Henry Mattinson
Christ's
HCM50

Computer Science Part II Project Proposal

Music Generation in Microsoft Excel

10/10/2018

Introduction

Calculation view is an extension to Microsoft Excel that was announced in a conference on October 2nd 2018 by Advait Sarkar from Microsoft Research [1]. Its purpose is to provide an alternative representation of the spreadsheet as a textual program. This project would explore using this new functionality towards end-user music programming in Excel as in Advait's SheetMusic [2].

The aim of this project will be to convert an existing musical format such as MusicXML to the Calculation View language so that the music can be represented in a spreadsheet format. By using a Typescript version of Excel run in browser, music playback could then be achieved from within Excel. This could then be extended to have application in live music coding.

Starting Point

No existing work or further knowledge than part Ia and Ib courses.

I will be building on top of existing Microsoft products but I have not been able to discuss the details of this with my supervisor. For example, I will not need to reimplement Excel.

Substance and Structure of Project

Most important is the implementation of an API for music generation. A version of Excel built in Typescript can be used, therefore, it can be run from the browser. Playback functionality can be built on top of the web audio API. Functionality for note and sequence synthesis functions will be required. Additional library functionality for notes of chords and scales could also be added to help reduce the complexity of the representation within the spreadsheet.

As outlined in Advait's SheetMusic paper, a method for representing time will be required. Time should elapse independently of the layout of the spreadsheet. Cells either recalculate on a publish/subscribe basis, or polling basis. This is implemented in Excel's Real Time Data feature. I suspect this will be used to facilitate timing in the spreadsheet API for music generation.

I have not been able to discuss the details of this with Advait.

Once playback is facilitated, the next stage of the project would be to build a compiler from the formal musical notation, e.g. MusicXML, to the Calculation View Language. As edits in Calculation View are propagated to the standard grid, understanding the structure of Excel files wouldn't be required.

Once within Calculation View, the desired musical notation can then be represented within the spreadsheet from which playback will occur.

As an extension I could explore reducing the size of the representation within the spreadsheet. For example, a repeated chord sequence could only be shown once in the spreadsheet whilst keeping an understandable representation. Whilst this is not conventional compression, similar lossless or lossy algorithms for eliminating statistical redundancy can be employed.

The project has the following main sections:

1. Facilitating Musical playback through a TypeScript version of Excel run from the browser.
2. Execution and playback of musical definition code in the grid cells
3. Converting a formal music notation to Calculation View to then populate a spreadsheet for music generation.
4. Evaluation and the preparation of examples to demonstrate the success of the implementation.
5. User testing to evaluate the success of extension tasks.
6. Writing the dissertation.

Evaluation and Success Criteria

The spreadsheet API for music generation could be evaluated using a list of requirements that the implementation covers. For example, a user is able to play loops, compound operations exist for chords, notes can be played above a minimum playback frequency, etc.

Usability testing can be carried out for evaluating the interface and the language provided for generating music.

For evaluating the conversion to Calculation View, unit tests can be run on a series of inputs where outputs are verifiable. For example, we can test based on the calls to browsers sound API. If compression is used, user testing can be used again to establish if the compression version sounds sufficiently similar. Round trip verification could be used for lossless compression.

Success criteria

For the project to be deemed a success the following must be completed:

- Implementation of an API for music playback within the Microsoft Excel.
- Conversion of a formal music notation to Calculation View to then populate a spreadsheet for music generation.
- (Extension) Successful conversion from a musical notation to Calculation View. This can be evaluated by round-trip verification.
- (Extension) Lossy compression can be evaluated by user testing.
- (Extension) Usability testing can be employed if the ability to use Excel for live music coding is realised.

Plan of work

Below I outline the plan for successful completion of a successful project. I have outline above various extensions, some of which I hope to be able to also implement.

Before Proposal Submission: - 19/10/18

Submit the final project proposal before Friday 19th October, 12:00.

Section 1: 19/10/18 - 9/11/18

3 weeks
Weeks 3-5 of Michaelmas term

Gain familiarity with the system which I will be building on. Work on facilitating basic music playback so that basic notes can be played from cells within the Excel grid.

Milestone: Ability to sound from within Excel grid

Section 2: 9/10/18 - 30/11/18

3 weeks
Weeks 6-8 of Michaelmas term

Begin implementation of spreadsheet API for music generation and implement tempo/tick so that timing can be specified. Implement playing of arbitrary notes at arbitrary times.

Milestone: Ability to play arbitrary notes at arbitrary timings

Section 3: 10/12/18 - 4/1/19

3.5 weeks
Out of Cambridge

Increase API for music performance so that chords, scales and precision can be specified. It may also be useful to provide other functionality that may aid in compression. However I don't want to sacrifice usability for live coding in the process.

Milestone: Completion of spreadsheet API for music generation

Section 4: 4/1/19 - 17/1/19

2 weeks
In Cambridge before term starts, Lent term week 0

Write progress report.

Implement a simple conversion from formal music format to Excel grid via Calculation View.

Milestone: Conversion of formal music format to Excel grid via Calculation View

Section 5: 17/1/19 - 7/2/19

2 weeks
Weeks 1-3 of Lent Term

Overflow period. As this marks the end of key task of the project it is also a time where I may wish to restructure/neaten certain parts of the code to help with future tasks. Time to research and explore techniques for compression of the musical representation in Excel. Familiarity with the music generation API should help inform decision making at this point.

Prepare Presentation

Milestone: Submission of Project Report and Presentation

Progress Report Deadline: Fri 1 Feb 2019 (12 noon)
Progress Report Presentations: Thu 7, Fri 8, Mon 11 or Tue 12 Feb 2019 (2:00 pm)

Section 6: 7/2/19 - 28/2/19

2 weeks
Weeks 4-6 of Lent Term

(Extension) Implement compression of musical representation in Excel grid. This will be done as part of the conversion to Calculation View process.

Evaluation of conversion to Calculation View.

Prepare examples and methods for evaluation. If human evaluation is required, the examples that are played must be prepared, and the interface and tasks to perform planned.

Milestone: Prepare examples and methods for evaluation

Section 7: 28/2/19 - 14/3/19

2 weeks
Weeks 7-8 of Lent term

Run evaluation trials and summarise results of trials. Prepare suitable examples for evaluation and inclusion in final dissertation write-up.

Milestone: Complete coding and evaluation for dissertation write up

Section 8: 14/3/19 - 18/4/19

5 weeks
Easter vacation

Full time Dissertation write up. As marks are awarded on the final dissertation I would like to be able to allocate a lot of dedicated time for writing this up. I would also like to be almost complete by the time I return to university for Easter term as I would like to spend this time onwards mostly on revision. By submitting a draft before the end of the vacation I will be able to go over it with by supervisor when I return to Cambridge and have time to go over any changes.

Milestone: Submit Dissertation First Draft

Section 9: 18/4/19 - 9/5/19

3 weeks
Start of Easter term

I will have returned to Cambridge by this time and hope to be spending the majority of my time revising for my final exam. This, however, allows time between a draft submission and final deadline to make any final changes. This also leaves time if evaluations have run over.

Milestone: Submit Final Dissertation

Dissertation Deadline (electronic): Fri 17 May 2019 (12 noon) Source Code Deadline (electronic copies): Fri 17 May 2019 (5:00 pm)

Resources Required

Development Machine I shall use my personal laptop for most development work for this project. It is an *Apple MacBook Pro* 13" (2015), 2.9 GHz *Intel* i5 CPU with 16GB RAM. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. I can use MCS machines to do any lighter, more portable work. These shall certainly be used if my machine become unavailable.

Software Access to version of *Excel* run in typescript with Calculation View would be required. This can be facilitated by my supervisor (Advait Sarkar, advait@microsoft.com) who works at Microsoft Research. *Git* will be employed for version control of both implementation source code and documentation. The Dissertation shall be written in *LaTeX*.

Backups I shall use *Github* to remotely back up source code and documentation. These can then be pulled to an MCS machine in the case of personal machine failure. I shall periodically pull this repository to the MCS anyway so that a recent snapshot is always stored on the University system.

References

[1] A. Sarkar, A.D. Gordon, S. Peyton Jones and N. Toronto, "Calculation View: multiple-representation editing in spreadsheets" in *Visual Languages and Human-Centric Computing (VL/HCC), 2018 IEEE Symposium on.* IEEE, Oct 2015 pp. 85-94

[2] A. Sarkar, "Towards spreadsheet tools for end-user music programming", Computer Laboratory University or Cambridge