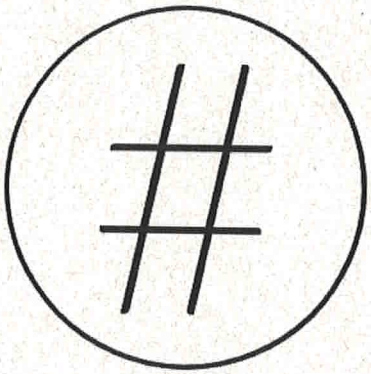
Henry Mattinson
Christ's College

Computer Science Tripos - Part II Project

Music Generation in Microsoft Excel



May 7, 2019

Declaration

I, Henry Mattinson of Christ's College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

I, Henry Mattinson of Christ's College, am content for my dissertation to be made available to the students and staff of the University.

Signed [signature]

Date [date]

Proforma

Candidate Number:	**2393G**
Project Title:	**Music Generation in Microsoft Excel**
Examination:	**Computer Science Tripos – Part II, June 2019**
Word Count:	**11745**[1]
Line Count:	**1510**[2]
Project Originator:	Prof. Alan Blackwell
Supervisor:	Dr. Advait Sarkar

Original Aims of the Project

The main aim of the project was to create a system for music expression and playback in Excel. This would allow users to play notes with defined durations at a defined tempo. Notes can be grouped to define multiple parts, play loops, and define sequences of notes and chords which can be referenced for playback. This is followed by an implementation of a converter from MIDI, an existing musical notation scheme, to the Excel system (with compression as an extension) and usability testing of the Excel system.

Work Completed

I designed a notation for music expression in Excel and built a prototype (Excello) satisfying the success criteria. Participatory design sessions with 21 users provided formative evaluation that lead to the implementation of many additional features as extensions. I contributed part of my implementation to an open-source library; this has been merged and published. I built a converter from MIDI to the Excello notation, which can convert exactly or perform compression. This was used to translate a corpus of music to the Excello notation. Finally, I performed summative evaluation with the users from the participatory design.

[1] Computed by summing `texcount -1 -utf8 -sum -inc diss.tex` using flags to including words in tables over the five main chapters.

[2] File line count for all Typescript and Python files and the JSON file for customFunctions definitions. No typescript or node configifiguration files, css, or markup included.

Special Difficulties

None.

Contents

Chapter 1

Introduction

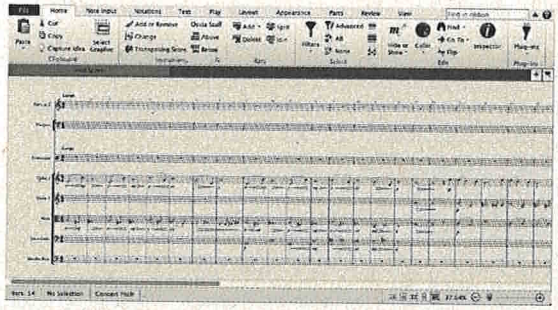
1.1 Motivation

There exist many programs for music notation and composition. In Sibelius (fig 1.1 (a)), users write scores using traditional western music notation, whilst music is produced in the live programming interface Sonic Pi (fig 1.1 (b)) by writing Ruby code [1]. These require users to gain familiarity with a new interface, often with a large threshold to creating simple musical ideas. There are four times more spreadsheet users than programmers [26], it being the preferred programming language for many people [22]. I believe that this ubiquity, along with the affordances of the spreadsheet, would enable new ways to interact with musical notation that capitalise on existing familiarities with spreadsheets and their data handling capabilities.

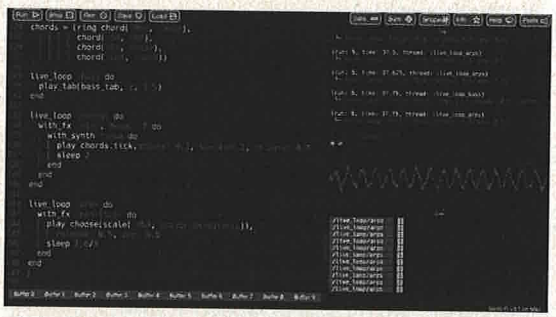
The use of grid structures is an established concept in music programs, with most sequencing software using one axis of the screen for time, and the other for pitch or musical parts. Nash's Manhattan [21] (fig 1.1 (c)) uses a grid structure where formulae can define a cell's value, like in a spreadsheet. However, it is limited to columns defining tracks and rows corresponding to time. Sarkar's SheetMusic [25] investigated including formulae with sound output within the spreadsheet paradigm. This also introduced abstracting time away from the grid, in this case using an incrementing global `tick` variable which could be referred to in formulae. Both axes can be used interchangeably for SheetMusic notation or non-musically interpreted markup that the user wishes to include, a concept idiomatic to Excel usage. Simple formulae such as `if(tick%2==0) p('snare') else p('kick')` allow musical structures to be defined without advanced programming knowledge. This plays an alternating snare and kick sound. Formulae quickly become unwieldy for larger pieces, especially if they are not highly repetitive. Whilst other spreadsheet music projects exist [13], these simply use the spreadsheet as the medium for conventional sequencing, so the spreadsheet flexibility is lost.

I have built Excello (fig 1.1 (d)), an Excel add-in for end-user music programming where users define music in the spreadsheet and can play it back from within Excel. It maintains the abstraction of time from the grid to keep the flexibility spreadsheets offer, but was designed so individual cells would not become too complex. Existing Excel functionality can be used, both accelerating the learning curve and increasing the available functionality.

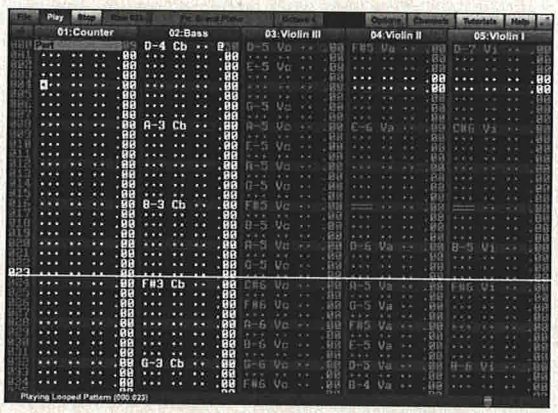
why are these nocd at the top? (7)



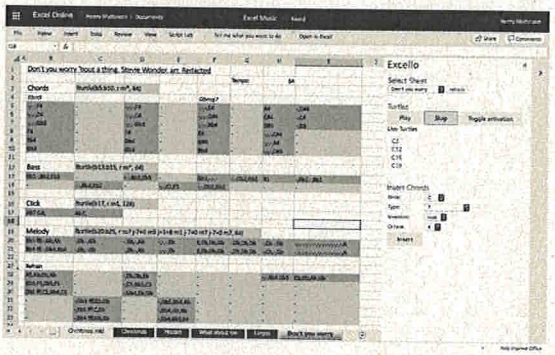
(a) Sibelius - an editor with playback for staff notation © Gorazd Bozic[1]



(b) Sonic Pi - code is written on the left and output information on the right



(c) Manhattan - columns of parts played from top to bottom



(d) Excello - a spreadsheet of music notation with a window for playback on the right

Figure 1.1: The interfaces of (a) Sibelius, (b) Sonic Pi, (c) Manhattan, (d) Excello

1.2 Outline of Work

My achievements can be summarised by:

1. I designed and built a prototype for musical expression and playback within Excel satisfying all the success criteria for the system.

2. Participatory design began using this initial prototype. Following formative evaluation sessions with 21 participants, issues and feature requests were identified. Users continued to give feedback as they used the updated prototypes.

3. A series of extensions were implemented, solving problems identified by participants and adding requested features.

4. A converter from MIDI to the Excello format was built and a large MIDI corpus translated to quantify the expressiveness of the notation.

5. Summative evaluation was performed with the participants. Evaluating the features implemented during participatory design and Excello's usability using the Cognitive Dimensions of Notation (CDN) framework [11], using Sibelius as a comparison.

not sure where this leads - where does the sentence go?

[1]https://creativecommons.org/licenses/by-nc/2.0/. (?) space dot?

Chapter 2

Preparation

This chapter first addresses the refinements made to the project proposal. ~~Then I shall~~ explain the tests that established Excel's suitability for musical development. Next, the initial design decisions for Excello ~~shall be~~ are explained. The software engineering tools and techniques employed will then be introduced. Finally, the research conducted to decide to implement the converter from MIDI to Excello is summarised.

2.1 Proposal Refinements

The project designed a notation for defining music within a spreadsheet, along with a system for interpreting this to produce audio output. This continued to explore abstracting time away from the grid.

This would be implemented as an Excel add-in subject to successful initial testing. An add-in is a web application displayed within Excel that can read and write to the spreadsheet using the Office Javascript API.[1] Arbitrary additional data and markup can be included in the spreadsheet. Tests verified that audio output can be produced for end-user music programming within Excel.

A sizeable addition to the project beyond the initial proposal was to perform participatory design [20] to advise on improvements and new features beyond the initial prototype. Participants were able to identify aspects of the current design that worked well or added cognitive difficulty. As a result, the proposed extension of live-coding (allowing notes and turtles to be modified during playback) wasn't implemented as it was not deemed high priority by the participants. Participants who gained sufficient familiarity with the project were used for more informed summative evaluation.

The converter translated MIDI to a CSV file for use with Excello. Explanation on the choice of MIDI is provided below. It was also motivated by participants who wished to integrate Excello with their use of digital audio workstations such as Logic Pro, Ableton Live, and GarageBand.

[1] https://docs.microsoft.com/en-us/javascript/api/excel?view=office-js.

2.2 Feasibility Analysis

The following section outlines the tests carried out to assess the feasibility of synthesising notes with data in a spreadsheet using an Excel add-in. All tests were carried out in Excel Online[2] using Script Lab,[3] an add-in that allows users to create and test simple add-ins experimenting with the Office Javascript API. These add-ins can access libraries and data elsewhere online.

If add-ins were run in an older version of Internet Explorer, playing sound or using the Web Audio API would not be possible [19]. An add-in that played a WAV file verified that sound could be created.

2.2.1 Note Synthesis Library

The Web Audio API allows audio to be synthesised from the browser using Javascript [19]. Creating a program for users to define and play musical structures requires synthesising arbitrary length, pitch and volume of notes. To avoid the lower-level audio components (e.g. oscillators), I researched libraries that would allow me to deal with higher level musical abstractions of the synthesised notes. Sarkar's SheetMusic used the library tones,[4] an API where only the pitch and volume envelope[5] of the notes can be defined. This also only included simple waveform synthesisers.

Tone.js[6] is a library built on top of the Web Audio API that provides greater functionality. An `Instrument` such as a `Synth` or `Sampler` is defined. The `triggerattackrelease` release method of these instruments allows a note of a given pitch, volume and duration to be triggered at a particular time. Notes are defined using scientific pitch notation (SPN) (e.g. `F#4`), name (with accidental[7] if required, e.g. `F#`) combined with octave (4). As Script Lab can reference libraries from the Node Package Manager (NPM), I tested playing notes with pitch defined in the add-in Javascript.

2.2.2 Office Javascript API

To produce music within Excel, the musical output must be informed by the data in the spreadsheet. Previous tests defined notes in the add-in Javascript. To test the Excel API, I played a note with the pitch defined in the spreadsheet. This was extended so note playing, not just the pitch, was defined within a cell (`play(F#4)`), detected and executed.

Next, I was able to play a sequence of constant length notes defined in consecutive cells (`play(H1:K1)`). The range of cells was accessed using the Excel API and the values were

[2] https://office.live.com/start/Excel.aspx

[3] https://www.microsoft.com/en-us/garage/profiles/script-lab/

[4] https://github.com/bit101/tones

[5] How the note's volume changes over its duration.

[6] https://tonejs.github.io/

[7] Sharp or flat symbol used for black notes on a piano.

played using the `Tone.Sequence` object. These tests confirmed Tone.js combined with the Excel API provided adequate functionality to implement the project as an add-in.

2.3 Excello Design and Language

2.3.1 Abstracting Time

Do you need to explain why you are discussing Al-Jazari?

Griffith's Al-Jazari [17] takes place in a three-dimensional world where robotic agents navigate around a two-dimensional grid. The height and colour of the blocks over which the agents traverse determines the sound they produce. The characteristics of the blocks are modified manually by users at run-time whilst the agents are moving. The basic instructions have the agents rotate and move forwards and backwards in the direction that they are facing. There exists a dual formalism in both the agent's instructions and the block state. This design is intended to improve live coding accessibility, both when viewing performances and becoming a live coder.

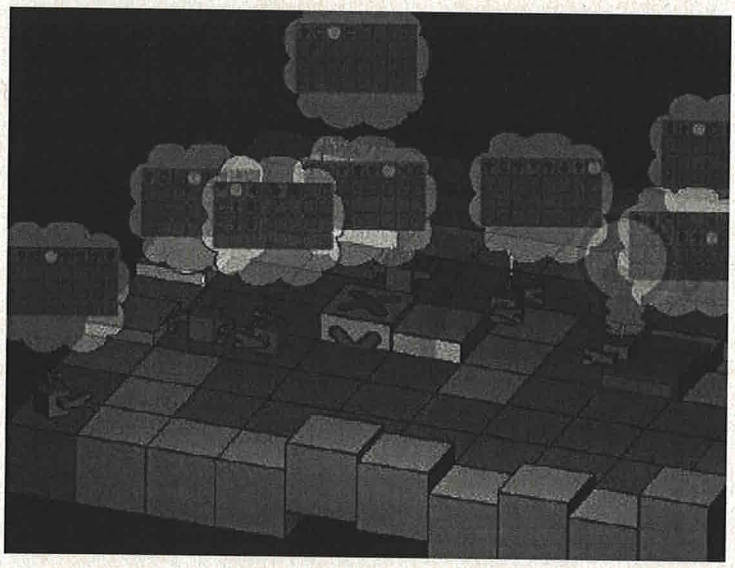


Figure 2.1: Programmed agents moving around a grid in Al-Jazari © Alex Mclean

In Al-Jazari, the agents are programmed with symbols corresponding to different movements in thought bubbles above them. This is not suitable for programming within spreadsheets where all data must exist alphanumerically within cells. If an agent was to continue moving forwards many times in a row, it would become tiresome to keep adding this symbol. This is less of an issue in Al-Jazari where the grid only measures ten cells wide and long.

Is this what happens in turtle graphics?

Having a cursor navigate around a cartesian plane is the method used by turtle graphics [27]. Just as this concept is used in Al-Jazari for agents to play the cell they occupy rather than colour it, it is suitable for spreadsheets. The turtle abstraction is employed by Excello by defining notes in cells and agents, known as turtles, to move through the spreadsheet activating them. To play a chord, multiple turtles must simultaneously pass

are the notes in cells and agents, or should it be by defining notes in cells, and agents, known as turtles, move through—?

through multiple cells corresponding to the notes of the chord. As each cell and turtle is only responsible for up to one note simultaneously, this maintains high notational consistency. However, this sacrifices the abstractions for musical structures that are available in languages like Sonic Pi - `chord('F#', 'maj7')`. By implementing methods in the add-in to add the notes of chords to the grid, abstractions can be maintained whilst preserving consistency and cleanness in the spreadsheet itself.

Turtle are the crux of the Logo programming language [10], where turtles are programmed entirely by text to produce graphical output. For example, `repeat 4 [forward 5 right 90]` has a turtle move forwards 50 units and turn 90 degrees to the right four times to draw a square. A similar method is employed in Excello but the language is designed to be less verbose.

2.3.2 Initial Prototype Design

Notes are placed in the cells of the spreadsheet and turtles are defined to move through the grid using a language based on Logo. The notes in cells are played when turtles move through them. When the play button is pressed, the melodic lines produced by all turtles defined in the grid are played concurrently. Turtles are defined with a start cell, movement instructions, the speed they move through the grid at (cells per minute) and the number of times they repeat their path. As in Al-Jazari, distance in space maps to time [18]. Excello extends upon this as turtles can move at different speeds. Therefore parts with longer notes can be defined more concisely and phase music[8] can be easily defined. The Excello turtle movement instructions are explained below with examples.

Turtles begin facing north (towards the top of the screen). The move command, `m`, moves the turtle one cell forward. Like Logo, turtles always move in the direction they are facing. The commands `l` and `r` turn the turtle 90 degrees left and right respectively. Logo commands are repeated with `repeat` followed by the number of repeats and the commands [10]. To reduce verbosity, single commands are repeated by placing a number immediately after it. For example, `m4` has the turtle move forwards four cells in the direction that it is facing. The direction a turtle is facing can be defined absolutely with `n`, `e`, `s` and `w` to face the turtle north, east, south and west. These rotate the turtles rather than moving them to maintain the consistency that turtles always move in the direction they face. To change notes' volume, dynamics (`ppp`, `pp`, `p`, `mp`, `mf`, `f`, `ff`, `fff`) can be placed within the turtle instructions. Any notes played after are played at that dynamic. Just as dynamics in western notation are a property of the staff, not individual notes, dynamics were originally defined in the turtle, not notes. To repeat multiple instructions, they are placed in brackets followed by the number of repeats. For example, `(r m5)4` defines a clockwise path around a five by five square. This seven character example is equivalent to the above Logo example, which requires 29 characters. This repetition of instruction is why relative movements `l` and `r` are included in the language despite being less explicit than the compass based directions. This is demonstrated in Figure 2.2.

[8] Music where identical parts are played at different speeds.

	A	B	C	D
1	turtle(A2, e m3 (r m2)2 m)			
2	C4	D4	E4	F4
3				G4
4	C4	C5	B4	A4

Figure 2.2: The instructions for a turtle following the path of notes from A2 to A4

Constraining each melodic line to a single path of adjacent cells may be inconvenient. Just as conventional score notation often spans across multiple lines, splitting up parts is a useful form of secondary notation (notation that is not formally interpreted). This requires the turtle to move to non-adjacent cells. For graphic drawing in Logo, lifting the pen allows the turtle to move without colouring the space beneath it. The number of steps the turtle takes does not affect the output, only what it colours does. However, Excello's musical output is dependent on the turtle's spatio-temporal information, so this would introduce large rests. Analogous to lifting the pen for graphical turtles, the turtle could enter a mode where it passes through cells immediately without playing them. But as the path in this case would be insignificant, I have added jumps to the language so the cell where play resumes can be given without having to program a path to that point. This is defined with `j` in absolute terms with a destination cell (e.g. `jA5`), or relatively (e.g. `j-7+1`), with the number of rows and columns jumped. An absolute jump may be more explicit for human readers, but relative jumps facilitate repeats jumping to different cells each time. For example, `r(m7 j-7+1)9 m7` plays 10 rows of 8 cells from top to bottom playing each row left to right. A jump is demonstrated in Figure 2.3.

	A	B	C	D
1	turtle(A2, r m3 jA4 m3)			
2	C4	D4	E4	F4
3				
4	G4	D4	E4	C4

Figure 2.3: The instructions for the path from A2 to D2 then A4 to D4

The language for turtle movement instructions is summarised by the following context-free grammar, $(N, \Sigma, S, \mathcal{P})$:

- Non-terminal symbols N = (⟨Full Instructions⟩, ⟨Instruction Series⟩, ⟨Single Block⟩, ⟨Command⟩, ⟨Relative⟩, ⟨Absolute⟩, ⟨Sign⟩, ⟨Cell⟩, ⟨Dynamic ⟩)
- Terminal symbols Σ = ($z \in \mathbb{Z}$, $n \in \mathbb{N}$, $c \in$ `[A-Za-z]`$^+$, `m`, `j`, `l`, `r`, `n`, `e`, `s`, `w`, `+`, `-`, `ppp`, `pp`, `p`, `mp`, `mf`, `f`, `ff`, `fff`)
- Starting symbol S = ⟨Full Instructions⟩
- Grammar rules $\mathcal{P}$ are shown in Figure 2.4

for my info: what does this mean?

⟨Full Instructions⟩ ::= ⟨Instruction Series⟩
⟨Instruction Series⟩ ::= ⟨Single Block⟩ | ⟨Single Block⟩⟨Instruction Series⟩
⟨Single Block⟩ ::= (⟨Instruction Series⟩)n | ⟨Command⟩
⟨Command⟩ ::= `m`z | ⟨Relative⟩ | ⟨Relative⟩z | ⟨Absolute⟩ | ⟨Dynamic⟩
| `j`⟨Cell⟩ | `j`⟨Sign⟩n⟨Sign⟩n
⟨Relative⟩ ::= `l` | `r`
⟨Movement⟩ ::= `n` | `e` | `s` | `w`
⟨Sign⟩ ::= `+` | `-`
⟨Cell⟩ ::= cz
⟨Dynamic⟩ ::= `ppp` | `pp` | `p` | `mp` | `mf` | `f` | `ff` | `fff`

Figure 2.4: Turtle movement instructions grammar rules, $\mathcal{P}$, in Backus-Naur form [16]

Notes are defined in the cells using SPN. Empty cells are interpreted as rests. To create notes longer than a single cell, a cell containing only the character `s` sustains the note that came before it as in Figure 2.5 (a). A cell can be subdivided time-wise into multiple notes with multiple comma separated notes as in Figure 2.5 (b). This was designed so the length a cell corresponds to is not bound by the length of the smallest note in the piece. For example, a piece defined primarily with crotchets (one unit) but with occasional quavers (half a unit) does not necessitate double the number of cells and introduce many additional `s` cells in the entire piece.

are these notes equal length?

clever

	A	B	C	D	E	F	G	H
1	C4	s	D4	s	E4	s	F4	G4

(a) Using sustain cells

	A	B	C	D
1	C4	D4	E4	F4,G4

(b) Using a multi-note cell

Figure 2.5: Two identical motifs defined by using sustains or with multi-note cells

2.4 Software Engineering

2.4.1 Requirements

The success criteria of the project are:

1. Implementation of an system for music playback within a spreadsheet where users can:
   - Play individual notes and chords with defind durations.
   - Define multiple parts.
   - Play loops.
   - Define sequences of notes and chords and be able to call these for playback.
   - Define the tempo of playback.

2. Participatory design with formative evaluation sessions.
3. Summative evaluation with participants who have gained familiarity with the system.
4. Implementation of a converter from MIDI to the spreadsheet representation.
5. In addition to these, the following extension work was completed:
   - Implement additional features that arise from participatory design.
   - Explore a compressive conversion from MIDI to the Excel system.

2.4.2 Tools and Technologies Used

Table 2.1 outlines the tools, languages and libraries used.

Software	Type	Usage
Scriptlab	Add-in	Writing initial Excel add-in tests.
Typescript	Language	Writing Excello. Used for static type checking and Javascript libraries.
Yeoman[9]	Add-in Generator	Creating the blank Excel add-in project.
NodeJS	Javascript Environment	Manage library dependecies and run local web servers.
Surge[10]	Content Delivery Network	Hosting Excello online for participants' use.
Jupyter Notebook	Python Environment	Implementing the MIDI to Excello converter.
Tone.js	Library	Synthesising and scheduling sound via the Web Audio API.
tonal	Library	Generating the notes of chords.
Mido[11]	Library	Reading MIDI files in Python.

Table 2.1: Tools used during the project

2.4.3 Starting Point

Having used the Yeoman generator to create an empty Excel add-in, all the code to produce Excello and the MIDI converter was produced from scratch using the libraries described above.

I had written simple Javascript for small websites, but had no experience using Node, libraries or building a larger project. Having never used any of the libraries before, reviewing the documentation was required before and during development. I had gained experience with Python and Jupyter Notebooks from a summer internship.

[9] A generator for scaffolding Node.js web applications, https://github.com/OfficeDev/generator-office.
[10] Static webpage publishing tool and hosting https://surge.sh/.
[11] https://mido.readthedocs.io/en/latest/.

2.4.4 Evaluation Practices

To best tune Excello's to the needs of potential users, formative evaluation sessions were carried out with participants. A spiral development methodology [6] was used. This iterates the following steps: determining objectives, identify problems and solutions, develop and test, deploy and prepare next iteration. This was more suitable than sequential, document-driven process methods such as the waterfall method. Due to the number of participants and the project timeframe, there were only two major development iterations with additional incremental updates. The first prototype, and the second fixing issues and implementing requests from participants.

Summative evaluation was carried out with users involved in participatory design. Therefore, experienced users of Excello could be used for evaluation despite the product not yet being released in the public domain. This includes analysis using the CDN framework - a tool for reasoning about cognitively relevant system attributes [11].

2.5 MIDI files

MIDI [2] is a communications protocol to connect electronic musical instruments with devices for playing, editing and recording music. A MIDI file consists of event messages describing on/off triggerings to control audio [15]. MIDI files were designed to be produced by MIDI controllers such as electric keyboards. As such, MIDI files contain controller-specific information that is not necessary for the creation of an Excello file. Musical formats such as MusicXML specify the musical notation and could be suitable for conversion to Excello.

Many programs support importing and exporting MIDI files. By converting MIDI files to the Excello notation, Excello is more integrated into the environment of programs for playing, editing and composing music. Furthermore, there exist many datasets available for MIDI [14], which can immediately be played back for comparison.