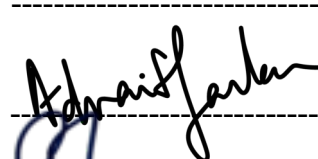# Computer Science Tripos

Part II Project Proposal Coversheet

Please fill in Part 1 of this form and attach it to the front of your Project Proposal.

Part 1

Name: **2393G**

CRSID:

College:

Overseers: (Initials) **AMP & RKM**

Title of Project: **Music Generation with Excel's Calculation View**

Date of submission: **19/10/2018**

Will Human Participants be used? **Yes**

Project Originator: **Alan Blackwell**   Signature: --

Project Supervisor: **Advait Sarkar**   Signature: --

Directors of Studies: **Richard Mortier**   Signature: --

Special Resource Sponsor: **Advait Sarkar**   Signature: --

Special Resource Sponsor: Signature: --

Above signatures to be obtained by the Student

Part 2

Overseer Signature 1: --

Overseer Signature 2: --

Overseers signatures to be obtained by Student Administration.

Overseers Notes:

Part 3

SA Date Received:

SA Signature Approved:

Computer Science Part II Project Proposal

Music Generation in Microsoft Excel

16/10/2018

Introduction

Excel and other spreadsheet tools have become universally popular, both in businesses and individually, for storing, processing and visualising data. However, at present there is not the functionality for the playback of music. Many existing music production packages utilise a grid like format with time passing along the x-axis and parts down the y-axis. Therefore, spreadsheets seem like a promising environment from which to be able to carry out basic music composition in this format and others.

Many people are already familiar with representing concepts in spreadsheet form. This project will explore the use of Excel for musical expression and, as an extension, as a live music coding environment.

Starting Point

No existing work or further knowledge than part Ia and Ib courses. I am a seasoned musician and musical theory enthusiast so possess all the required musical theory knowledge.

I will be building on top of existing spreadsheet service. I would aim to use the Microsoft Office API Office.JS (a public API) and use Add-ins for adding my own functionality, if there is sufficient support for sound. This is publicly available. If not, I would either be able to use a Typescript prototype of Excel from Microsoft or an existing open source JavaScript implementation of a spreadsheet.

Substance and Structure

By using a TypeScript or JavaScript version of Excel run in browser, playback functionality can be built on top of the web audio API. Functionality for note and sequence synthesis functions will be required. A converter from an existing formal music specification to the spreadsheet representation will be implemented. As an extension, live coding can be implemented.

Firstly I will have to establish what notation is used within the cells. Within a given cell, I would like to be able to play single notes and chords. Beyond this defining scales and arpeggios would be useful for reducing the size of grid required to define pieces. This notation must then be interpreted with a resulting call to the browser audio API. It would also be desirable to be able to define sequences of notes (e.g. baselines, repeating melodies) elsewhere in the grid and then be able to call these elsewhere in the playback. This means that users do not have to copy and paste repeating sections and it would also be clearer where sections are repeated.

Next, the representation that is supported between cells must be decided and implemented. The flexibility of spreadsheets allows users to define their own secondary notation in the way that sections within the grid are laid out. As a result, allowing for the relative positions of different sections within a sheet and their orientation to vary would allow familiar Excel users to continue

defining their own layout. The definition and re-use of phrases and parts would allow for fast prototyping of musical ideas. The representation will likely be that of a main playback loop (which can be split into multiple parts), with definitions of sequences outside of this main loop section.

After establishing my notation and supported layouts, the program must compile this representation into audio output for playback. Firstly, defined variables (e.g. Tempo) and regions where melodic lines are defined out of the main playback loop must be detected. Next, the main playback loop region and its orientation must be detected. After this, the information can be processed and converted into calls to the audio API.

As an extension, I can add support for live music coding. To facilitate live music coding, it should be possible to change notes within the grid and recompile whilst playback is occurring. Live music coding encourages a loop based approach to music so run/compiled changes to the grid should become apparent in the playback whilst not requiring a restart of the output. The program would be able to parse the data within the spreadsheet and identify different regions and declarations. From this it would convert the main playback loop with the output being calls to the growers audio API. This would include integrating sequences that are defined out of the main loop but called with in.

Once the representation of musical structure has been decided and the playback of this representation implemented, I will implement a conversion from some form of formal music notation (e.g. MusicXML or MIDI) to the spreadsheet representation. Existing pieces can then be immediately transformed into the spreadsheet layout and played back using the spreadsheet music API.

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 audio playback from a spreadsheet, run from the browser.
2. Execution and playback of musical definition code in the grid cells.
3. Playback of multiple cells where time is represented in an axis or the code within cells.
4. Implementation of a converter from a formal music notation to the spreadsheet representation.
5. Evaluation and the preparation of examples to demonstrate the success of the implementation.
6. User testing.
7. Writing the dissertation.

Evaluation and Success Criteria

A successful implementation should allow a user to carry out the following:

- Play individual notes and chords and define their durations.
- Defining multiple parts.
- Play loops.
- Define sequences of notes and chords and be able to call these for playback.
- Define the tempo of playback.

Qualitatively, use of the music playback API can be analysed using a friction analysis approach as in [3] and a cognitive dimensions profile strategy.

With some basic explanation, users can be measured carrying out simple tasks or free composition. From this we can measure Time To Hello World (TTHW) (e.g. playing a note). Friction diagrams generated based on observation of a user working with the program in a usability study can be used to evaluate the productivity of users of the tool.

We define the following desirable features in a cognitive dimensions profile. This defines the desirable structural usability properties of the API and interaction UI.

- Reasonable **closeness of mapping** (use of the grid structure should allow for much higher closeness of mapping than e.g. Sonic Pi where there is only one file of code).
- High **consistency** for the definition of notes and chords within phrases.
- Layout within the grid should allow for high **secondary notation**
- Low **viscosity**
- High **visibility**

We shall then use a Cognitive Dimensions questionnaire to empirically categorise users' response to it. Evaluation can be carried out by comparing that response to this desired profile.

Quantitatively, the expressiveness of the API can be verified by a translation of a musical corpus from the formal notation to the spreadsheet representation.

The compression rates achieved in the compression of the representation can also be measured and compared to a benchmark of a naive conversion.

Success criteria

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

- Implementation of an API for music playback within a spreadsheet using the above implementation features.
- Implementation of a converter from formal music notation to the spreadsheet representation.
- Usability testing for music generation implementation.

Plan of work

Below I outline the plan for successful completion of a successful project. I have outlined above various extensions, some of which I hope to be able to also implement. I am aiming to finish coding in good time to allow for user testing, evaluation and the dissertation writeup to be completed in time for me to carry out ample revision before my finals.

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.

This time can also be used to consider the layouts of musical representation that will be supported by the API.

Milestone: Ability to create sound from within Excel grid

Section 2: 9/11/18 - 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. At this point sequences can be defined and played back.

Milestone: Ability to play through arbitrary notes at arbitrary timings

Section 3: 10/12/18 - 24/1/19

2 weeks
Out of Cambridge

Make it possible to define note/chord sequences outside of the main playback loop and have this integrated into playback. The sections where these are defined must be found within the spreadsheet and their definitions matches to names in the main playback section.

Increase API for music performance so that chords, scales and precision can be specified.

Milestone: Completion of spreadsheet API for music generation (not live coding)

Section 4: 24/12/18 - 4/1/19

1.5 weeks
Out of Cambridge

History would suggest that the presence of Christmas and the end of the year will require a reasonable amount of my attention. My family will most likely appreciate this period being a little less demanding.

This period can be used to neaten the existing codebase. It may be useful to reimplement certain functions to help with the following stages for implementing live coding and conversion. This time can also be used to research and consider the method for implementing the conversion and live coding. At this point I should be familiar with the audio API and have more sensible ideas for doing this.

This would also be a good time to write a first draft of the introduction section to ensure adequate time can be given to the implementation and evaluation sections at the end of the project.

Milestone: Introduction section draft

Section 5: 4/1/19 - 17/1/19

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

Build converter from formal music format to the spreadsheet representation. Demonstrate success with the conversion of a corpus to the spreadsheet format.

Milestone: Conversion of formal music format to spreadsheet representation

Section 6: 17/1/19 - 7/2/19

2 weeks
Weeks 1-3 of Lent Term

Prepare Presentation

This time can be used to catch up if any of the previous milestones have not been adequately reached. Then this time can be used to work on extension tasks.

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 7: 7/2/19 - 28/2/19

2 weeks
Weeks 4-5 of Lent Term

Prepare examples and methods for evaluation. For human evaluation, the interface and tasks to perform must be planned and prepared.

Milestone: Prepare examples and methods for evaluation

Section 8: 28/2/19 - 14/3/19

3 weeks
Weeks 6-8 of Lent term

Perform write up of results of user testing and analysis. This time can be used to perform small changes for potential improvements that may arise during testing and evaluation.

Milestone: Complete coding and evaluation for dissertation write up

Section 9: 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 10: 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 exams. This, however, allows time between a draft submission and final deadline to make any final changes.

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 a suitable spreadsheet tool will be required. This will depend on the audio capabilities of the implementations outlined above. If OfficeJS if unsuccessful, this will 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

[3] Macvean. A, Church. L, Daughtry. J, Citro. C, "API Usability at Scale" in *Psychology of Programming Interest Group (PPIG), 2016 - 27th Annual Conference.*
Accessed (15/10/2018): http://www.ppig.org/sites/default/files/2016-PPIG-27th-Macvean.pdf