Excello: exploring spreadsheets for music composition

Henry Mattinson
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue, Cambridge, UK
henrymattinson@westfieldhouse.org

Advait Sarkar [1,2]
[1]Microsoft Research
21 Station Road, Cambridge, UK
[2]University of Cambridge
advait@microsoft.com

ABSTRACT

Excello is a spreadsheet-based music composition and programming environment. We co-developed Excello with formative and summative feedback from 21 musicians at varying levels of musical and computing experience. We asked: can the spreadsheet interface be used for programmatic music creation? Our design process encountered questions such as how time should be represented, whether amplitude and octave be encoded as properties of individual notes or entire phrases, and how best to leverage standard spreadsheet features, such as formulae and copy-paste. We present the user-centric rationale for the design we arrived at, as well as a user study suggesting that Excello's notation retains similar cognitive dimensions to conventional music composition tools, while allowing the user to write substantially complex programmatic music.

Author Keywords

Spreadsheets, end-user programming, notation, cognitive dimensions

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; •**Information systems** → *Music retrieval;*

1. INTRODUCTION

Programmatic environments for music creation, such as ChucK [15] or Sonic Pi [1], enable the creation of complex and dynamic compositions with rich and interactive feedback. However, these systems are extremely hard to learn, especially for users new to programming.

On the other hand, spreadsheets are a well-known and easy to learn programming environment. The 2-dimensional grid, along with support for computation, annotation, and visualization, forms the basis for the world's most ubiquitous non-expert end-user programming environment. There are four times more spreadsheet users than software developers [14], and spreadsheets are the preferred programming language for many people [9]. This ubiquity, along with the affordances of the spreadsheet, enables new ways to interact with musical notation that capitalise on existing familiarity with spreadsheets and their data handling capabilities.

We present Excello (Fig. 1, shown here in use with a study participant's arrangement), an Excel add-in for end-user music programming. The Excello add-in opens a pane on the right side of Excel. The user defines notes and 'turtles' in the cells of the spreadsheet. Turtles are programmable playheads that move through the grid using a simple instruction language. 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 using a piano sound.

2. EXCELLO DESIGN

2.1 Abstracting time with turtles

The spreadsheet's chief advantage is its 2-dimensional grid, which allows the end-user to spatially organise their computations and data. Many music composition environments also use grid structure, albeit in a limited fashion. For example, MIDI sequencers [8] typically use the horizontal axis for time, and the vertical axis for pitch or musical parts. Manhattan [12] uses a grid 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. Similarly, other spreadsheet music projects[1] only use the spreadsheet grid with the conventional sequencing layout, losing the flexibility of using both grid axes.

SheetMusic [13] investigated formulae with sound output within the spreadsheet paradigm. SheetMusic abstracts time away from the grid using an incrementing global `tick` variable which could be referred to in formulae. Both axes can be used interchangeably for SheetMusic notation or non-musical markup (e.g., data, labels, formatting), a concept idiomatic to spreadsheets. Music is notated with formulae such as: `if(tick%2==0) p('snare') else p('kick')`, which plays an alternating snare and kick sound. However, such formulae quickly become unwieldy for larger pieces, especially if they are not highly repetitive.

How do we compactly represent time without sacrificing a grid axis? To solve this problem, we apply the metaphor of turtle graphics [5]. In the Logo programming language, agents known as 'turtles' are programmed 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.

[1]`https://hackaday.com/2019/02/02/never-mind-the-sheet-music-heres-spreadsheet-music/`

NIME'20, July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

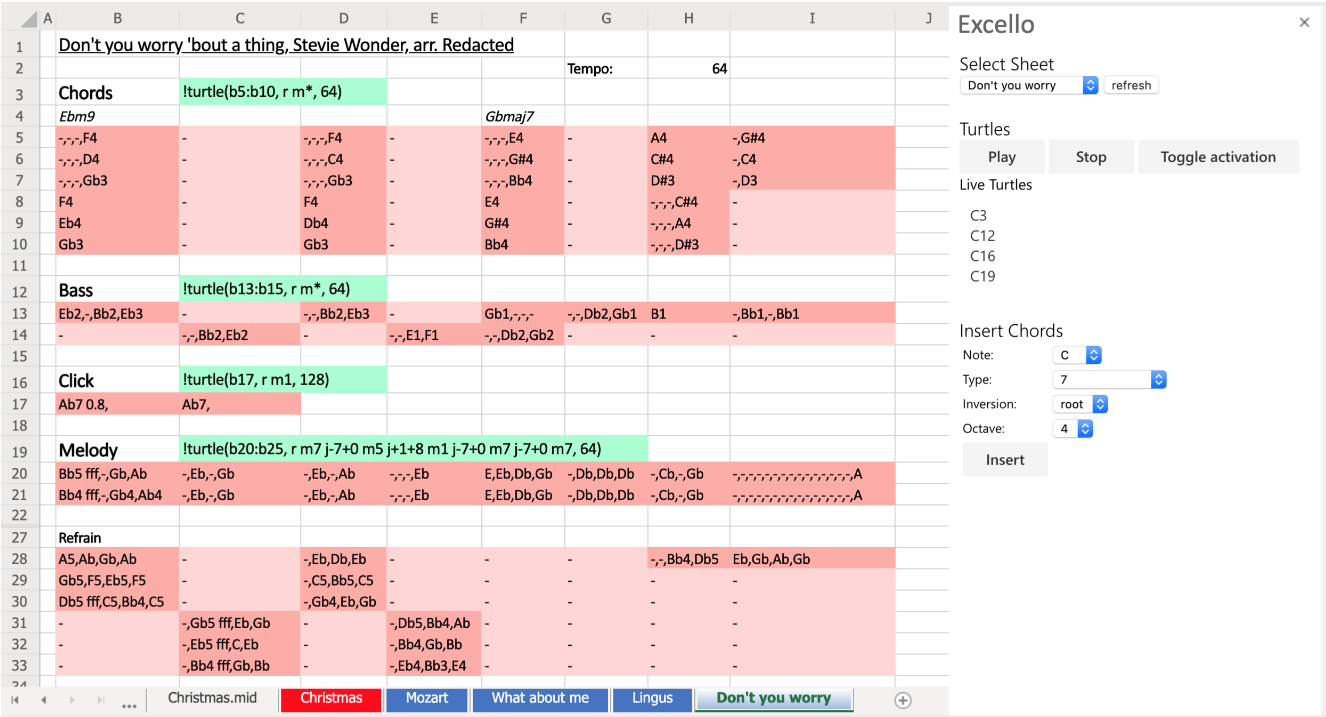

Figure 1: An arrangement with separated and labelled parts per instrument. Turtles use a global tempo in cell `H2`.

The turtle abstraction is employed by Excello by defining notes in cells, and agents, known as turtles, move through the spreadsheet playing them. Al-Jazari [10] also uses the turtle metaphor, in a limited fashion. In Al-Jazari, robotic agents navigate around a two-dimensional grid. Distance in space maps to time [11]. Excello extends this, as turtles can move at different speeds. Therefore parts with varying speeds, and phase music (where identical parts are played concurrently at different speeds) can be defined more concisely. Moreover, Al-Jazari's agents are programmed with movement symbols in thought bubbles above them. This is unlike spreadsheets where both data and computation logic exist in the same grid. Al-Jazari's grid only measures ten cells wide and long, which greatly simplifies and constrains agents' movements.

2.2 Design process

Excello's design is grounded in concrete user needs and feedback. Twenty-one University of Cambridge students across a range of subjects, took part in the participatory design process. We conducted initial feedback sessions as follows: one-on-one tutorials on the initial prototype were given, followed by a short exercise of the participants' choice; either transcribing a piece from memory or from staff notation into the Excello notation, or modifying and extending a composition already written in Excello. Next, users were interviewed about their experience, drawing particular attention to actions that they found unintuitive or requiring notable mental effort. Comparisons were made to musical interfaces with which participants were already familiar.

These sessions were conducted in January 2019. Participants continued using Excello for the next 7-8 weeks, until the summative evaluation sessions in March, ensuring that the evaluation was conducted on participants with significant experience using Excello. Additional feedback was collected as participants used Excello in their own time.

In the next section, we explain and describe some of the interesting and important design issues that arose in consultation with our participants.

2.3 Encoding musical notes

Notes are written using scientific pitch notation (SPN). Empty cells and the full stop character (`.`) denote rests. The character `s` or `-` instructs the turtle to sustain the previously played note, and a cell can be subdivided using commas into multiple equal length notes (Figure 2). The combination of sustains and subdivisions allows the composer to choose the most convenient note duration to correspond to each cell. Without subdivision, a piece defined primarily with crotchets (one unit) but with occasional quavers (half a unit) would need twice as many cells and many additional `s` cells.

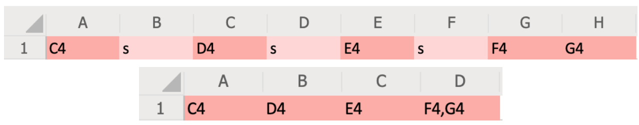

Figure 2: Two identical phrases, defined (above) by using sustains or (below) with subdivided cells.

Octave numbers can be omitted for brevity; during testing we discovered that repeatedly writing the octave number was tiresome. If omitted, we use the last explictly notated octave. We tested two methods for octave inference: playing the note in same octave as the previous note, or choosing the octave in which the note would be nearest to the previous note. Both have advantages and disadvantages. The nearest-octave approach may require the fewest explicit statements of octave numbers, but it is harder for a reader to immediately identify the octave of any given note;

Concept	Encoding
Note	Name (`A-G`), optional accidental, octave number and dynamics e.g. `F♯4 pp`
Sustain	`s` or `-`
Time-subdivided notes	Notes, rests or sustains separated by a comma. Rests must be a space or an empty string e.g. `E4, ,C4,s`
Rest	Any cell not interpreted as a note, sustain or multi-note. Rests can be explicitly denoted with '`.`'

Table 1: Summary of note encoding.

they would need to locate the last explicit octave notation and walk through subsequent notes, keeping track of the inferred octave. The same-octave approach may require many octave definitions if a melody frequently crosses the boundary between octaves, but it is much easier for the reader to identify the octave of a note by backtracking, and this was the balance we found to be better for our users.

Dynamics Just as dynamics in western notation are a property of the staff, not of individual notes, dynamics were originally defined in the turtle, not in notes, using the symbols `pp p mp mf f ff` (etc.) next to turtle movement commands. However, users found it hard to read the turtle's path, since dynamics were unrelated to movement. Furthermore, as dynamics weren't next to the notes to which they corresponded, knowing the volume of a note or where to place the dynamics within the turtle to apply to notes in the spreadsheet was challenging.

Thus, we settled on dynamics being defined in the cells after the note, separated by a space as in Manhattan [12]. In addition to Western dynamic symbols, a number between 0 (silent) and 1 (equivalent to `fff`) can be used. Like octave numbers (and indeed, like staff notation), a dynamics specification applies to all following notes until the turtle encounters another explicit specification. Table 1 summarises our note encoding.

2.4 Encoding the turtle's path

The following formula-like syntax defines a turtle:

```
!turtle(Start_Cell, Instructions, Tempo, Loops)
```

The prefix "`!`" signals that the turtle is active; omitting this prefix causes the turtle not to play, analogous to muting and soloing in digital audio workstations / music sequencers.

Instead of typing this text directly, the user can also define a turtle using the formula function `EXCELLO.TURTLE`, enabling users to leverage the built-in autocompletion and cell referencing features of Excel. The output of this function is our textual turtle notation (Figure 4).

Start Cell The turtle's starting cell (`A2` in Figure 3),

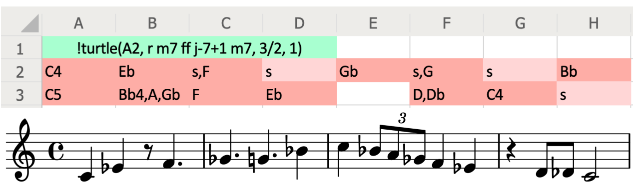

Figure 3: A short melody in an early prototype (above) with its output in staff notation (below).

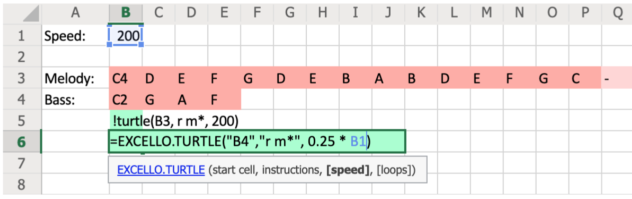

Figure 4: Defining a turtle using `EXCELLO.TURTLE`.

which is also played, is a cell reference (a concatenation of letters for the column and numbers for the row). As each turtle only plays one note at a time, multiple turtles must be defined for polyphony. A common user pattern was to define turtles following identical paths but in adjacent rows or columns. To simplify this process, we made it possible to instantiate multiple turtles using the existing Excel range notation. Setting the starting cell to `A2:A5` defines four turtles in the cells `A2,A3,A4,A5`. This removes the need for multiple turtle definitions differing in only the start cell.

Tempo An optional third argument is the speed of the turtle in cells per minute (the default is 160). An early implementation required this to be the multiple of 160 (thus 1 corresponds to 160 cells per minute, 0.5 corresponds to 80 cells per minute, etc.), so that it would be easier to tell the speed relation between turtles. This particularly suits phase music. However, participants felt calculating this speed factor was effortful and unnatural so we switched to cells per minute. Luckily, in a spreadsheet environment, it is still straightforward to implement relative tempi using formulae: turtles could reference a single cell containing a 'base' speed, and apply a turtle-specific multiplier.

Loops An optional fourth argument defines the number of repetitions of the turtle's entire path (e.g., the turtle in Figure 3 plays 1 time). By default, turtles loop infinitely.

Turtle motion Turtles begin facing north (towards the top of the screen). 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, and commands `n`, `e`, `s` and `w` directly re-orient the turtle north, east, south and west. The command `m` moves the turtle one cell forward. Commands are repeated by placing a number immediately after it (this is less verbose than Logo's notation: `repeat` followed by the number of repeats and the commands [5]). Thus, `m4` moves the turtle forwards four cells in the direction it faces. Commands can be nested within parentheses, and nested commands can be repeated. Thus, `(r m5)4` defines a clockwise path around a five-by-five square. Nested instructions with repeats allow concise notation of repeated sections and movements.

Just as conventional staff notation spans multiple lines, splitting melodies into parts spanning multiple rows is a useful layout for human readers. This requires the turtle to move to non-adjacent cells. In Logo, lifting the pen allows the turtle to move without drawing a line; the graphical output is unaffected by the turtle's path in 'pen-up' mode. However, Excello's musical output depends on the turtles' movements in time, so a 'pen-up/pen-down' metaphor would introduce large rests as the turtle moved to its destination. Thus, our language supports jumps with `j`. Jumps are either absolute, with a destination cell (e.g., `jA5`), or relative (e.g., `j-7+1`), with a row-column offset. Rela-

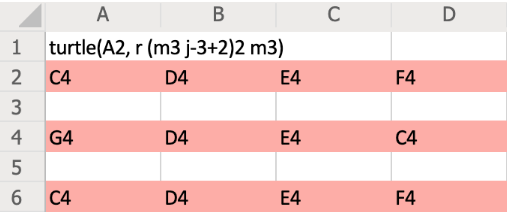

Figure 5: Turtle with repeating jump instructions.

tive jumps enable concise patterns. For example, `r (m3 j-2+2)2 m3` plays 3 rows of 4 cells from top to bottom, playing each row left to right (Figure 5).

Automatic path length counting Writing turtle instructions requires counting cells on the grid. If a sequence of notes is in a straight line, the user can select the cells and see a count in the Excel status bar. However, this requires manual effort and is error-prone, and users found it particularly inconvenient when adding notes to a partially complete line and periodically testing the composition written so far. Some users instructed turtles to move forward significantly more steps than required to avoid counting steps, but this strategy doesn't work for repeating paths.

We thus implemented `m*`, which instructs a turtle to move as far as there are notes defined in the direction it is facing. After adding notes to the end of a line, the turtle instructions do not need editing before pressing play. A cell can be explicitly defined as a rest with a full stop (`.`). This is required if multiple turtles are playing a repeating section where some turtles end on rests, and others on notes. Without an explicit rest, the turtle would repeat too soon and the parts would consequently be out of alignment.

2.5 Spreadsheet affordances

Highlighting To provide visual structure, turtle definitions are automatically highlighted green. Cells containing definitions of notes, or multiple notes, are highlighted red. Sustain cells are highlighted a lighter red, showing correspondence to notes whilst maintaining differentiation.

Chord Input To play a chord, multiple turtles must simultaneously pass through multiple cells corresponding to the notes of the chord. Each cell and turtle is only responsible for up to one note at a time, maintaining high notational consistency. However, this sacrifices the chord abstractions in languages like Sonic Pi (e.g., `chord('F♯', 'maj7')`). Our solution was to include a tool for adding chords (right sidebar of Fig. 1). The user selects the chord root, type, inversion and starting octave from menus. The insert button enters the notes of the chord into the grid where the user has made a selection. Notes are inserted to "fit" the shape of the selection (vertical or horizontal), and for a vertical selection, notes are inserted from top to bottom in decreasing pitch order, to mimic staff notation (based on user feedback).

Transpose Some users found it more intuitive to consider a melodic line by the intervals between notes rather than by the note names. Moreover, many users sought to define harmony lines by a transposition from a melody line.

Thus, a transposition function (`EXCELLO.MODULATE`) lets melodic lines be defined by the intervals between notes and transposition of existing sections of a piece. The function takes a cell and an interval and outputs the cell with any notes transposed by the interval, maintaining any dynamics.

The advantage of implementing this (and `EXCELLO.TURTLE`) as spreadsheet functions is that it allows users to take advantage of functionality such as drag-fill, formula autocomplete, graphical cell referencing, etc. For example, a section can be modulated by calling this function on the first note with a provided interval and then using spreadsheet "drag-fill". A melodic line can be produced from a starting note and a series of intervals as shown in Figure 6.

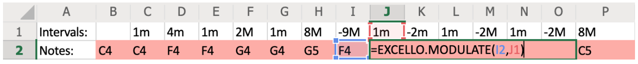

Figure 6: Transposing notes using `EXCELLO.MODULATE`

2.6 Example: piano phase

The first section of Reich's Piano Phase is two identical piano melodies, one played slightly faster than the other [3]. The parts move out of phase, periodically aligning at different offsets. Manhattan implements it using three rows of 24 columns [12]. Sonic Pi requires one line for the notes and eight for playback. Excello only requires two cells to define two turtles with different speeds in addition to the notes. All three implementations are shown in Figure 7.

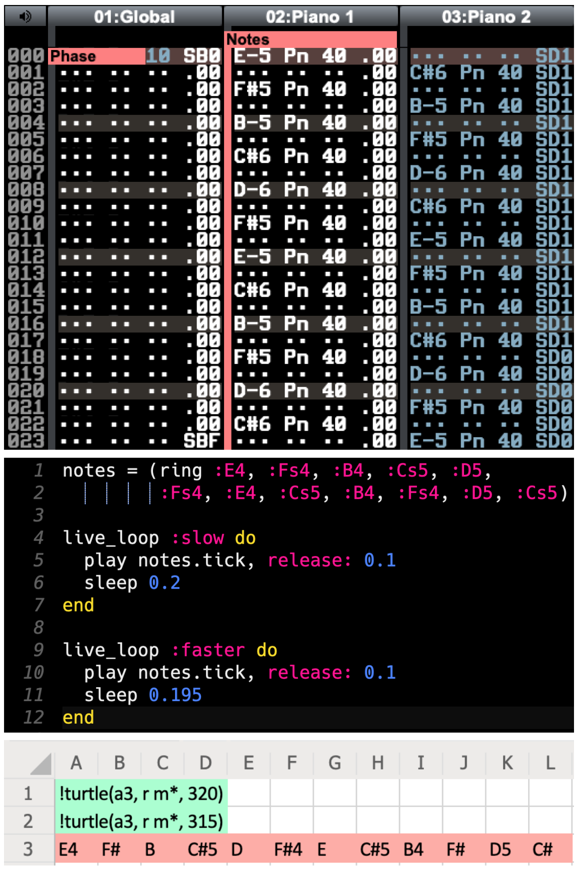

Figure 7: Piano Phase in (top to bottom) Manhattan, Sonic Pi, Excello

2.7 Implementation details

We implemented Excello as an add-in using the Office.js API.[2] When the play button is pressed, turtle definitions in the grid are identified. For each, the starting cell and movement instructions are used to establish the contents of the cells it passes through. This is converted to a series of note definitions - pitch, start time, duration, volume. These are in turn passed to the the Tone.js library[3] to schedule and initiate playback.

3. SUMMATIVE EVALUATION SESSIONS

Of the 21 initial participants, 19 continued using Excello after formative evaluation sessions and participated in a summative evaluation session. To ensure users sufficiently understood the interface before giving feedback, features added after the initial sessions were recapped, and participants completed a short task requiring transcription of a short melody and authoring an additional phrase.

To study the properties of our system, we applied Blackwell and Green's questionnaire [2] for evaluating information devices' usability using the Cognitive Dimensions of Notations (CDN) framework. For example, the dimension *Role Expressiveness* (how much an element suggests its purpose) is assessed by: "*Are there some parts that are particularly difficult to interpret?*". CDN can be used to analyse musical notation [4] and software systems [7], so is ideal for evaluating Excello's notation and interface.

We focused on closeness of mapping, consistency, secondary notation, viscosity and visibility. The questions used are shown in Table 9. Users responded with a five-point Likert scale. Responses were combined into negative and non-negative categories. The significance of the results was verified with chi-squared tests. Chi-squared test p-value and modal responses are shown in Table 2. The distribution of responses is shown in Fig. 9.

3.1 Comparative evaluation

By way of comparison, CDN results were also collected for the user's preferred music composition interface. 12 users chose Sibelius, which was used for comparison. We think this is a better comparison than to ChucK or Sonic Pi, since Excello is designed to be an accessible introduction to music programming for musicians familiar with more conventional notations.

The significance of each dimensions varies for different cognitive activities [6], so users identified the percentage of time they spent carrying out these activities (searching for information, translating, incrementation, modification and exploratory design). Figure 8 shows the time users reported spending on the different cognitive activities in Excello and in Sibelius, based on results from 19 users for Excello and 12 for Sibelius. Translation is important for both interfaces. Users perceived spending more time on modification and incrementation in Excello. Little time is spent searching in either tool.

We compared participants' answers to these questions for Excello and Sibelius. We performed a Wilcoxon matched pairs sign-rank test on the 12 pairs by encoding the five

[2]`https://docs.microsoft.com/en-us/javascript/api/excel`
[3]`https://tonejs.github.io/`

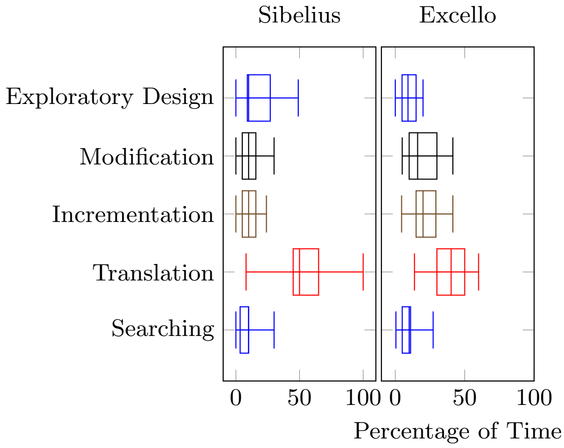

Figure 8: The percentage of time spent performing the different cognitive activities.

Statement	CDN	Mode	p
(a) The notation used (In Excello: notes/ dynamics in cells and the definition of turtles) is related to the result you are describing (In Excello: Musical output)	Closeness of Mapping	Agree	0.0004
(b) Where there are different parts of the notation that mean similar things, the similarity is clear from the way they appear.	Consistency	Agree	0.0087
(c) You can add extra marks (or colours or format choices) to clarify, emphasise or repeat what is there already.	Secondary Notation	Agree	0.0020
(d) When you need to make changes to previous, work it is easy to make the change.	Viscosity	Agree	0.0004
(e) It is easy to see or find the various parts of the notation while it is being created or changed.	Visibility/ Juxtaposition	Agree	0.0087
(f) If you need to compare or combine different parts, you can see them at the same time.	Visibility/ Juxtaposition	Agree	0.0312

Table 2: CDN of Excello: questions and results.

responses as -2,-1,0,1,2. For all six questions, there is no indication that the answers for the two interfaces come from populations with different means (i.e., no significant differences).

Closeness of mapping We found no significant difference between Sibelius and Excello, suggesting Excello's notation with spreadsheets has not compromised the closeness of mapping of staff notation.

Consistency We found no significant difference between

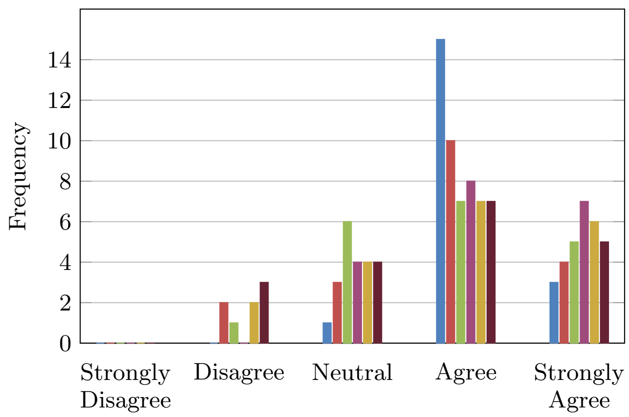

Figure 9: CDN of Excello: distribution of responses (colour legend from Table 2).

Sibelius and Excello, suggesting that Excello's notation is no less consistent than staff notation. Each cell and turtle only produces one note at a time. Excello keeps consistency with Excel by sharing notations (e.g. A1:A5 for ranges) and using the existing formula editor.

Secondary notation We found no significant difference between Sibelius and Excello, suggesting that the spreadsheet paradigm can provide secondary notation abilities similar to Sibelius, software already equipped with numerous ways to customise a score. Given the time spent translating, secondary notation is particularly important [4]. As Excello abstracts time from the grid axes, existing Excel features for formatting and grouping cells remain available.

Viscosity We found no significant difference. This suggests the interfaces have comparable viscosity. Allowing dynamics and octave marking to be omitted and letting turtles count steps automatically, provides low resistance to making additions and changes to the music. Furthermore, Excel provides easy editing and movement of cells.

Visibility/Juxtaposition We found no significant difference. This suggests that the spreadsheet interface can provide a similar ability to view components as Sibelius.

4. CONCLUSION

We set out to explore the hypothesis that spreadsheets would provide a productive medium for musical expression. Excello is a notation and corresponding program for musical playback implemented within Microsoft Excel. By abstracting time away from the axes of the grid, the existing functionality of Excel (whitespace, formatting, layout, copy-paste and data storage) remains highly useful. Excello was developed in close consultation with a group of 21 users, as a result of this, many nuances in notational design were discovered, and our solutions were shown to improve the interface and make it competitive (on cognitive terms) with conventional composition tools.

5. ETHICAL COMPLIANCE

Our study was approved by the Cambridge University computer science department ethics review board. We conducted pilot studies of the formative and summative evaluation sessions, resulting in revisions to the protocol. Participants were briefed, and signed forms of informed consent. Participant data was anonymised and any audio recorded during the sessions was transcribed and deleted.

6. ACKNOWLEDGEMENTS

We thank Alan Blackwell for discussions and our study particpants for their time and effort.

7. REFERENCES

[1] S. Aaron. Sonic pi–performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, 12(2):171–178, 2016.

[2] A. F. Blackwell and T. R. G. Green. A Cognitive Dimensions questionnaire optimised for users. In *PPIG*, 2000.

[3] P. Epstein. Pattern Structure and Process in Steve Reich's "Piano Phase". *The Musical Quarterly*, 72(4):494–502, 1986.

[4] A. F. Blackwell, T. Green, and D. Nunn. Cognitive Dimensions and Musical Notation Systems. *Workshop on Notation and Music Information Retrieval*, 11 2000.

[5] R. Goldman, S. Schaefer, and T. Ju. Turtle geometry in computer graphics and computer-aided design. *Computer-Aided Design*, 36:1471–1482, 2004.

[6] T. Green and A. Blackwell. Cognitive dimensions of information artefacts: a tutorial. Technical Report Version 1.2, BCS HCI Conference, 1998.

[7] T. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages*, 7:131–, 06 1996.

[8] D. Hosken. *An introduction to music technology.* Routledge, 2014.

[9] S. P. Jones, A. Blackwell, and M. Burnett. A user-centred approach to functions in excel. In *Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, pages 165–176, 2003.

[10] A. McLean, D. Griffiths, N. Collins, and G. Wiggins. Visualisation of live code. *Electronic Visualisation and the Arts (EVA 2010)*, pages 26–30, 2010.

[11] A. McLean and G. A. Wiggins. Texture: Visual notation for live coding of pattern. In *ICMC*, 2011.

[12] C. Nash. Manhattan: End-User Programming for Music. In *NIME*, 2014.

[13] A. Sarkar. Towards spreadsheet tools for end-user music programming. In *Psychology of Programming Interest Group (PPIG)*, pages 228–231, Sept. 2016.

[14] C. Scaffidi, M. Shaw, and B. Myers. Estimating the numbers of end users and end user programmers. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, pages 207–214, Sep. 2005.

[15] G. Wang, P. R. Cook, and S. Salazar. Chuck: A strongly timed computer music language. *Computer Music Journal*, 39(4):10–29, 2015.