Musical Concurrent Programming With Sonic Pi

Interim Report

Eleanor Vincent

January 25, 2015

1 Introduction

1.1 Motivation

For many years it has not been a requirement that children should learn much about the vast field of computing during their formative years in UK Education. Some of the earliest significant pieces of work towards the education of computing started with the invention of Logo, an adaptation of the LISP language, most remembered for its use of "turle graphics". Some time after this came the Computers in the Cirriculum Project, funded from 1972 to 1991 by the Schools Council and subsequently by the Microelectronics Education Programme in 1981. The first microcomputers appeared in both UK Primary and Secondary Schools in 1979. The Commodore Pets aided both spelling and arthimetic practice as well as the ability to teach either BASIC or Logo. The 1980's saw a huge amount of legislative reform and technical efforts in the vein of giving young people the ability to work with computers during their formative years but over the course of the late 80's and early 90's this focus on programming ability gives way to simply the education of practical use of existing computer software instead of a focus on the fundamentals behind these applications [2].

In 2013, Ofsted published a report documenting their findings relating to ICT in UK schools from 2008 to 2011 and found that in half of all secondary schools, school leavers had not been given adequete education to move into a technical career in their future. In 2007, 81,100 pupils were enrolled in the ICT GCSE but this had fallen to 31,800 pupils by 2011 [10] with a notable lack in the education of key skills such as computer programming itself. This lack was found to be as much a lack in knowledge from the teacher's as much as the cirriculum's failure to address the issues.

In 2012, the Government began to recog the signifiance of Computer Science and has replaced the National ICT curriculum with a revised Computing cirruculum. The new Computing Program of Study [11] aims to enable pupils to understand the world of computing, giving them the ability to think logically and apply the fundamental principles of the discpline to their real-world environments.

Along a similar vein there is a general struggle within the humanties courses available in schools

to remain relevant in light of a quickly developing technical world. Music schemes within the UK frequently report to have suffered funding cuts and education is focused on learning a variety of specific instruments, with little focus on musical technology until the later years of education.

As well as the necessity of Computer Science and Music within schools, there is an increasing recognition of the power of programming amongst the general populace. A growing hacker and maker movement has been making programming a much more accessible skill [7]; it presents itself as a viable and useful hobby amongst a vast range of ages and professions and this is as much because of the availability of useful resources that would be frequently used in such situations as a school classroom. There is existing research that has gone into the viability of programming tools for professional artists, as reported at PPIG [9, 8], and investigations into the craft practices of existing professional software developers who work in professional art contexts [12].

The movement is not purely restriced to the UK. In the US there is a similarly led campaign calling to recognise the topic as relevant to all contempary sciences. It calls "Computational Thinking" a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use [13].

Sonic Pi is one of many projects designed to support both computing and music lessons within schools. Sonic Pi is an environment for creating live-coded music at a level of complexity which is well suited to a first-programming language [3]. There are many languages in existence which are simply enough to also constitute as a good first-programming language, but many do not attempt to make themselves an inviting gateway into the realm of technical programming. Sonic Pi seeks to provide an exploratory and invigorating introduction into programming whilst being complex enough to lead a user through into much more complicated programming ideas and use cases with a managable learning curve. By presenting a musical system it seeks to break down the barrier between the technical elite and the hobbyist.

1.2 Objectives

This project's aim is to formalise the concurrency and timing aspects of the language and develop a program analysis tool that can be used to identify program bugs such as deadlock and thrashing behaviour. The formalism and analysis will build on initial work describing timing and concurrent interaction via effect systems and session types. The project combines theoretical aspects of type system and analysis design, as well as practical work developing a responsive program analysis engine.

1.3 Report Structure

The remainder of this report is broken down as follows:

-Background: We detail the histories of Sonic Pi, Live Programming and Session Types as fields of research and conclude with work relating to these subjects.

-Going Forward: The bulk of this section details the expected timeline of the project and ideas for how to evaluate progress during and at the conclusion of the project.

2 Background

In this section we begin by explaining the ideas and features of Sonic Pi, the living coding program that forms the basis of this project. We then move on to explain the subject of both Living Programming and Session Types in further detail and seek to relate them back to the current aims of the project. We conclude with details on the related work in these areas of research.

2.1 Sonic Pi

Sonic Pi is an imperative live programming language designed as an educational first language. It is a ruby-based domain-specific language designed for manipulation of synthesisers through time [5]. It is currently in its second iteration, with the main extension between the two languages being the work done to improve the timing system of the project, which is discussed in more detail below. Some of the concepts that Sonic Pi is well suited to teach, in direct relevance to the current UK Computing in Schools Cirruculum, are conditionals, iteration, variables, functions, algorithms and data structures. Sonic Pi also extends beyond these concepts to include such things as multi-threading and hot-swapping of code as these are likely to be of crucial importance in the future of programming contexts [6].

This section first gives a breif description of the Raspberry Pi then explores the implementation of Sonic Pi V1, mainly to give appropriate context to the timing effects system that Sonic Pi V2 currently imeplments. There is then be some short discussion on the particular features of Sonic Pi V2.

2.1.1 Raspberry Pi

The Raspberry Pi was developed as a very low cost computer system to enable technical experimentation amongst young people who had little other contact with computer systems. The idea came about in 2006 as an answer to the steadily decreasing levels of pupils applying to take up Computer Science after their A-Levels. The reasons for this were attributed to many contributing events such as the end of the dot come boom, the focus of IT lessons on Microsoft software and building very basic HTML websites and the increased availability of out-of-the-box games consoles over the Amigos, BBC Micro, Spectrum ZX and Commodore 64 machines that promoted individual experimentation so freely in the past [4].

The Pi is provided as a bar circuit board costing roughly $25, able to boot into a Linux environment with very little other commerical equipment required. The Raspberry Pi Foundation is a non-profit organisation and has sold over a million products since 2012. The main objective is to develop genuine technical competency by allowing the freedom to experiment with the whole system rather than taking the locked box approach of other systems. Learning becomes self directed for pleasure rather than at the behest of a mark molded system. It is this style of engagement that brought the Raspberry Pi to the attention of educational campaigners and has since enabled it to be used so successfully within the new movement towards better Computing education within Schools.

2.1.2 Sonic Pi V1.0

The initial development time given to Sonic Pi v1.0 was roughly three weeks.

In this report Sonic Pi will be presented in terms of the formalisation of its timing effects and the existing concurrency primitives of the language.

2.2 Live Programming

For much of the prevailing history of programming there has been an idea that the programmer is inherently separated from the system that they are producing. The task of the programmer is to create a system based on some formal specification that will take effect at some unknown point in the future, and the time between implementation and action has no effect on the results that the system will produce. In this way, there is a strong sense of separation between the program, process and task domains where the progam is the code implementation and specifications, the process is the running of the code on a specific machine and the task is the visible real world results****. This is a viewpoint that many would not think to challenge as it is natural to assume that the methodology of a computer programmer would naturally lend itself to implementation of actions that were set for execution in the future and, in general, would process a deterministic set of results that can be repeatedly used as the users required.

Live programming (also referred to as With Time Programming or Just In Time Programming) seeks to apply the improvisional nature of time to the existing methodology of the programmer. With this idea it becomes possible to define a tigher system of feedback between the program and task domains through means of whatever process domain is most suitable. The improvisional nature of the acitity also removes the inherent requirement of a specific program specification and allows for new level of freedom and creativity in the programs being created.

Given the nature of Live Programming the languages that invoke it are often dynamic language which allow for the flexibility, conciseness and ease of development*(5)* to enable the act of live programming to feel as natural as standard programming practice.

Live Programming lends itself also to acts of performance, where Live Coders will often perform to live audiences, often producing such things as live improvisational music or artwork, whilst having some means by which the audience will also see the code written at the same time. From a social and cultural viewpoint, Live Programming lends itself to breaking down the barriers that have been built up between software technology and the creative users*(6)*.

****Programming with Time Improcess *(5)*Living it up with a Live Programming Language *(6)*The Textual X

2.3 Session Types

2.4 Related Work

3 Going Forward

This section illustrates the plan of action laid out for the rest of the project over the next few months. The first section illustrates the rough idea of how long each implementation of the program features will take, with documentation and evaluation time factored in. After that is the outline of how the project will be evaluated.

3.1 Project Outline

3.2 Evaluation Outline

4 Final Thoughts

References

[1] Auther, *LaTeX: Title*, Publisher, Place, Edition, Year

[2] *http://www.naec.org.uk/events*

[3] *http://Sonic Pi.net/*

[4] *http://www.raspberrypi.org/about/*

[5] Aaron, S., Blackwell, A.F., *From Sonic Pi to Overtone: Creative Musical Experiences with Domain-Specific and Functional Languages*, The First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design, Boston, Massachusetts, USA, ACM, pp. 35-46, 2013

[6] Aaron, S., Orchard, D., Blackwell, A.F., *Temporal Semanics for a Living Coding Language*, Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling & Design, Boston, Massachusetts, USA, ACM, pp. 37-47, 2014

[7] Blackwell, A.F., Aaron, S. and Drury, R., *Exploring Creative Learning for the Internet of Things Era*, In B. du Boulay and J. Good (Eds) Proceedings of the Psychology of Programming Interest Group Annual Conference, pp. 147-158, (PPIG 2014)

[8] Blackwell, A.F. and Collins, N., *The Programming Language as a Musical Instrument*, In Proceedings of the Psychology of Programming Interest Group Annual Conference, pp. 120-130, (PPIG 2005)

[9] Church, L., Rothwell, N., Downie, M., deLahunta, S. and Blackwell, A.F., *Sketching by Programming in the Choreohraphic Language Agent*, In Proceedings of the Psychology of Programming Interest Group Annual Conference, pp. 163-174, (PPIG 2012)

[10] Department for Education and Ofsted, *ICT in schools: 2008 to 2011*, Piccadilly Gate, Manchester, 110134, 2013

[11] Department for Education, *National curriculum in England: computing programmes of study (key stages 1 - 4)*, 2013

[12] Woolford, K., Blackwell, A.F., Norman, S.J. & Chevalier, C., *Crafting a Critical Technical Practice*, Leonardo 43(2), 202-203, 2010

[13] Wing, J.M., *Computational Thinking*, Communication of the ACM, Vol. 49, pp. 33-35, 2006