppp pppp reprezintă înălțimea notei. ‘p’ reprezintă prima litera a termenului englez ‘pitch’.
Octet date 2	0vvv vvvv	0000 0000	Viteza de apăsare a notei. La pian in funcție de cat de rapid apeși o clapa nota se aude mai tare sau mai încet. La NOTE OFF avem doar ‘0’ pentru ca nu contează cum ridici degetul de pe clapa, sunetul se oprește instantaneu.

Mai multe detalii pot fi găsite General MIDI elaborat de un consorțiu americano-japonez. Deoarece pot fi destul de multe detalii si timpul pentru a le implementa poate fi destul de lung lucrarea de fata își propune sa descrie un program care folosește un set de API pentru a extrage informațiile, câteva clase proprii pentru a le pune împreuna si ale prelucra ca apoi sa fie folosit un alt program pentru a le afișa sub forma de partitura. Scopul este de a crea o partitura cat mai apropiata de realitate deoarece fișierele midi create de om au un mare dezavantaj. Așa cum am descris anterior un fișier MIDI conține informații exacte despre când a fost cantata o nota, dar cel care cânta nu este un robot si ca atare nu poate cânta exact. Astfel doua durate care ar trebui sa fie identice, sa spunem doua șaisprezecimi, nu vor fi cantate de un om identic. Astfel pot fi propuse o serie de variante pentru a aproxima durata muzicala la care se refera timpul unei note. Deoarece acest program este pentru a-I ajuta pe compozitori in crearea unei partituri inițiale anumite inadvertente pot fi neglijate. Pentru realizarea unei partituri finale se pot folosi programe ca Sibelius sau Finale.

2. Prezentare pe scurt a suportului tehnic, scurtă trecere in revistă a unor realizări similare susținută prin referințe bibliografice

Pentru a realiza acest program am folosit un set de API numit JFugue. Acesta a dus la apariția unui standard numit Staccato de descriere a unui fișier MIDI printr-un sir de caractere ASCII ușor interpretabile de ochiul uman. Este dezvoltat in Java oferind astfel o portabilitate a programelor realizate cu el foarte buna si poate fi obținut de pe (1). De asemenea se poate găsi un set similar de API in C++ numit Midifile. Acesta poate fi obținut de pe site-ul (2) sau de pe repository-ul sau de pe GitHub. Pe site se poate găsi si documentație pentru toate funcțiile. Ambele soluții nu rezolva însă obținerea unei partituri oferind doar capacitatea de extragere a informațiilor din fișier. Pentru a putea realiza o partitura se poate folosi un program numit LilyPond bazat pe LaTeX. Acesta are propriul format de reprezentare a informației muzicale. Rezumând informația anterioara, procesul de lucru poate fi descris prin următoarea diagrama:

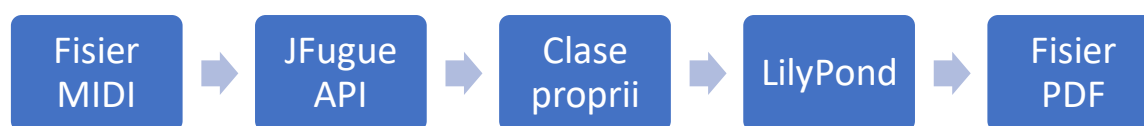


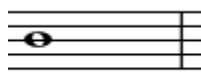
Diagrama 1

Realizări similar pot fi găsite pe internet. Câteva exemple ar fi

<https://musescore.org/en/handbook/midi-import> , https://www.8notes.com/midi_sheet_music/ si <https://solmire.com/miditosheetmusic/> . Codul este însă proprietar si nu poate fi accesat.

3. Prezentare tehnica a etapei de realizare/implementare

Algoritmul principal este cel de conversie a duratei fiecărei note din timp Staccato în durate muzicale¹. Pentru început este extras String-ul în format Staccato din fișierul MIDI. Ulterior aceste este prelucrat și transformat într-un String în format LilyPond.

Durata muzicala	Simbol	Durata in timp(durate de bază)
Nota întreaga		1

¹ Prin durate muzicale înțelegem nota întreagă, doime, pătrime, optime, șaisprezecime și combinațiile cu punct ale acestora.

Doime		0.5
Pătrime		0.25
Optime		0.125
Şaisprezecime		0.0625

Cum am precizat mai devreme un om nu poate atinge perfecţiunea de a avea aceeaşi durată în timp pentru fiecare notă. De aceea am decis să folosesc un algoritm de recunoaştere a notei bazat pe principiul înjumătăţirii. Durata cantată de om o vom nota cu α . Duratele din coloana „Durata în timp” le vom nota cu α_k . Astfel am presupus că deoarece $\alpha \in [\alpha_k, \alpha_{k+1}]$ şi $\alpha_{k+1} = 2 * \alpha_k$ durata notei este α_k dacă $\alpha \in [\alpha_k, \frac{3\alpha_k}{2})$ şi α_{k+1} dacă $\alpha \in [\frac{3\alpha_k}{2}, \alpha_{k+1}]$. În realitate există şi doime cu punct, pătrime cu punct, optime cu punct şi şaisprezecime cu punct.

De aceea vom considera şi notele cu punct vom obţine că duratele de bază se află în intervale de forma $[\alpha_k - \frac{1}{8} * \alpha_k, \alpha_k + \frac{1}{4} * \alpha_k)$ iar cele cu punct în intervale de forma $[\alpha_{k+1} + \frac{1}{4} * \alpha_{k+1}, \alpha_k - \frac{1}{8} * \alpha_k)$.

Algoritmul se bazează pe faptul că duratele în timp sunt inversul puterilor lui doi. Se trece prin puterile lui 2 şi să se verifice cărui interval din cele două de mai sus aparţine durata respectivă, unde $\alpha_k = \frac{1}{2^k}$, $k \in [1, 4] \cap \mathbb{N}$.

Avem funcţii care îndepărtează elementele care sunt de interes pentru un sintetizator cum ar fi secvenţe care încep cu „CE:”, „@” şi altele. Aceste au numele funcţiei începând cu „delete[...]()”.

De asemenea în clasa PrelucrareString sunt funcţii de conversie din notaţie Staccato în notaţie LilyPond:

- „extractDeleteTime()” extrage din String-ul inițial raportul care definește fiecare măsură și-l șterge din String-ul Staccato. Un exemplu de raport ar fi $\frac{3}{8}$ care ne spune că în fiecare măsură sunt 3 optimi.
- „changeAccidentalFromStaccatoToLilypond(char c)” transformă notația accidentalelor din notație Staccato(„#”, „b”) în LilyPond(„is”, „es”).
- „changeNumberFromStaccatoToLilypond(char c)” are rolul de a transforma numărul care reprezintă octava în String-ul necesar în a doua notație. Exemplu „3” trece în „”.
- „changeStaccatoToLilyPond()” este funcția principală a clasei care le apelează pe o parte din celelalte pentru a transforma String-ul inițial în format Staccato într-o matrice de note, unde liniile reprezintă portativele iar intersecția dintre linii și coloane notele.
- „getString()” are rolul de a return String-ul în format LilyPond.

Mai avem o clasă numită Fișier cu două funcții „salveazaInFisier(String notes, String numeFisier)” și „transformaInPDF()”. Prima salvează String-ul în format Lilypond într-un fișier text iar următoarea crează un process prin rularea unui script bash, rezultatul fiind partitura dorită.

4. Prezentare mod de utilizare, interacțiune cu utilizatorul, configurare

Programul este făcut să ruleze în consola. Mai întâi utilizatorul este întrebat cum se numește fișierul pe care dorește să-l transforme în partitură. Pe următoarea linie utilizatorul introduce numele complet al fișierului, inclusiv extensia „.mid”. Acest fișier trebuie să se afle în același director cu programul sau utilizatorul va trebui să introducă adresa absolută a acestuia.

```
Cum se numeste fisierul pe care doriti sa-l transformam?
furelise.mid
|
```

Imagine 1

Fișierul rezultat va fi salvat în același folder. De asemenea va fi afișat automat cu Okular dacă utilizatorul îl are instalat,

Pentru ca programul să poată rula trebuie ca utilizatorul să instaleze anterior programul LilyPond. Acesta se poate instala folosind comenzile :

```
sudo apt-get install abcm2ps
```

sudo apt-get install lilypond

Okular se poate descărca de pe pagina lor de internet. (3)

Versiunea actuală a programului va funcționa doar pe Linux sau pe un Windows cu subsistem de Linux.

5. Concluzii în care se vor urmări aspecte legate de: obiective, utilitate,

performanta etc

Programul își propune să rezolve o problemă stringentă a compozitorilor și anume lipsa întreruperilor în procesul creației. Acest lucru poate fi atins doar prin obținerea partiturii după crearea piesei și îi scutește pe aceștia de efortul de a se opri pentru a-și scrie partitura.

Obiectivul principal este obținerea unui fișier PDF ce conține o partitura dintr-un fișier MIDI. Acest proces este complicat de imperfecțiunile umane în utilizarea unor durate de timp egale pentru durate muzicale egale.

La capitolul performanță putem vorbi de un timp foarte bun la partea de prelucrare în Java. Pentru obținerea PDF este nevoie de un timp mai lung deoarece vorbim prelucrare grafică în lipsa multor elemente necesare grăbirii prelucrării cum ar fi : lipsa notației sfârșitului de măsură, acesta fiind adăugat automat de LilyPond sau numărul de portative pe pagină, acesta fiind calculat automat de asemenea.

În ce privește acuratețea partiturii depășește unele programe. Spre exemplu voi prezenta în continuare primul portativ al piesei „Für Elise” de Ludwig Van Beethoven. Folosind programul descris în această lucrare am obținut :



Imagine 2

Iar folosind (4) am obtinut:



Imagine 3

Se poate observa că măsura a doua de pe portativul 2 obținută de pe site-ul solmire.com accesat în luna aprilie 2020 nu reprezintă o măsură posibilă deoarece dacă adunăm o pătrime cu o optime obținem echivalentul a 3 optimi, maximul posibil pentru o măsură. Acest maxim este descris de timpul descris la începutul portativului $\frac{3}{8}$. Totuși acolo mai este și o pauză, ceea ce încalcă durata măsurii. Portativul obținut de programul meu oferă valoarea corectă a notelor, având însă limitări în dreptul pauzelor.

Partitura pe care a folosit-o cel care a interpretat piesa se găsește cel mai probabil la adresa (5) .
Primul portativ se găsește în poza de mai jos.

Für Elise

WoO 59

Ludwig van Beethoven
(1770–1827)



Imagine 4

Se poate observa ca prima măsura nu este întreaga, in muzica acest lucru purtând numele de anacruza. Acest lucru este greu de intuit având doar fișierul midi la îndemâna. Ținând cont ca

scopul programului este a-l ajuta pe compozitor sa obțină schema piesei si nu varianta finala
detaliile similare celui anterior pot fi modificate ulterior creări partituri.

Bibliografie

1. *JFugue*. [Interactiv] <http://www.jfugue.org/>.
2. *MidiFile*. [Interactiv] <https://midifile.sapp.org/>.
3. developers, Okular. Download. *okular.kde.org*. [Interactiv] 11 mai 2020.
<https://okular.kde.org/download.php>.
4. [Interactiv] <https://solmire.com/miditosheetmusic/sheetmusic.php>.
5. *MuseScore*. [Interactiv] <https://musescore.com/classicman/scores/33816>.