Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music Composition with Sonic Pi

by

Christopher Grant Petrie

A Dissertation Presented to the
College of Education, Health & Human Development of University of Canterbury
in Partial Fulfilment of the Requirements for the Degree of
Master in Education with e-Learning and Digital Technologies endorsement

University of Canterbury
2019

Acknowledgements

I wish to express my heart felt gratitude to my main supervisor Dr. Valerie Sotardi. Her patience and guidance during this whole process has contributed a great deal to advancing my thinking on academic research. In addition, I would like to thank Dr. Cheryl Brown and Niki Davis for their additional guidance and expertise to help shape this work. Finally, a big thank you to my friends and family for their support and love.

Table of Contents

List of Tables

List of Figures

Abbreviations

CI = Communicating and Interpreting (Music strand)

CT = Computational thinking

DI = Developing Ideas (Music strand)

DT = Digital Technologies

DTC = New and revised Digital Technologies content in the Technology learning area of the New Zealand Curriculum

PK = Practical Knowledge (Music strand)

MC = Music Curriculum

MoE = Ministry of Education

NZ = New Zealand

NZC = New Zealand Curriculum

SOLO = Structure of Observed Learning Outcomes

UC = Understanding Context (Music strand)

Data reference abbreviations

CP = Chris Petrie (interviewer)

rr = Researcher reflection

si = Student Interview (pre/post)

sr = Student reflection

T1 = Pre-trial questionnaire

T2 = Post-trial questionnaire

ti = Teacher Interview (pre/post)

tq = teacher questionnaire (pre/post)

Zach (pseudonym) = Participant music teacher

*All data references will have dates and pseudonyms of participants as appropriate; for example (Ben, si, 11/5/18).

Abstract

Digital technologies are changing the skills young people need in the future. To prepare school students for these anticipated changes, a metacognitive skill called Computational Thinking (CT) has been integrated into new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC). Interdisciplinary approaches to develop CT may be a partial solution to growing concerns about teacher capability on delivering the DTC in an already overcrowded school timetable. For example, Sonic Pi is a relatively new programming platform (based on Ruby) that is designed to help beginners at a school level, which can combine both music composition and programming into one activity. Sonic Pi may promote more positive attitudes towards programming because it enables a creative introduction to this skill with music making. However, there is a lack of research on how to effectively teach and assess both CT and Sonic Pi due to a paucity of case study research at a school level.

This research examined how interdisciplinary CT supports learning outcomes in music and programming with the Sonic Pi platform. Additionally, it investigated the extent to which the creative activity of music composition with the Sonic Pi platform could potentially promote more positive attitudes towards programming. A mixed-method case study with a designed unit of work by the researcher was conducted. Action research framed the methodology of the unit of work, which was trailed in a music classroom and led by the researcher with 22 Year 8 participants and their regular classroom music teacher (taking on a support teacher role).

Overall, the findings illuminated many successes and challenges not reported in the literature on how CT can support learning outcomes in music and programming. It is recommended educators seriously consider the identified challenges on integrating the CT concepts of conditions and operators as well as the CT practice of abstracting and modularising. However, the findings also highlighted pedagogical benefits unique to programming in Sonic Pi, which supported overlapping learning outcomes in music and

programming. For example, the CT programming concepts of parallelism and sequences in Sonic Pi overlapped in a way that was beneficial to emphasise the importance of silence (or time) between sounds in music. On measuring pre and post attitudes, the findings were especially encouraging because they suggested the unit of work promoted an increased sense of programming self-confidence and that programming is an important skill to learn for both the student participants and the participant music teacher. The major implication of these findings suggest educators can use Sonic Pi and the designed unit of work to: (a) teach interdisciplinary CT to meet many learning outcomes in music and programming; and (b) to potentially promote more positive attitudes towards programming for those students interested in music composition.

Thus, Sonic Pi is recommended as a unique and creative way to introduce both programming and music composition to school students in Year 8. The overall contributions for both research questions indicate the designed unit of work and the Sonic Pi platform may aid the successful integration of the DTC in NZ. Moreover, this case study expands the literature on both interdisciplinary CT and the Sonic Pi platform.

Chapter 1: Introduction

Rapid innovations in digital technologies are changing the skills young people need in the future. Computation in the digital world is a key driver of this change, which is the process of complex calculation when computers execute algorithms (Shute, Sun & Asbell-Clarke, 2017). As a result, many predict that those who possess the ability to use computation effectively will have a competitive edge in the modern workforce (Barr & Stephenson, 2011; Wing, 2011; Mohaghegh & McCauley, 2016).

A metacognitive skill called Computational Thinking (CT) is being promoted to develop this ability to use computation effectively in schools (Wing, 2011). Metacognitive skills involve 'thinking about thinking', as higher-order cognition, and play an important role in the development of a broad range of core competencies, like problem solving (Flavell, 1979). CT involves the various thinking habits of solving problems for computers (Grover & Pea, 2013; Kalelioğlu, Gülbahar, & Kukul, 2016; Shute et al., 2017; Wing, 2011) and is being recommended to assist with students' preparation for a modern workforce in New Zealand (Hipkins, 2017; Mohaghegh & McCauley, 2016).

Wing (2006) is credited for sparking the support needed to integrate CT into schools, and has since boldly declared it as the "new literacy of the 21st Century" (Wing, 2011, p. 4). Wing's idea resulted in enough backing from governments, schools, and industries to integrate CT into many curricula internationally (Bell, 2015; Grover & Pea, 2013; Kalelioğlu et al., 2016; Webb et al., 2017). For example, in 2018, the New Zealand Government formally introduced CT as one of two strands as new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC), which will be mandatory for all school students in 2020 (Ministry of Education [MoE], n.d.c). This change to the DTC in New Zealand (NZ) strengthens Digital Technologies importance in the Technology learning area, which can include other subjects in schools, like Food Technology (MoE, n.d.a).

The NZ Minister of Education in New Zealand, Chris Hipkins, stated in 2018 that the DTC is intended to teach “children how to design their own digital solutions and become creators of, not just users of, digital technologies, to prepare them for the modern workforce” (MoE, n.d.c, para. 3). The new CT strand in the DTC is a key element to enable this intention. Brown, Czerniewicz, and Noakes (2016) also reinforce this overarching notion that students should learn to be creators of digital technologies (rather than passive users) to gain a competitive edge in the future job market.

A major challenge concerning the DTC’s integration of CT is the ability to fit it into the school timetable (Mohaghegh & McCauley, 2016; MoE, 2017). One potential solution could be to take an interdisciplinary approach to CT, with learning outcomes in two (or more) subjects as one unit of work (Shute et al., 2017; Weintrop et al., 2016). In addition to saving time, this interdisciplinary approach could also offer opportunities to introduce the subject of Digital Technologies (DT) in new creative ways with subjects like music (Bell & Bell, 2018; Burnard, Lavicza, & Philbin, 2016). An emphasis on creativity through combining CT with music may shift the extrinsic focus away from only preparing young people for the modern workforce, and more towards intrinsic motivations. For example, interdisciplinary creative activities with music composition and programming may be a powerful way for students to develop an expressive voice with digital devices, which could also promote more positive attitudes towards programming (Burnard et al., 2014; Davies et al., 2013; Engelman et al., 2017).

Programming is often viewed as a core part of CT because it enables humans to easily test new designs of computational agents (Shute et al., 2017; Tedre & Denning, 2016). Computational agents could include mechanical machines (or even humans), however, today most people use digital devices and make software through a programming language (Wing, 2008). A programming language and platform called Sonic Pi (based on the Ruby programming language) provides a unique opportunity for beginners to learn programming through the creative activity of making music with code (Aaron, 2016). Sonic Pi was built in collaboration with teachers and many hours of observation of students to ensure it is pedagogically sound (Blackwell & Aaron, 2015; Burnard et al., 2014).

A difference between Sonic Pi and many other learn to code platforms aimed at school students is that it is both text-based (forcing students to type instructions – rather than dragging and dropping commands with the mouse), and it primarily deals with audio output (typical programming languages have capabilities for a range of outputs) (Aaron, 2016). For example, to test output similar to a ‘Hello, World!’ statement, you can simply type “play 60”, where the number corresponds to midi notes on a keyboard (60 plays C in the 4th octave). Parallel voices (e.g. chords) can be simply be programmed by typing these play commands on adjacent lines. However, in order to compose melodies, a “sleep” command is needed with the number of desired seconds following each note so that the computer knows *when* to execute them. In this way, the general nature of coding in Sonic Pi is initially counterintuitive because it is designed for music making.

A study on Sonic Pi by Burnard et al. (2016) observed learning overlaps between music and programming that also positively contributed to students’ confidence in both subjects. However, while learning was claimed to have taken place in both music and programming, useful details for teachers on specifically *what* was learnt and *how* students were learning are not reported.

Thus, this thesis conducted a mixed-methods case study, with a designed unit of work, which aimed to understand how CT can support learning outcomes in both music and programming. The main purpose was to expand the research on interdisciplinary CT and Sonic Pi through designing learning outcomes in music and programming as one unit of work. Moreover, a sub-purpose of this research was to investigate the extent to which the creative activity of making music with Sonic Pi may help to promote more positive attitudes towards programming.

1.1 Rationale and Research Questions

A growing body of literature is calling for more case studies on CT to be conducted in regular school classroom settings to better understand how teachers can teach and assess this metacognitive skill (Denning, 2017; Grover & Pea, 2013; Lye & Koh, 2014; Mannila et al., 2014). This thesis helps to addresses this need with a mixed-methods case study, which was conducted in a school classroom context. It was also conducted at a time when CT was first being introduced to all students in New Zealand and about to be made mandatory in the year 2020.

Previous studies on metacognition have suggested young people only start to develop metacognitive skills when they reach eight years of age, with most development starting in early adolescence (Flavell, 1979; Mannila et al., 2014). CT requires many metacognitive abilities, like sequencing instructions, problem solving, and self-reflection (Brennan & Resnick, 2012; Kalelioğlu et al., 2016; Shute et al., 2017). Thus, the ideal time to start developing CT is in middle school. For this reason, Year 8 has been chosen as the ideal school level to conduct this research.

Many students view programming as difficult and complicated (Grover, Pea, & Cooper, 2016; Margolis & Goode, 2016; Anderson, Lankshear, Timms, & Courtney, 2008). However, linking programming with the creative activity of music composition has been shown to potentially promote more positive attitudes towards programming (Burnard et al., 2014; Cheng, 2018). In turn, an interdisciplinary approach could increase both music students' interest in programming and programming students' interest in music composition. This approach may also provide insights into ways music teachers could participate in students' development of CT skills (Aaron, Blackwell, & Burnard, 2016), which has the potential to increase teacher capability to aid the successful integration of the DTC.

From a music perspective, CT may highlight the development of skills that may be useful for music composition, for example: pattern recognition, abstraction, algorithmic thinking, and decomposition (Bell & Bell, 2018; Edwards, 2011). With Sonic Pi, there may be transferable and overlapping concepts to strengthen learning outcomes in both music and programming (Aaron & Blackwell, 2013; Burnard et al., 2016).

The aim of this mixed-methods study was to connect music and programming through the Sonic Pi platform with support from CT for learning outcomes in both subjects. A designed unit of work was trailed in a music classroom (teaching was led by the researcher) with 22 Year 8 student participants their regular classroom music teacher (who took on a support teacher role). The project sought to: (1) see how CT can support learning outcomes in music and programming using the Sonic Pi platform; and (2) to investigate the extent to which the creative activity of making music with Sonic Pi helps to promote more positive attitudes towards programming. Thus, the research questions for this study are:

Main research question: How can computational thinking (CT) support learning outcomes in Programming and Music with the Sonic Pi platform?

Sub-research question: To what extent can the creative activity of composing music with the Sonic Pi platform help to promote more positive attitudes towards programming?

These questions were answered through conducting a mixed-method case study with a designed unit of work by myself. Action research framed the methodology of the designed unit of work. Theoretical frameworks were also employed for both research questions to help design instruments and code the data gathered as part of a directed content analysis approach recommended by Hsieh and Shannon (2005). The main research question employed the CT assessment framework by Brennan and Resnick (2012) with the three CT dimensions of *concepts, practices* and *perspectives*. The sub-research question used a modified framework used by (Teo, 2007) for measuring participant attitudes, which had the elements of *enjoyment*, *importance*, and *self-confidence*. In addition, research and policy recommendations for both questions will be made in Chapter 5 as a result of reflecting on this study.

I led the teaching of the unit of work and the participant teacher's role was to help students where he could. So that my perspective and possible influence is clearly established, I will briefly introduce myself as recommended by Herr and Anderson (2015):

The lens I am considering the results of this research is from having over ten years' experience teaching music and computer science at a high school level in Wellington and Auckland (New Zealand). One of the unique aspects about my professional life in education is the unusual breadth of experience I have had with a wide range of learners from different backgrounds, cultures, and ages. For example, I have taught students in New Zealand (state, kura kaupapa, private and Montessori), Germany (international) and Nepal as a volunteer (Montessori). In music, my training involved learning jazz saxophone, classical piano, music composition and production. In programming, my training largely focused on web development with JavaScript, however, I have also taught Python and Scratch. Moreover, I have completed two years of Masters level coursework on teaching computer programming (2016) and computer science (2017) at the University of Canterbury.

The following chapters present a literature review (Chapter 2), the methodology used for this study (Chapter 3), findings from the data gathered (Chapter 4), and a discussion of the findings (Chapter 5).

Chapter 2: Literature Review

To position this study within existing research, this Chapter aims to present a review of literature relating to the research questions previously stated in the introduction (see Chapter 1). It is structured in two parts as follows: first, literature on Computational Thinking (CT) and how interdisciplinary links between music and programming can support learning outcomes is reviewed for the main research question; second, literature relating to creative activities promoting more positive attitudes towards programming is reviewed for the sub-research question.

2.1 Literature Review of the Main Research Question

The ideas around CT have been in development and discussion since the 1950s (Tedre & Denning, 2016), however, Seymour Papert is often credited as the first to describe the idea of computational thinking at a school level (Papert, 1980). Papert initiated the first major research group to understand how best integrate CT for school students with a programming language called Logo (Lye & Koh, 2014). In his book, *Mindstorms*, Papert (1980) presents his ideas on CT as a mental skill children develop so they can use computers as instruments for powerful learning. This effort to make CT widespread from Papert and others during the 1980s and 1990s did not gain widespread adoption at the time, largely because digital devices were considered too costly for the majority of schools to purchase (Lye & Koh, 2014).

Since then, efforts to promote the benefits of CT at a school level have seen a resurgence, both because first world countries today can afford powerful digital devices and from an urgent push to prepare children for the modern workforce (Barr & Stephenson, 2011; Mohaghegh & McCauley, 2016). Wing (2006) is credited with sparking this resurgence; she argued CT is now a “fundamental skill for everyone, not just computer scientists” (p. 33), and later declared CT as a “new literacy of the 21st Century” (Wing, 2011, p. 3). These articles by Wing sparked a groundswell of support to integrate CT into schools across the world (Tedre & Denning, 2016), which influenced the New Zealand Government to add CT into their curriculum (Ministry of Education [MoE], n.d.a).

The New Zealand Government started implementation of a new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC) in 2018 that includes CT as one of two strands, which is to be made mandatory for all schools by 2020 (MoE, n.d.c). CT was not explicitly included in the original DTC prior to this change. Despite the pedagogical benefits CT has been promoted to offer, major concerns are growing in New Zealand (NZ) around insufficient teacher capability and a lack of time for current teachers to upskill (Martin Jenkins, 2017; MoE, 2017). Similar issues have been voiced in other countries on the integration of CT into their curricula (Denning, 2017; Mannila et al., 2014; The Royal Society, 2017). As a result, the DTC is likely to be implemented at a low standard from teachers who struggle to understand CT, which may promote more negative attitudes towards Digital Technologies (DT) as a subject in both teachers and students.

2.1.1 Definition of computational thinking. CT is a metacognitive skill being promoted as essential to students of all years in their preparation for a less predictable future (Wing, 2006; 2011). Flavell (1979) defines metacognitive skills to involve 'thinking about thinking' – as a higher order of cognition – that play an important role in students' development for a broad range of competencies, like problem solving. Metacognitive skills are said to require both self-awareness and regulation (Cox, 2005). In order for students to exercise regulation, they need to plan, monitor, and evaluate their thinking (Flavell, 1979).

Denning (2017) states that a high proportion of educational research focuses only on students acquiring knowledge, and a metacognition like CT is a paradigm shift in the kinds of skills teachers are familiar with delivering. The introduction of metacognitive skills in schools (like CT) present many significant challenges for implementation because there is a lack of understanding in the best ways to teach and assess them (Lye & Koh, 2014; Denning, 2017). Moreover, there are less obvious limitations for the way metacognition (planning, monitoring and evaluating) can generally happen in a developing child's brain (Flavell, 1979; Mannila, et al., 2016), which will be explored later in this Chapter.

Definitions of CT vary in their balance between knowledge and skill acquisition (Curzon, Bell, Waite, & Dorling, 2019; Denning & Tedre, 2019). Knowledge can be described as the memorisation and recalling of ideas (Flavell, 1979), whereas a skill is a core ability practised and improved over time (Denning, 2017; Shute et al., 2017). For example, in a programming context, a student may simply understand the knowledge of conditional logic, but not yet have the cognitive skill to recognise when it is essential to solve a given problem (Brennan & Resnick, 2012). In a music context, developing skills can be in the form of purposeful practise of the piano over time that can improve a player's scope for a wider range of musical expression – as opposed to the memorisation of music theory concepts like key signatures (Aaron, 2016; Burnard et al., 2014). A debate in the literature exists around the right balance of the knowledge and skills required in CT, consequently, a widely agreed upon CT definition has yet to be established (Curzon et al., 2019; Denning & Tedre, 2019; Grover, 2018; Denning, 2017; Kaelioglu et al., 2016; Grover & Pea, 2013).

Current definitional debates also generally divide around whether CT is a transferable skill that can be used in other domains for interdisciplinary learning (Denning, 2017; Denning & Tedre, 2019). Tedre and Denning (2016) claim that "everyone, not just those who major in computer science, can benefit from thinking like a computer scientist" (p. 120). However, Doleck, Bazelais, Lemay, Saxena, and Basnet (2017) and Denning (2017) found CT skill transfer to other disciplines has yet to be been substantiated in case studies, which makes Wing's (2011) claim problematic for the widespread adoption of CT in non-computing subjects.

The New Zealand e-Learning website *Te Kete Ipurangi* at the time of this writing provide a range of resources to explain CT from organisations such as: CAS Barefoot, Google, International Society for Technology in Education, and New Zealand Curriculum Online (MoE, n.d.c). The high number of links to outside sources suggest educators can adopt any one of the definitions of CT in these to suit their purpose. However, the range of information provided is likely to cause confusion and mixed perspectives amongst educators. For example, many are not clear on whether programming is essential to develop CT, which Denning (2017) argues it is. If programming is essential, then school teachers without programming knowledge and skills will need to learn and apply it to their discipline if they are to develop CT skills in their students.

Despite this confusion, a pattern is emerging where the literature finds common ground on how CT can aid general problem solving skills without programming. For example, an empirical review from 125 papers on the topic of CT by Kalelioğlu et al. (2016) found that the top five most commonly used words were: abstraction, problem solving, algorithmic, and thinking. Developing CT skills based on these terms can give a relatively robust degree of assurance to educators on what skills could be common across domains for interdisciplinary overlaps. Nevertheless, most definitional debates on CT still divide into two opposing camps: CT is only for programming and CT is for all subjects.

2.1.2 CT is only for programming. CT may be easily confused with many other terms associated with computing subjects for non-specialists. For example, ICT, IT, Digital Technologies, Informatics, Coding, Programming, and Computer Science all have different meanings and historical associations (Abbiss, 2011; Bell, Andreae, & Robins, 2014; The Royal Society, 2012). Consequently, these subject terms also all have different emphases, scopes, and depths in the extent CT can be developed within them (Mannila et al., 2014). For example, Abbiss (2009) states ICT or Text and Information Management were historically classes that mainly focused on office work tasks, which provided little or no opportunity to develop CT.

For subjects involving programming, CT helps to dismantle a common misconception that it only involves ‘coding’ (Bell, 2015; Brennan & Resnick, 2012). The activity of coding is predicted as taking only a relatively small fraction of the total time it takes to create software in a professional setting (Fronza, Ioini, & Corral, 2017). Lye and Koh (2014) found when students only learn about programming language syntax without CT, they often create code in an impulsive manner, which lacks structure and focus. For example, in the popular introductory programming language Scratch (scratch.mit.edu), a study by Brennan and Resnick (2012) found students can make projects that have many sprites (characters in an interactive game or story) that serve no dynamic function for the overall purpose of the project. To avoid this issue, CT may help to focus the conception, refinement, optimisation and purpose of using computation to solve problems (Wing, 2008). This indicates CT can promote a more holistic awareness of the range of skills needed to develop software (Grover, 2018).

Relatively few case studies were found when reviewing the literature on CT at a school level. Lye and Koh (2014) conducted a review of computational thinking and found only nine case-studies on various aspects of CT for school level students. Of the nine identified, three were aimed at kindergarten and one aimed for the learning impaired (Lye & Koh, 2014); these were all not considered similar enough for the focus of this thesis to review. The remaining five case studies gave little information about the nature of the code students wrote and predominantly reported general observations in students' use of programming concepts. Nevertheless, some of the challenging CT concepts were identified, which helped to highlight areas of programming and CT beginners at a school level find difficult. In summary, these studies in the Lye and Koh (2014) review have generally indicated across a range of programming languages and environments that students could find the following CT concepts particularly challenging:

- Conditional logic in Scratch (Burke, 2012) and Stagecast Creator (Denner, Werner, & Ortiz, 2012)
- Variables in Scratch (Lee, 2010; Burke, 2012) and Stagecast Creator (Denner et al., 2012)
- Events in Scratch (Lee, 2010), and Boolean logic in Scratch (Burke, 2012).

None of these studies highlighted the programming concepts of sequences and loops to be challenging for participants to implement.

It was difficult to compare similarities and differences with other case studies since because they had entirely different foci. For example, Kafai et al. (2014) used a CT assessment framework by Brennan and Resnick (2012) for an electronic textiles approach to programming in high school and found the CT practice of remixing was important to students' learning and creative expression. Kafai et al. (2014) observed some students found the testing and debugging process in programming to be challenging. Moreover, Allsop (2018) in a case study on CT with student participant programming in Scratch and Alice 2.4 found most student code had the following overall findings:

- no student used custom built functions
- sequences were used in a complex way by almost all students
- confident use of parallelism and conditionals
- little use of operators
- most projects used loops
- variables were used in around half of all projects.

With the focus of this thesis being on the Sonic Pi platform for creating music compositions, comparisons with all the reviewed case studies on CT at a school level are difficult because of the highly unique differences between the programming languages used. Additionally, the types of projects (for example: games, interactive stories, and e-textiles) and length of units varied considerably (from weeks to a whole school year), which significantly changed the level of sophistication expected from participants for each study. Therefore, the reviewed literature can only give tentative indications on what CT concepts and practices students found challenging and successful to support learning outcomes at a school level.

The identified challenges in these studies may also reinforce a limitation when developing metacognitive skills in young people. Research has shown the abilities of self-monitoring in memory, comprehension and other cognitive skills do not commonly develop until early adolescence (Flavell, 1979). CT requires metacognitive abilities to sequence instructions in logical order with algorithms, and reflect on what happens when testing to solve problems (Cox, 2005; Grover & Pea, 2013; Wing, 2006). Thus, the various challenges identified in the reviewed literature may be a result of limitations in developing metacognitive abilities in young people. Therefore, the ideal time to start teaching CT is in middle school (around 12 or 13 years of age), and efforts to integrate CT earlier run counter to the literature on metacognition by Flavell (1979).

Assessment of CT. To further complicate the integration of CT in the NZ DTC, a weakness exists in the literature of established assessment frameworks and approaches due to the low volume of CT research at a school level (Allsop, 2018; Brennan & Resnick, 2012; Lye & Koh, 2014; Selby, Dorling, & Woollard, 2014). The creators of Scratch have conceptualised a possible framework for researching and assessing CT after several years of studying the online Scratch community (Brennan & Resnick, 2012). This research and assessment approach to CT has been also used as a theoretical framework in research for the review of CT case studies at a school level by Lye and Koh (2014) and another case study on programming with electronic textiles by Kafai et al. (2014). The Brennan and Resnick (2012) framework involves the three key dimensions *computational concepts*, *computational practices*, and *computational perspectives* to collectively cover the range of knowledge and skills required to develop CT. A recent study on assessing CT by Allsop (2018) with thirty children aged 10-11 reinforce it is not possible to assess CT solely through CT concepts, and emphasized the role of metacognitive skills that involve CT practices and CT perspectives. Allsop (2018) proposes a CT assessment model with *metacognitive practices*, and *learning behaviours*, which are reflective of the *practices* and *perspectives* dimensions in Brennan and Resnick's (2012) CT framework.

Lye and Koh (2014) have also argued this framework by Brennan and Resnick (2012) to be appropriate for assessing computational thinking because it is in agreement with the Logo programming language (Scratch adopted much of the design and research conducted on Logo). Therefore, it is backed by decades of pedagogical research at a school level since the 1970s. Moreover, Scratch is appropriate for learning in schools because it is designed with "low-floor and high-ceiling" principles (Lye & Koh, 2014, p. 54), which enables beginners to start easily and more advanced students to develop their understanding to a high level of sophistication. Thus, the CT framework by Brennan and Resnick (2012) will likely be suitable for the purposes of this study as Sonic Pi shares the low-floor and high-ceiling design of Scratch (Aaron, 2016).

However, Brennan and Resnick (2012) found CT is not easily and convincingly evidenced in a single test or project outcome. For a student to describe *how* they formulate a solution to an example problem, they need to practice the skills of metacognitive reflection in their own thinking processes. Familiar skills like catching a ball and riding a bike are often difficult to reflect on and describe, therefore, capturing evidence of CT development is likely to be demanding for adolescents (Denning, 2017; Lye & Koh, 2014). To overcome this difficulty, assessment recommendations from various studies offer promising solutions, for example:

- check-ins at multiple points as recommended by Brennan and Resnick (2012)
- on-screen recordings (Lye & Koh, 2014)
- and a think out loud protocol (Vivian, Lozanovski, & Falkner, 2017).

These approaches indicate a shift towards performance based assessment practices, which are recommended by Denning (2017). Moreover, these recommendations for CT assessment are similar to what a professional computer scientist may expect at a professional programming interview, where candidates can be asked to think out loud on their process to solve a given problem (Google, 2016). Thus, the assessment framework by Brennan and Resnick (2012) provides a holistic way to assess CT for the Sonic Pi platform, which will be employed for this research and outlined in Chapter 3. This research is needed because the current version of the DTC provides *progress outcomes* at each year level but no guidelines on how CT can be assessed from student work (MoE, n.d.c).

2.1.3 CT is for all subjects. A common misconception with CT is that computation and algorithms need to be executed by the digital devices we are familiar with today (Denning & Tedre, 2019; Wing, 2011). CT can be broadly described as the thinking process for humans to express themselves through a computational agent, which do not need to be digital (Shute et al., 2017; Tedre & Denning, 2016). For example, Ada Lovelace is often credited as the world's first programmer with mechanical machines in the 19th century (Başer, 2013). Humans can be a computational agent as well; for example, *Computer* was a job title for women at Harvard University (also in the late 19th century) who looked over photographic plates of the night sky and compared the positions of stars (Howell, 2016). Moreover, in a similar way to the rationale behind teaching the basics of science in school, *intradisciplinary* topics in computing (like Human Computer Interaction or HCI, artificial intelligence, computer vision, algorithm design and many more) are also generally useful to teach because of the increasing importance of digital technologies in our daily lives (Denning & Tedre, 2019). Thus, the skill of CT can also go beyond learning programming language syntaxes to a much broader meaning of the thinking processes involved when solving problems for computers (including non-digital computers). This flexibility suggests that CT may be used to benefit problem solving in other subjects through the power of computation that Papert originally envisaged (Anderson, 2016; Papert, 1980).

Another common justification for interdisciplinary CT is that many supporters claim it as a generic skill necessary to prepare everyone for the future workforce regardless of their chosen profession (Barr & Stephenson, 2011; Mohaghegh & McCauley, 2016; Wing, 2011). These arguments are often rationalised through citing reports predicting technological innovation will change the nature of all work and outdate many jobs the current education system is training children for (World Economic Forum, 2018; Frey & Osborne, 2017). Thus, interdisciplinary CT is promoted in schools because it is anticipated to improve students' future employability (Wing, 2011). However, teachers are often confused on specifics about how to develop CT in both an interdisciplinary and effective way (Denning, 2017).

Another related narrative in support of CT for all subjects is the gender and diverse cultural underrepresentation issues pervading computing education and the computing industry (Brown, 2016; Google & Gallup, 2016; Margolis & Fisher, 2002). For example, many studies on the gender balance in computing subjects found high school girls do not consider enrolling in these subjects because of many associated negative stereotypes (Brown, 2016; Google & Gallup, 2016; Gough-Jones, 2008; Margolis & Fisher, 2002). Consequently, Abbiss (2011) warns that as business and industries become software driven at an accelerating pace, a male-dominated and culturally skewed world view could be promulgated through the software we all use. Algorithmic bias can manifest in algorithms that control what we get access to, which may not be fair and neutral because they are designed by humans with their own set values and targeted goals (Garcia, 2016). For example, a research group called the Algorithmic Justice League at Massachusetts Institute of Technology Media Lab found that major face recognition algorithms uncovered severe gender and skin shade bias in the classification of people (Buolamwini, 2017; Buolamwini & Gebru, 2018). If computer algorithms control more of our lives in the future, then an emphasis on promoting a diverse workforce in computing will become increasingly important for education to help mitigate against inequities like these.

To help prevent these inequities, Abbiss (2008) has suggested computing subjects should be broadened to help break down the perception that only “computer nerds” (p. 11) are capable of pursuing knowledge in this domain. This suggestion supports the need to find common ground with other school subjects. The idea of broadening the computing curriculum with interdisciplinary links has been echoed by Kafai and Burke (2013) in their conceptual paper on CT. Thus, teaching other subjects with the support of CT could be an efficient way to meet learning outcomes for both subjects simultaneously, which saves time and helps to overcome the problem of fitting CT into an overcrowded curriculum (Barr & Stephenson, 2011). Specifically, interdisciplinary links between music composition and programming could help expose students to computing through a subject they are already passionate about (Bell & Bell, 2018; Burnard et al., 2016). This opportunity to link subjects like music may also help to improve participation in computing subjects as a creative and enjoyable introduction to programming (Burnard et al., 2014).

Interdisciplinary challenges. Despite the potential of CT as a problem solving skill to support any domain, evidence of skill transfer to learning outcomes in subjects other than computing and mathematics have yet to be substantiated at a school level (Denning, 2017; Doleck, et al., 2017). Historically, there was also little evidence found in the 1980s that learning programming with the introductory programming language Logo helped students' math or general problem solving skills (Klahr & Carver, 1988). Therefore, it is currently unclear how CT in non-computing subjects will benefit learning outcomes in other domains at a school level.

Furthermore, instances Wing (2008) uses to illustrate how CT benefit other disciplines are often with high level ground breaking research: for example, sequencing the human genome through the shotgun algorithm. These examples provide instances where experts at the edges of known knowledge use computation to make advances in the science and engineering fields. Wing (2008) does provide possibilities where students already use aspects of CT in their everyday life, but concrete examples on how non-computing subjects can utilise this skill that support learning outcomes is in its infancy (Grover, 2018; Mannila et al., 2014). To help overcome this issue, Grover (2018) suggests a spectrum from low CT to high CT be established as a progression of learning of year levels in schools (yet to be published). Consequently, many have recommended more case studies are needed on the integration of CT with other subjects in regular school classrooms (Brennan & Resnick, 2012; Lye & Koh, 2014; Grover & Pea, 2013; Wing 2008).

Another major challenge for inexperienced students and teachers is knowing *when* it is appropriate to use CT. Papert (1980) originally promoted a rich variety of approaches to solving problems with CT being only one of them and advocated to select the most appropriate for the task. A danger exists with the promotion of CT for all subjects that it will be applied in inappropriate contexts without an experienced understanding of what it is and when it is useful. For example, isolating and putting data into algorithms can limit understanding in qualitative contexts (Blackwell, Church, & Green, 2008; Denning, 2017; Resnick, 2017). With the definition of CT being unclear, these methodological tensions are likely to cause issues in schools between CT and disciplinary thinking native to each subject (Tedre & Denning, 2016). However, many disciplinary metacognitive thinking skills likely share overlaps of reasoning and problem solving with CT (Grover, 2018; Barr & Stephenson, 2011). Thus, the benefits that CT can offer each subject and year level need to be made clearer through more case study research at a school level.

2.1.4 Interdisciplinary links between music composition and programming. Webster (1990) believes that the creativity in music composition involves problem solving with sound, which creates a link between music and CT. Interdisciplinary CT with music composition and programming could offer many opportunities to support learning outcomes in both domains, with overlapping concepts like pattern recognition, abstraction, algorithmic thinking and decomposition (Aaron, 2016; Bell & Bell, 2018; Edwards, 2011; Ruthman, Heines, Greher, Laidler, & Saulters II, 2010). For example, counterpoint in classical music is the skill of constructing voices that are interdependent yet are independent in rhythm and contour (Fux, Mann, & Edmunds, 1971). Training in counterpoint has been developed as a widely used pedagogical tool in a book by Johann Joseph Fux called Gradus ad Parnassum (Fux et al., 1971). In this book, students progress through several 'species' as music composition exercises of increasing complexity and calculate intervals between notes within set rules for the development of harmonically strong polyphony. The thinking involved in intervallic calculation between voices is just one example that may develop transferable and overlapping CT skills in music composition. Another possible link was claimed by Leung (2004), who suggested that music composition requires effective metacognitive reflective abilities to develop and refine a piece of music to a high standard. Thus, it is possible that transferable and overlapping skills in CT can support learning outcomes in programming and music composition at a school level, which have yet to be substantiated in the literature.

The Sonic Pi platform. The Sonic Pi platform is open-source software designed to enable anyone to learn programming in an interdisciplinary way through coding music (Aaron, 2016). Sonic Pi was developed by Sam Aaron in collaboration with the Raspberry Pi team and has been reported to be downloaded more than 1000 times per week (Burnard et al., 2016).

To date, there has been a paucity of research on the teaching of Sonic Pi. However, student participants in three scenarios bound together in a research project called *Sonic Pi: Live & Coding* (Burnard et al., 2014) has been conducted. In this study, experimental units of work were made into both formal (school music education classrooms in six-week blocks) and informal (a five-day summer school at a performing arts venue) learning environments. Burnard et al. (2016) found a synergistic relationship between music and programming, which also engaged the participant students to a high degree. Further positive reports claimed that learning in maths, music, CT and programming took place: "While coding and composing, students often didn't even realize that they were learning mathematics and computational thinking in our environment" (Burnard et al., 2016, p. 347). However, it is unclear *what* students learnt and *how* these disciplines were assessed from the data gathered (observations, interviews, and student artefacts). Furthermore, a student voice of what they explicitly learnt in each domain was lacking in their reflections beyond general positive reports of increased enthusiasm and improvements to their learning (Burnard et al., 2014). If teachers in schools are to use the Sonic Pi platform for the assessment of learning outcomes linked with the curricula of DT and music, then an investigation of robust assessment practices in each domain using the Sonic Pi platform needs to be studied with a designed unit of work for this purpose.

Emphasis in these studies was placed on *live-coding*, where students made music in-front of an audience as a form of performance. Live-coding is a new activity for schools, however, it does not fit neatly into traditional assessments in music (because it is both composition and performance combined) and programming (because students are under pressure to *perform* to a set of requirements) (Burnard et al., 2014). Live-coding performance may become an overly stressful experience for an introduction to music composition and programming (Blackwell, McLean, Noble, & Rohrhuber, 2014). This issue could be avoided for beginners through first focusing on music composition for a non-performance based project.

Music composition in middle school. The current Music Curriculum (MC) was introduced in the year 2000, which helped to strengthen the place of music composition in schools (McPhail, 2012; MoE, 2000). However, many music teachers have had little training and experience in composition (McPhail, 2012). Consequently, understanding on how to best teach creating the vast range of music styles is relatively underdeveloped at a school level relative to music performance. However, group music composition has been found as an effective way for students with little experience and confidence to engage in this creative activity (Thorpe, 2017). Cloud based technologies and phone applications have also revolutionised music composing to the extent that many consider it to be a recreational activity for young people outside of school (Aaron, 2016). Therefore, many students today may already be aware of common music making practices on computers.

A music composition software package on Apple computers called GarageBand® is being used in many schools in New Zealand as an introduction to making, recording, and arranging music on computers (Bolton, 2008). This software is partly intended as an easy introduction to music composition, which is available for free on Apple computers since 2004 (Väkevä, 2010). However, little research has been conducted on the best ways to align activities using GarageBand with the New Zealand MC. Thus, comparisons of studies between GarageBand and Sonic Pi on challenges and successes with both platforms to meet learning outcomes in the MC are difficult.

GarageBand comes with many pre-programmed sequences, pre-set effects, and loops that make it easier to get started in making digital music for those with no music training (Ricket & Salvo, 2006). Some music teachers describe the use of these pre-made resources in GarageBand as "just cutting and pasting", "not a composition tool", and "not composing" (Wise, Greenwood, & Davis, 2011, p. 126). However, others thought GarageBand allowed for more freedom and creativity because it was perceived as easier than traditional music notation software for composition (Wise et al., 2011). One major pedagogical benefit for music education that Sonic Pi has over GarageBand is it unlocks understanding of these pre-sets for making music on computers because students have to create these pre-sets from scratch themselves (Aaron, 2016; Aaron & Blackwell, 2013). Therefore, this notable benefit of Sonic Pi supports the motivation behind the DTC for students to become *creators* of digital technologies.

2.1.5 Conclusion. CT is being promoted as an essential skill because: (a) it is predicted to give a competitive advantage for the modern workforce; (b) in a similar way to the rationale behind teaching science in school, *intradisciplinary* topics related to computing (like artificial intelligence and algorithm design) are also generally useful to teach in schools because of the increasing importance of digital technologies in our daily lives; and (c) it highlights that programming involves more than demonstrating knowledge of programming syntaxes. However, definitional issues around CT create a lack of clarity in its scope and application at a school level. In particular, there are problematic debates over whether programming is essential to CT (Denning, 2017) and if transferable problem solving skills to non-computing subjects are possible (Mannila et al., 2014). However, commonly agreed upon elements of CT like abstraction, problem-solving, algorithms, and thinking from the literature review by Kalelioğlu et al. (2016) provide some guidance for educators that are widely agreed upon.

Despite these definitional issues, the New Zealand Government has started the implementation of DTC in 2018 to be made compulsory for all schools and levels in 2020 (MoE, n.d.c). Insufficient teacher capability and time to upskill for these changes likely mean that the integration of CT will be implemented at a low standard, which may negatively influence the attitudes of teachers and students. Furthermore, literature on developing metacognitive skills in young people have previously indicated limitations, with development only beginning to be possible in the early adolescent years (Flavell, 1979). Therefore, more case studies on CT at a school level have been called for to understand these challenges more deeply (Denning, 2017; Lye & Koh, 2014; Grover & Pea, 2013).

There have been relatively few case studies at a school level on CT. Those found were conducted for a wide range of purposes and programming languages (Allsop, 2018; Kafai et al., 2014; Lye & Koh, 2014), which makes it hard to compare findings between them. However, patterns across these studies indicate students find the challenging CT concepts are variables, conditional statements, Boolean logic, functions, and events.

These challenges may also reinforce studies by Flavell (1979) that metacognitive skills only start to develop in the early adolescent years.

CT has the potential to benefit other disciplines, which may help increase participation from underrepresented groups in computing related professions. Increasing participation of underrepresented groups in computing will become essential as equitable issues are likely to become more of an issue in the future (Buolanwini, 2017; Garcia, 2016). Abbiss (2008) suggests broadening computing subjects to increase participation of girls in computing. CT being taught in an interdisciplinary way could become a key link to computing with other subjects to enable this wider scope. However, transferable CT skills in other non-computing subjects have yet to be substantiated at a school level (Doleck, et al., 2017; Harel & Papert, 1990; Tedre & Denning, 2016). Therefore, more case studies on how CT can support learning outcomes in other subjects at a school level are needed.

Another challenge with the implementation of the DTC in NZ, is that there is little information for educators on how to effectively assess the development of CT. However, Brennan and Resnick (2012) published a holistic approach to assess CT, with the addition of the practices and perspectives dimensions. This framework has also been recommended by Lye and Koh (2014) in their review of 27 studies on CT and used in a case study on CT by Kafai et al. (2014). However, CT was still found to be difficult to assess using traditional forms of assessment with a single test or project outcome (Allsop, 2018; Brennan & Resnick, 2012; Lye & Koh, 2014). Recommendations to use check-ins at multiple points and think-out-loud protocols provide suggestions for possible ways to overcome these issues (Lye & Koh, 2014). This CT framework by Brennan and Resnick (2012) is appropriate for assessing CT with the Sonic Pi platform because it shares the low-floor and high-ceiling design principle of Scratch (Aaron, 2016). For these reasons, the Brennan and Resnick (2012) CT framework and recommendations by Lye and Koh (2014) have been employed into the methodology of this thesis.

Interdisciplinary CT could support learning outcomes in programming and music composition with the Sonic Pi platform (Burnard et al., 2014), which may help to save class time in school to deliver CT (Bell & Bell, 2018). However, a paucity of research has been conducted on Sonic Pi, which are also limited because they do not indicate what and how students were achieving learning outcomes in both subjects (Burnard et al., 2014). Furthermore, the focus on live-coding with Sonic Pi create assessment challenges (Burnard et al., 2014). Simply focusing on music composition could be a valid way to overcome these challenges. Thus, further case study research is needed to understand how CT can support learning outcomes in music composition and programming with the Sonic Pi platform.

Overall, this literature review on CT indicates research into interdisciplinary CT and Sonic Pi is not well understood at the school level. Therefore, robust case study research in a school setting is needed to help understand how to best implement the DTC in New Zealand. Consequently, this thesis proposes to expand the research on the Sonic Pi study by Burnard et al. (2014), and the research on CT reviewed by Lye and Koh (2014) with a mixed-method case study. Moreover, this literature review found the assessment framework by Brennan and Resnick (2012) provides an appropriate framework to assess CT with the Sonic Pi platform at a school level, which is employed as a theoretical framework in this thesis.

2.2 Literature on the Sub-Research Question

Studies on attitude recognise its importance as playing a crucial role in the aims of many educational endeavours relating to computing subjects (Özyurt & Özyurt, 2015; Başer, 2013; Kong, Chiu, & Lai, 2018). Fishbein and Ajzen (1975) defined attitude as feelings (positive or negative) towards behaviours that could affect aspects contributing in the success of a behaviour's aim; for example, enjoyment, anxiety, and importance (Teo, 2007). Research from many disciplines for various educational contexts reinforce more broadly that students' attitudes are a main factor affecting learning efficacy (Bovée et al., 2007; Özyurt & Özyurt, 2015; Teo, 2007). Moreover, a positive attitude may be critical around Years 7 and 8 in school because these years dominantly inform students' enrolment choices in senior high school and university (Fronza et al., 2017; Grover et al., 2016; Margolis & Fisher, 2002).

More exposure to computers in middle school may lead to increased self-confidence (Teo, 2008), which could improve more positive attitudes towards computing subjects (Barron, Walter, Martin, & Schatz, 2010). The case studies conducted on Sonic Pi have reinforced this idea and generally indicated that it promoted positive attitudes in young people to music and programming both in and out of school contexts (Burnard et al., 2014; Cheng, 2018). For example, an interviewed student after experiencing a unit on Sonic Pi said “It [Sonic Pi] just makes people happy” (Burnard, et al., 2014, p. 19). However, no pre and post measures were collected, therefore, comparisons demonstrating how creative activities with the Sonic Pi platform helped to promote more positive attitudes in programming were not possible. The study was also conducted closely with the creator of Sonic Pi (Sam Aaron), thereby having a likely confirmation bias towards positive findings, which was found to be an issue for studies on the Logo programming language with the involvement of Seymour Papert (Klahr & Carver, 1988). Thus, more rigorous independent studies are needed to help understand the changes that may occur on attitudes towards programming as a result of exposure to Sonic Pi.

There are relatively few studies conducted for school-aged students when compared with studies at a university level on relating attitudes towards programming achievement (Başer, 2013; Karaci, 2016). The scope, rigour, and challenges of learning programming at a school level are vastly different to learning this skill at university level (Özyurt & Özyurt, 2015). Furthermore, studies on feelings towards computer programming have found it is commonly perceived as a complicated and unappealing skill to learn (Anderson et al., 2008; Margolis & Fisher, 2002). Many previous studies on measuring programming attitudes have largely focused on gender differences, with the majority reporting that males generally have more self-confidence that females (Başer, 2013; LaBouliere, Pelloth, Lu, & Ng, 2015; Papastergiou, 2008; Vekiri, 2010). This difference often results with males reporting more positive attitudes towards programming, which has been shown to be positively correlated to academic performance in programming with university students (Başer, 2013). Moreover, Cazan, Cocoradă, and Maican (2016) found that attitude towards computers is a significant contributing factor in the extent students use them for learning, which has been reinforced by Teo (2006; 2007). These studies collectively indicate that negative attitudes can be a barrier for young people learning to program on computers.

2.2.1 Measuring attitude. Reliable quantitative instruments measuring attitudes towards programming have not been widely developed and studied at a school level (LaBouliere et al., 2015; Kong et al., 2018). The majority of instruments used to measure attitudes towards computer programming are self-developed for university students, with few that have reinforced their validity and reliability in other contexts (Karaci, 2016; Özyurt & Özyurt, 2015). Furthermore, the studies on Sonic Pi by Burnard et al. (2014) only gathered qualitative post unit of work data from short semi-structured interviews with five students. Thus, a gap in the literature exists on reliable instruments for quantitatively measuring attitudes towards computer programming at a school level.

However, research on attitudes towards general computer usage has been studied relatively often in a variety of settings (including school-aged participants) with instruments that have found to be reliable (Cazan et al., 2016; Bovée, Voogt, Meelissen, 2007; Teo, 2006). For example, the computer attitude questionnaire developed by Knezek, Christensen, and Miyashita (1998) has been found to be reliable with the subscales enjoyment, importance, and anxiety (Teo, 2007;2006). These subscales can be defined as follows:

- enjoyment can be defined as the degree to which the student likes to work on computers
- importance can be defined as the perceived need for the present and future
- anxiety can be defined as the apprehension towards activities that involve computers (Teo, 2007).

These three subscales also link to Fishbein and Ajzen (1975) three attitude elements of affective (associated with the subscale of enjoyment), cognitive (associated with the subscale of importance), and conative (associated with the subscale of anxiety) (Teo, 2007). Moreover, these subscales have been reinforced to be effective to measure computer attitudes for middle school students in South Africa (Bovée, Voogt, & Meelissen, 2007). Because programming in Sonic Pi involves a high degree of computer usage, using this established theoretical framework from questionnaires employing these subscales will help to ensure a valid and reliable instrument is created to measure changes in attitudes towards programming for this study.

2.2.2 The importance of creativity in programming. The development of students' creativity is seen as important by many education experts (Davies et al., 2013; Kafai et al., 2014; Peppler & Kafai, 2009; Resnick, 2017; Vekiri, 2010). This is because the issues current students will have to work on in the future are likely to be large and complex, which will require creative approaches to solving problems (Resnick, 2017; Frey & Osborne, 2017). Creativity can commonly be defined as the making of both original and useful outcomes (Runco & Jaeger, 2012). In an educational context, Resnick (2017) suggests that creativity involves the novel connection of ideas that diverge from established solutions, which can be made from iterating through a *creative learning spiral*:

1. Starting an idea
2. Building prototypes
3. Sharing work and getting feedback
4. Running experiments
5. Revising these experiments based on feedback received (Resnick, 2017).

Moreover, activities that involve the freedom to be creative are said to help promote more positive attitudes as they enable students to align their interests and passions with the activity (Davies et al., 2013; Guzdial, 2003; Papert, 1980; Resnick, 2017; Vekiri, 2010). Thus, the creative activity of composing music with Sonic Pi could help to promote more positive attitudes to programming because it allows students to arrange and modify sound aligned with their intrinsic preferences through code.

However, educators need to be aware of romantic preconceptions of creativity with students (Glăveanu, 2018; Edwards, 2011). Idealist student preconceptions on creativity may result in reduced self-confidence in learning when engaging with creative activities because of a perceived *right* way to do it. Resnick (2017) provides some guidance here for educators, where he promotes the task design principle of *wide-walls*, which aim for a diverse range of outcomes from students (rather than many similar outcomes). He states that if student outcomes are too similar, then the growth of creativity could be stunted as the *walls* or requirements of the task are too restrictive. Therefore, creative activities designed with a wide-wall principle will be beneficial to develop for the successful integration of the DTC. The Sonic Pi platform allows for a *wide-walls* principle because students can learn programming as a highly creative activity with the freedom to explore and experiment through making music with code (Arron, 2016).

2.2.3 Conclusion. Attitude has been established as important because it potentially influences the outcomes of educational endeavours (Fishbein & Ajzen, 1975). Students' attitudes towards subjects in school is also especially important around Years 7 and 8 because it significantly impacts their direction in senior high school and university (Abbiss, 2008; Margolis & Fisher, 2002). However, previous studies on attitudes towards programming have predominantly been at a university level with self-developed instruments that focus on gender stereotypes (LaBouliere et al., 2015; Başer, 2013). Few reliable quantitative instruments on measuring programming attitudes have been developed at a school level (LaBouliere et al., 2015; Kong et al., 2018). However, attitudes toward general computer usage have been studied and found to be reliable with the subscales of enjoyment, importance, and anxiety (Teo, 2007). This framework has also found to be reliable multiple times in other contexts (Bovée et al., 2007; Teo, 2006), which could be adapted for a programming context at a school level for the purposes of this thesis.

Creativity is also being advocated as an important to skill to develop in the future for increasing student interest and participation in computing subjects from more diverse groups (Resnick, 2017). The creative activity of making music with the Sonic Pi platform may potentially improve more positive attitudes towards programming (Burnard et al., 2014; Burnard et al., 2016). However, Glăveanu (2018) warns educators of idealist preconceptions of creativity, which may influence students' sense of self-confidence in creative activities. To counter this preconception, Resnick (2017) promotes the task design principle of wide walls that emphasises a diverse range of creative responses from students. Sonic Pi enables creative activities in programming with a wide walls principle through the diverse ways of making music compositions and enabling students to engage with their intrinsic preferences to select and arrange sound (Aaron, 2016). Thus, the creative activity of making music compositions through programming in Sonic Pi has the potential to improve more positive attitudes towards programming.

Studies have found positive attitudes towards programming were recorded from participants through exposure to Sonic Pi (Burnard et al., 2014). However, these findings are likely to be subject to confirmation bias because they were closely conducted with the creator of Sonic Pi, which was also discovered to be an issue when comparing findings for the Logo programming language with the involvement of Seymour Papert (Klahr & Carver, 1988). Additionally, no pre and post measures were collected in the study by Burnard et al. (2014), therefore, comparisons demonstrating how creative activities with the Sonic Pi platform helped to promote more positive attitudes in programming were not possible. Therefore, independent and rigorous studies are needed on Sonic Pi, which measure pre and post changes in student attitudes.

Overall, this literature review for the sub-research question suggests creative activities with the Sonic Pi platform have the potential to promote more positive attitudes towards programming. This potential may support the successful integration of the DTC in New Zealand. However, robust case studies on attitudes towards programming are lacking in the literature. Thus, this thesis hopes to expand our understanding through case study research on how the creative activity of music composition using Sonic Pi can promote more positive attitudes towards programming.

Chapter 3: Methodology

The two aims of this study were: (a) to determine how Computational Thinking (CT) can support Year 8 students' learning outcomes in programming and music with the Sonic Pi platform (main question); and (b) to determine what extent the creative activity of music composition using the Sonic Pi platform can help to promote more positive attitudes towards programming (sub-question). To answer both questions, a mixed-method case study using an experimental unit of work designed by the researcher was used to frame the methodology of this research.

A key advantage of case studies is they can illuminate a detailed snapshot of reality (Cohen, Manion, & Morrison, 2011). A case study is an appropriate methodology for this research because it closely investigated a unit of work in a school classroom. The participants involved were volunteers in an existing compulsory Year 8 music class with 22 students (10 female and 12 male) and their music teacher. The research site was an urban state primary and intermediate school in a well-resourced music classroom with 25 desktop computers that could run Sonic Pi.

Creswell (2014) recommends case studies use a mixed-method approach to support the different strengths and weaknesses of both qualitative and quantitative methods. A mixed-method approach may also provide a higher degree of validity and reliability through the merging and triangulation of data. For these reasons, this case study used a mixed-method methodology to answer both research questions. Mixed-method methodologies are relatively new to educational research and were originally founded in psychology (Cohen et al., 2011). Creswell (2014) states that many terms have been used for this approach; for example: multimethod, integrating, and synthesis. Since then, mixed-method studies have developed in education research with its own distinguishable method of inquiry (Creswell, 2014).

The designed unit of work used an action research methodology. Cohen et al. (2011) state action research is about the improvement of a study's focus (with collaboration with its participants) to bridge the gap between research and practise. In this thesis, the teaching methods and learning strategies designed for the unit of work aimed to improve the educational outcomes and experiences for the participants involved. However, the scope of this research is small and its main purpose is to increase understanding of both research questions. Cohen et al. (2011) recommend an eight stage workflow for designing and conducting action research, which was followed in this study (described under the following sub-headings in this Chapter).

3.1 Design of the Unit of work

The unit of work was developed by the researcher prior to selecting the specific participants for this research and designed for a class of beginner programmers and music composers at a Year 8 school level. It first adopted aspects from Phase 2 of the Sonic Pi Live and Coding Research Project (Burnard et al., 2014) in the following four ways:

- it was designed for Year 8 students
- it consisted of six one-hour and 40-minute lessons in their regular school music class with their music teacher
- it incorporated projects with students grouped in pairs
- it had a final presentation of students' work.

Adopting these aspects enabled comparisons to be more easily drawn with the study on Sonic Pi by Burnard et al. (2014). Furthermore, the decision to focus on Year 8 students is reinforced with existing literature on metacognition, because early adolescence has been found as the age most young people start developing a skill like CT (Flavell, 1979).

Many positive findings were reported from the Sonic Pi Live and Coding study, but it was unclear specifically what and how the participants were learning (Burnard et al., 2014). Therefore, three major modifications and additions were made to align with the research questions in this study:

- learning outcomes were created that aligned with the new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC) and the Music Curriculum (MC) in New Zealand (NZ) for the main research question

- it integrated measures for assessing student attitudes towards programming for the sub-research question
- students focused on music composition (rather than live-coding performance).

The decision to focus on music composition was made because the pressure to perform with live-coding was likely to be a stressful experience for beginner students. For example, negative experiences were reported in the previous study on Sonic Pi where two female students said performing on stage was “a bit scary” because the boys “booed” the girls when their code did not work (Burnard et al., 2014, p. 23). Moreover, Burnard et al. (2014) found assessment challenges in using live-coding for traditional forms of assessment, which are difficult to align with the DTC. These issues can be avoided for beginners through a focus on music composition with the Sonic Pi platform instead.

After these considerations were made, the overall design of this unit of work employed the four principles of projects, passion, peers, and play advocated by Mitchel Resnick (Resnick, 2017). These principles could help with the successful integration of the DTC through making learning more meaningful and engaging (Petrie, 2018). Projects, passion, peers, and play were predominantly applied through a music composition brief students were assigned to complete by the end of the unit of work. Students worked on two music composition projects based on this brief (one individual and one in pairs).

Pair programming and music compositions in pairs. Pair programming has shown to be effective for sharing knowledge between students in middle school classrooms (Werner & Denning, 2009) and in higher education music technology courses (Moore, 2014). Moreover, CT thought processes were more likely to be revealed through student interactions if they worked in groups to help provide data for the main research question (Brennan & Resnick, 2012; Lye & Koh, 2014). The activity of pair programming has one student on a keyboard coding, while the other reviews their code (Werner & Denning, 2009). Additionally, group music composition has also been found as an effective way introduce music composition to students for those with less confident music ability (Thorpe, 2017). Thus, this unit of work had students work on a music composition in pairs, which utilised these benefits of both pair programming and group music composition.

Individual composition. Each individual contribution in group work has been evaluated as difficult to assess and track (Van Aalst, 2013). Therefore, to more effectively track individual differences between students for the main research question, they were also assigned an individual music composition based on the same designed project brief. Student work from this project helped to understand if the set learning outcomes were able to be met by each student.

3.1.1 Project brief. The Level 4 DTC emphasise authentic projects for both strands (Ministry of Education [MoE], n.d.c). Resnick (2017) also recommend authentic projects be given to students to increase student engagement and interest in programming. The designed project brief had students create two music compositions for videos on topical issues facing the world today based on the United Nations 17 Sustainable Development Goals (SDG) (United Nations, n.d.). This authentic purpose of composing music in response to current events was intended to engage students in highly individual musical responses to the topics depicted in the videos.

Three video topics were chosen based on the 17 SDGs: climate change (goal 13), plastic in the oceans (goal 14), and the refugee crisis (goal 16). Because of the limited time assigned to this unit of work, videos were required to be between 1 and 2 minutes in length. Therefore, the videos were selected from short public service advertisements freely available on YouTube (www.youtube.com) featuring an aspect of the problem for each sustainability goal (see Appendix J for a links to these). In total, five videos were curated based on these topics from the organisations Greenpeace, UNICEF, and the United Nations Refugee Agency. It was not deemed necessary to seek permission to use these videos because they were not used outside of the classroom, not downloaded, and were freely available for the public to watch online.

The terms 'project' and 'music composition' will be referred to throughout this thesis. The term 'project' refers to the brief students were given (see Appendix B) and 'music composition' will refer specifically to the audio output from code students wrote in Sonic Pi.

Algorithmic music. Sonic Pi code is unique in comparison to traditional forms of music composition in popular genres (Burnard et al., 2016). The advantage for music composition through Sonic Pi is programming can leverage the computer's power to execute algorithms (Edwards, 2011). Making music compositions through designing algorithms provides a deep link between music and programming (Burnard et al., 2014). As Edwards (2011) observes, algorithmic music composition has been in the Western music tradition for over a millennium. For example, a piece called Musikalisches Würfelspiel by Wolfgang Amadeus Mozart (1793) generates short compositions from harmonically compatible music fragments that are randomly selected with the roll of dice. Thus, an algorithmic style of music composition was made a focus of the unit of work through the design of the lesson tasks because of these links between music and programming.

3.1.2 Lesson tasks. It was anticipated that technical problems could become an obstacle in the beginning lesson. Therefore, Lesson 1 introduced only a few simple commands that aimed to get students comfortable with the Sonic Pi platform. Three main tasks were then selected that were deemed appropriate for Year 8 beginners in music composition and programming. These tasks were the main foci in Lessons 2, 3, and 4 as follows:

- *Lesson 2:* This lesson focused on giving students the opportunity to make their own music scale (or ladder of notes). For this task, an interdisciplinary link with making a scale in music and learning about lists in programming was intended (see Lesson 2 of Appendix A).
- *Lesson 3:* The second major task engaged students in making algorithmic music with conditions and random number generators. Based on Lesson 2, students could creatively make algorithms that produced a potentially infinite variety of melodies and chords from the scale they made (e.g. the way windchimes make infinite variations within its scale or collection of notes).

- *Lesson 4:* The third and final task required students to sculpt their own electronic instrument or synthesiser with simple audio waveforms (sine, square, triangle). With this final task, features in the Sonic Pi platform were used to sculpt a waveform with its attack, release, sustain, and decay.

Lesson 5 was dedicated to students refining their music composition for a final hand in. Lesson 6 was entirely dedicated to the presentation and sharing of all final music compositions. Thus, the six lessons in the unit plan main activities can be summarised as follows:

1. Lesson 1: Get *comfortable* – Students get comfortable making a few sequences and loops in Sonic Pi
2. Lesson 2: *Scales* – Students create their own unique music scale (ladder of notes) that suits the mood of their chosen video
3. Lesson 3: *Algorithms in Music* – Students explored various ways to make algorithmic music with random numbers and conditional statements
4. Lesson 4: *Synthesisers* – Students created their own synthesizer (electronic instrument) from various waveforms (sine, triangle, square)
5. Lesson 5: *Finalising each project* – Students refine, finalise and hand-in both projects
6. Lesson 6: *Showcase of all* projects – All students' music compositions are presented to the class

Except for Lesson 6, all lessons also planned the following activities:

- Revision of any previous concepts covered at the beginning of each lesson to aid learning retention and inform the teacher of student gaps
- Time for reviewing students' music compositions and for students to give feedback to aid the process of refinement

- Large chunks of time for students to make their music compositions to ensure they had enough time to meet the requirements of project briefs
- Time for end of lesson quizzes and individual reflections before the end of class to inform the teacher of student understanding to aid refinement and optimisation of processes
- Time to introduce any new concepts through a class demonstration from myself to ensure new concepts were appropriately introduced to students
- Students were exposed to algorithmic music listening examples to help inspire and illustrate links between music and programming

After all lessons were sequenced with a progression of learning activities, students had an approximate total time of four hours over five lessons to complete their individual and group music compositions. A detailed view of this arrangement can be seen in the unit plan (see Appendix A, which includes links to all individual lesson plans).

3.1.3 Learning outcomes. Appropriate learning outcomes were then created to be in line with the main lesson activities and curricula links in the MC and DTC (see Appendix A). The process for designing learning outcomes was through identifying elements of Level 4 Curriculum Statements in both subjects (MoE, 2000; MoE, n.d.a) that could fit with achievable and appropriate outcomes by the end of each lesson. These learning outcomes were designed to: (a) develop understanding of programming and music composition through breaking down the skills and knowledge needed into achievable tasks for beginners; and (b) to help the development of their final music compositions. For example, a music learning outcome in Lesson 1 was “All students will compose a melody in Sonic Pi” (see Appendix A), which helped students to think about the CT concept of sequences in programming and melodies as a sequence of notes in music (which feature in both Curricula at Level 4).

3.1.4 Teaching the unit of work: pedagogy and teaching strategies. The teaching pedagogy of constructionism was adopted as recommended by Lye and Koh (2014) for the effective development of CT. In adopting this pedagogy, I aimed to guide students through problems with targeted questioning, rather than providing answers and solutions. As recommended by Grover (2018), I also encouraged students to adopt a 'thinking before doing' approach as opposed to *trial and error* approach to solve problems.

A student directed style of learning like constructionism is most effective when it is scaffolded with guidance to aid understanding of new concepts with beginners (Kirchner, Sweller, and Clark, 2006; Schwartz & Bransford, 1998). Thus, the introduction of new programming concepts in this unit of work utilised a think out loud protocol recommended by Lye and Koh (2014). Live-coding is where someone types code in-front of an audience while voicing their decision-making out loud in real-time – thereby revealing their thinking and problem solving processes for others (Werner & Denning, 2009). This strategy has shown to be effective in demonstrating the process of developing algorithms (Rubin, 2013), which helps to unravel CT for students. This protocol was adopted to guide students with the understanding of new Sonic Pi syntax, programming and music concepts.

3.2 Data Collection

Three instruments and a measurement tool were designed to gather data to answer both research questions:

- pre (T1) and post (T2) unit of work questionnaires (see Appendix D and E)
- pre and post interviews (see Appendix H for interview guides)
- end of class quizzes and student reflections (see Appendix C)
- measurement tool as a marking rubric for all final music compositions (see Appendix B).

3.2.1 Pre and post unit of work questionnaires. Questionnaires are recommended to quickly gather data about participants (Cohen et al., 2011). Pre and post unit of work questionnaires were designed for all participants and had three main sections.

1. The first section (pre-questionnaire only) gathered information about prior experiences in music and programming to help understand if participants' previous experiences influenced results for both research questions.
2. Also for both research questions, the second section helped to assess changes in perspectives on music and programming.
3. Using a Likert scale, the third section's purpose was to measure changes in attitudes towards music composition and programming to help answer the sub-research question. Likert scales are commonly used as a way to quantitatively measure intensity of responses from participants in educational research (Cohen et al., 2011).

The pre and post-questionnaires were modified minimally for the participant music teacher where they focused more on the *teaching* of both subjects, rather than the *learning* designed for student participants. All questionnaires can be viewed in Appendix D and E.

The first section (pre questionnaire only). Because participants' prior experiences were likely to influence learning outcomes and attitudes, this section intended to provide some background perspective for both research questions. Design of the first section's items included seven questions. The first question asked participants their name. Next, three questions followed that asked participants' prior experiences in programming about:

- the length of their programming experience
- what programming language(s) they have used
- if they completed any programming projects.

Then, three items on prior music experiences asked:

- about previous music training and for how long this training was
- if they had made their own music before
- if they had made music on computers.

The second section (pre and post questionnaire). The second section had three items:

- The first two items asked students what their feelings were towards programming and music composition to help record changes in perspective for both research questions
- The third item asked students if they thought programming can be creative to help assess individual thoughts towards creativity in programming for the sub-research question.

The third section (pre and post questionnaire). The reviewed literature found no reliable quantitative instrument that specifically measured attitudes towards programming for middle school students. To help structure this section, a related theoretical framework used by Teo (2007) on computer usage was adapted for this study because programming involves using a computer. This framework had the three subscales of *enjoyment*, *importance*, and *anxiety* to assess different factors that may contribute to an attitude towards using computers. Teo's (2007) reliability estimates found that these subscales had internal consistency and acceptable intercorrelations. These subscales are also linked with the three components of attitude defined by Fishbein and Ajzen (1975) of affective (enjoyment), cognitive (importance), and conative (anxiety). With previous reliability to measure computer usage, these attitude subscales were considered appropriate to measure pre and post differences in programming and music for the sub-research question in this study. However, a *self-confidence* subscale was considered more relevant for the context of this research than the factor of *anxiety* because this study was about how students learn to program and compose music. Therefore, self-confidence replaced anxiety for one subscale from the original framework by Teo (2007). Thus, the three subscales of enjoyment, importance and self-confidence were used for this study.

The sub-research question asked to measure attitudes towards programming only. However, attitudes to music composition were also measured in this section to provide a comparison of similarities and differences with programming. This additional gathered data helped to reinforce the extent of the impact this unit of work had on attitudes for the sub-research question. Thus, design of this section of the questionnaire included 18 items total that measured attitudes towards music composition and programming (nine for each subject). The pre and post questionnaire items for this section were identical.

Questionnaire items to measure attitude. Teo (2007) had between five to eight items for each attitude subscale. From the original questionnaire, only three relevant items for each subscale were selected to modify as to not take up too much lesson time. These three items were modified minimally and reworded to replace aspects of computer usage with *programming or coding* and music composition. For example "I enjoy doing things on the computer" (Teo, 2007, p. 132) was modified to "I think *programming or coding* is a lot of fun" (see Appendix D).

For all music items, the text "programming or coding" was replaced with "music composition" to measure student attitudes towards music composition. This consistency in the instrument for both subjects enabled a comparison to be made between them, which could help to assess the significance of the effect size.

For each subscale, there were two positively worded items and one negative (reverse coded) to reduce acquiescent and extreme response bias (Cohen et al., 2011). However, there is also evidence that reverse items can distort or be misinterpreted by participants (Weijters, Baumgartner, & Schillewaet, 2013), which is a potential limitation of this strategy. The following nine questions were asked for each subscale in programming:

Enjoyment subscale:

1. I think programming (or coding) is a lot of fun.
2. I think programming (or coding) is not enjoyable. *
3. I am excited to learn the skill of programming (or coding)

Importance subscale:

1. I think learning programming (or coding) skills are relevant for my future.
2. I can't see the point in learning the skill of programming (or coding). *

3. I might want to study programming (or coding) in senior high school or university.

Self-confidence subscale:

1. I find (or might find) programming (or coding) easy.
2. Programming (or coding) can only be learnt by really intelligent people. *
3. I feel I know what kinds of activities are involved with programming (or coding).

* reverse coded items

The Likert scale used for this study was also modified from the original Teo (2007) to a six-point Likert scale (from a five-point scale). Six-point Likert scales have shown to be more reliable than 5-point scales in psychology tests (Chomeya, 2010). They also have the advantage of forcing the participant to choose if their attitude towards a given question is positive or negative (Cohen et al., 2011). Thus, the Likert scale used for this section of the questionnaire had six-points as follows: 3 = Strongly agree; 2 = Moderately agree; 1 = Slightly agree; -1 = Slightly disagree; -2 = Moderately disagree; -3 = Strongly disagree.

3.2.2 Pre and post interview guides. Interviews are an effective way to gather data for descriptive insights from participants' perspectives (Taylor, Bogden, & DeVault, 2016). All interviews intended to provide rich descriptions (not provided in the other forms of data gathered) of participant experiences and their learning to help inform both research questions. The music teacher and five pre-selected student participants were invited for a pre and post semi-structured interview with myself (see Appendix H for all interview guides). Pre-selections of the five interviewed students were based on recommendations by the participant music teacher (Zach) because they were students likely to provide in-depth responses from both genders. All interviews were audio recorded and took place in the music classroom on the research site.

Pre-unit of work interviews were designed to take a maximum of 10 minutes for the five selected students and 15 minutes for the music teacher. All post-unit of work interviews were designed to take 5 minutes longer than pre interviews because of the need to reflect in-depth on what happened in the unit of work. As recommended by Cohen et al. (2011) in a semi-structured interview approach, each item could follow with questions related to each interviewee's response. As all interviews were designed to be semi-structured, question items were intended as a flexible framework for the interviewer.

Student pre-unit of work interviews. Student pre-unit of work interviews took place during lunchtime in the week before the delivery of the unit. The questions were structured as follows (see Appendix H):

1. The first three questions were related to the three subscales that were used to measure attitude in the questionnaires for the sub-research question (enjoyment, importance, and self-confidence). These questions were asked to record rich descriptions of existing attitudes towards music composition and programming, which the questionnaires could not provide through responses to a Likert scale. For example, students were asked "Can you describe your feelings towards programming?". Follow-up questions on specific aspects that contributed to their enjoyment or displeasure were then asked.
2. Next, student interviewees were asked if they thought programming or coding can be creative and if they thought of themselves as creative. These questions were to help inform if students had pre-existing mental barriers about the creative activity of music composition with Sonic Pi for both research questions.
3. If the interviewee had prior experience composing music or programming, the last prepared question asked students to describe these experiences and their feelings towards them. The purpose of these questions was to understand a more in-depth

perspective on pre-existing attitudes towards programming and music for the sub-research question.

Student post unit of work interviews. After the unit of work's final lesson, the same students who participated in the pre-unit of work interviews participated in the post-unit of work interviews. Design of the questions were structured in three main parts:

1. The same questions for the pre-unit of work interview were asked so that a comparison could be made (albeit questions relating to prior experiences were not asked a second time).
2. They were asked for their thoughts on the unit of work and programming with Sonic Pi in music class (if students had prior experiences before the unit of work, they were asked to compare both experiences).
3. Questions were asked on pre-selected saved files from their projects that were loaded on a computer in the interview to help capture CT processes, as recommended as a data gathering strategy in studies on CT (Brennan & Resnick, 2012; Lye & Koh, 2014). Code blocks from saved files were selected based on: (a) their potential to inform a dimensions and elements of Brennan and Resnick's (2012) CT assessment framework; and (b) understanding of associated learning outcomes.

The first and second parts of this interview were to help answer the sub-research question on attitudes towards programming. The third part intended to capture what extent students were able to describe their CT for the main research question, which included questions like:

- "Tell me what's going on here?"
- "Step me through how each line of this code block works"
- "What does this code block do exactly?"
- "Why did you decide to have this sound/pattern?"
- "What did you struggle with the most?"
- "What did you tinker with the most, why?"

Pre-unit of work interview with the participant music teacher. The pre-unit of work interview with the participant music teacher (Zach) took place after a one-hour session with me that intended to prepare him for the teaching phase of the unit of work. The information gathered from this interview intended to inform both research questions. The overall design of the questions were structured in four parts:

1. Background experience and training in music and programming to help understand his perspective in both subjects.
2. Questions similar to the pre-unit of work interviews with students were asked on the subscales (enjoyment, importance, and self-confidence) to help understand his attitude towards programming. However, they focused on teaching rather than learning programming and music composition (in the student interviews).
3. He was asked if he thought programming can be creative and if he thought of himself as a creative person to help understand his predisposition towards engaging with the creative activities in this unit of work.
4. Finally, he was asked about what predictions he would make about the potential successes and challenges this unit of work may have to help understand his predisposition towards the unit of work.

Post unit of work interview with the participant music teacher. The post-unit of work interview with Zach took place on the day of the final lesson in a non-teaching period he had. It was structured in four parts:

1. Attitude towards teaching programming were asked again on the three subscales.
2. Again, Zach was asked if he thought programming can be creative and if he thought of himself as creative.
3. His predictions were quoted back to him from the pre-unit of work interview about potential successes and challenges of the unit of work. He was asked whether he thought these predictions turned out to be true, and why.
4. Zach was asked about his confidence levels in teaching a unit on Sonic Pi and if he intends to plan a unit with Sonic Pi in the future.

The answers to the first three parts were compared with answers in the pre-unit of work interview. Results from these comparisons were intended to help inform answering both research questions from another teacher's perspective. The final question helped to inform the potential implications of this study in the participant music teacher's willingness to plan a unit on Sonic Pi.

3.2.3 End of class quizzes and reflections. All students took part in quizzes and reflections 20 minutes before the conclusion of Lessons 1-5 (see Appendix C). Lesson 6 instead was dedicated to the presentation of all projects where a reflection only (no quiz) on evaluating all project work and the unit of work itself took place. With a provided hyperlink, all quizzes and reflections were to be completed individually in class and submitted digitally.

Reflections. The purpose of student reflections was to provide qualitative snapshots of how students were using CT (for the main research question) and what aspects in each lesson they liked/disliked (for the sub-research question). Because of time restrictions, all reflection questions were designed with a short-answer format. Lessons 1-5 had three items for students to answer:

1. What did you struggle with the most today? If something didn't work, how did you get it working?
2. How do you think your piece could be further developed? How do you think you would go about this?
3. What did you like most about today? Was there anything you disliked or were surprised about?

The first two questions for Lessons 1-5 aimed to help inform the main research question so that qualitative snapshots of each lesson is provided on how students developed their projects. The third reflection question aimed to gather information on how students felt about each lesson, which intended to inform the sub-research question on student attitudes.

The final reflection in Lesson 6 had two questions to evaluate the unit of work as a whole. First it asked "What parts or aspects of your projects did you tinker with the most?" to help inform the main research question; second, it asked "What did you like/dislike most about the whole unit of work? What were you most surprised about?" to help inform the sub-research question.

Quizzes. The purpose of the quizzes was to help provide another indication that students could use CT to understand the skills and knowledge required for the learning outcomes set in each lesson. These quizzes emphasized the understanding of: (a) all elements in the CT concepts dimension (see Table 2); and (b) the CT practices element of testing and debugging from the Brennan and Resnick (2012) framework. The other CT dimensions and elements were considered not practical (e.g. being incremental and iterative) to test in a 10-minute quiz at the end of class for beginners.

The quizzes were designed to be 'open-book' where students could look up the internet or documentation and test answers with Sonic Pi before submitting them. This open-book design intended to emphasise the testing of CT (as opposed to memorisation of concepts) to help inform the main research question. However, they were completed individually under typical quiz conditions where communication (except with the teacher to help understand the question) and collaboration between students were both prohibited.

This unit of work was conducted in a general music class setting with students who already had lessons in music. Therefore, it was considered that more content was needed to be covered for programming. For this reason, the whole unit of work had a total of 50 quiz items (30 for programming and 20 for music). For each lesson, there were six items assigned to programming learning outcomes and four for music.

All quiz items were multiple-choice and had only one correct response. The multiple-choice design enabled students to be asked more questions in a shorter space of time, which reduced the amount of writing they needed to complete at the end of each lesson. They were simply marked with one mark for each question with a total of 10 marks per lesson (six for programming and four for music).

Programming quiz items. As mentioned, the programming question design for quizzes in Lessons 1 to 5 were designed to be aligned with the set learning outcomes in each lesson. For example, a programming learning outcome in Lesson 3 was "All students will understand how to use conditional logic (if/else) in Sonic Pi"; quiz questions for this lesson included the interpretation of code with if/else statements to predict what the resulting output may be (see Appendix C, Quiz 3). In summary, the six programming questions were in the following format and order for each quiz:

1. Understanding key Sonic Pi commands
2. Find the syntax error in a block of code
3. Selecting the right description of a code block
4. Debugging a code block (a) so it sounds as intended (according to a given description) with an audio reference
5. Debugging a code block (b) so it sounds as intended (according to a given description) with an audio reference
6. Sequencing the logical order of Sonic Pi code.

Music quiz items. Music question design for quizzes in Lessons 1-5 focused on the set music learning outcomes in each lesson. All questions had students listen to audio excerpts to identify different music composition and production concepts by ear, which is a core skill to develop for music students (MoE, n.d.b). The four music questions used the following format and order in each lesson:

1. Identifying different sounds (e.g. identifying the highest or lowest note)
2. Identifying compositional devices by ear (e.g. repetition or ostinatos)
3. Identifying the difference between two sounds (e.g. trumpet and synthesizer)
4. Questions on synthesiser or audio effect manipulation (e.g. attack and sustain).

3.2.4 Measurement tool. To measure the success of students' final music compositions, a Structure of Observed Learning Outcomes (SOLO) Taxonomy (Biggs & Collins, 1982) was used as an assessment framework to design a suitable marking rubric. The SOLO Taxonomy has shown to provide a simple and reliable model to assess understanding of learning outcomes in project-based work (Biggs & Collins, 1982; Chan, Tsui, Chan, & Hong, 2002) and is also recommended to assess CT for a programming context (CSER, 2017; Kallia 2017; Seiter, 2015). There are five levels of understanding in this model from minimal understanding to advanced levels:

1. *Pre-structural* grade is given for project work that does not address the project brief.
2. *Uni-structural* grade is given when few disconnected and limited aspects are provided of the project.
3. *Multi-structural* grade is given when there is evidence of several aspects of the project, but are disconnected as a whole.
4. *Relational* grade is given when their evidence of aspects of the project are coherent and linked, which collectively provide a deeper understanding of the brief as a whole.

5. *Extended abstract* grade is given when a new conceptual understanding of the project is evidenced. (Briggs & Collins, 1982)

The purpose of the marking rubric with the SOLO framework was to grade students' final music compositions (see Appendix B). The core learning outcomes set in Lessons 2 (making a scale), 3 (making a musical algorithms) and 4 (making a synthesiser) for music and programming were mapped onto the rubric and written based on the SOLO Taxonomy levels of understanding. Thus, students who received higher grades achieved a higher level of understanding of the set learning outcomes. For example, a programming learning outcome in Lesson 4 was “all students will create their own personalised synthesiser”, which is reflected in the marking rubric for each SOLO level as:

1. *Pre-structural:* “No attempt or synthesiser has bugs that prevent it from running”
2. *Uni-structural:* “A functioning synthesiser has been created but with little refinement and exploration”
3. *Multi-structural:* “A synthesiser has been created which manipulates a waveform in some way”
4. *Relational:* “A synthesiser has been created which demonstrates creative sculpting of waveforms with attack, release, sustain and decay”
5. *Extended Abstract:* “Obvious demonstration of creative exploration, refinement, and optimisation in final outcome” (see Appendix B).

Grading of projects. A final overall grade based on this rubric for both music and programming was given for each music composition (see Appendix I). This made a total of four grades for each student based on the two music compositions they completed. The final music compositions were graded holistically, meaning they could be strong in one criterion in the rubric to compensate for another that may be weak. However, the majority of the criteria were required be at the level of the grade awarded. Thus, these grades give some indication of the extent students were able to *integrate* the learning outcomes in Lessons 2, 3 and 4 for music and programming into a project.

3.3 Procedure

On the 14th of March 2018, ethical approval was granted from Educational Research Humans Ethics Committee ERHEC to commence the study. At this stage, a possible research site and participants had already been established from a professional contact I previously had. This professional contact was the participant music teacher in this school (Zach). Zach introduced the research to the school's principal and made the initial negotiations to gain approval to conduct this research (see Appendix G for all information sheets and permission forms). After a series of meetings with the school's principal and this music teacher, permission was officially granted from the principal (April 2018) to conduct the research. Formal permission was also given at this point from the participant music teacher to participate in the research. Thus, six lessons were then planned to be conducted on Wednesday and Friday mornings between 8:50am and 10:30am over three weeks in Term 2, 2018.

Next, all students and parents gave permission after their music teacher distributed information sheets and permission forms before the unit of work took place. On these permission forms, students and parents had the opportunity to raise issues or questions. No issues or questions were noted. All student participants were also invited to participate in a semi-structured pre and post unit of work interview. Ten students agreed to be selected that also had parental permission to be interviewed. Five students were selected from recommendations by Zach from this group that were likely to give rich descriptions of their experiences and for a gender balance (see Section 3.2.2 in this Chapter). Finally, Zach and the school informed the staff, participant students and their parents that this unit of work and research was to go ahead on the planned dates.

3.3.1 Participants. The sample of participants for this case study were volunteers in a compulsory Year 8 music class with 22 students (10 female and 12 male) and their music teacher. I led the teaching of the unit of work and the music teacher (pseudonym

Zach) primarily was there to support students where he could at all times during lessons. Zach was included as a participant because there was mutual interest for him to help inform the study from another teacher's perspective and as a professional development opportunity for him. The majority of students were from a Pākehā or New Zealand European descent (15/22). Other ethnicities included two Indian, two Chinese, one German, and two Samoan. As informed by Zach, no student had special needs and language issues. The pre-unit of work questionnaire revealed that:

- ten students reported prior programming experience before starting this unit of work in Scratch (five students) and Python (five students)
- eight students received (or had received) specialist music instrument training within a time range of one term to three years
- four students indicated that they had created their own music
- Sophia and Charlotte (pseudonyms) indicated they had prior experiences making music on computers.

Because no student had prior experience with the novel way of programming music with Sonic Pi, the unit of work was not modified or adapted to the students after learning assessing the prior experience of students in the pre unit of work survey.

Student interview participants. Five students were selected to take part in pre and post unit of work interviews (see Section 3.2.2 in this Chapter for the selection process). The selected students were given the pseudonyms: Sarah, Sam, Norman, Laura, and Ben. Table 1 provides information that was indicated from the pre-unit of work questionnaire prior to it starting.

Table 1

Background information gathered about the interviewed students

Student's pseudonym	Prior programming experience	Learnt a musical instrument	Had composed music before
Sam	Yes	No	No
Norman	Yes	Yes	Yes
Laura	Yes	Yes	Yes
Ben	Yes	Yes	No
Sarah	No	No	No

The participant music teacher (Zach). As informed in the pre questionnaire and interview, Zach had been teaching music for over 20 years in this school. His experience in teaching and playing music has focused on piano and singing performance for approximately 30 years. He reported that he had not composed or produced music before. Consequently, he did not focus on these skills in the music classes he teaches. Zach indicated he was positive about linking music composition and programming using Sonic Pi before the unit of work because he thought it was a "… really interesting way to compose and integrate the DT [Digital Technologies] curriculum with creative outcomes" (Zach, ti, 04/05/2018). He also said that it could be "very popular" (Zach, ti, 04/05/2018) for some students.

3.3.2 The unit of work procedure. The unit of work was conducted in three phases: (1) preparatory phase, (2) teaching phase, and (3) reflection phase. Brief descriptions of each phase is provided in the following paragraphs to illustrate how the designed unit of work unfolded. All participants were present for the majority of the study, and no major aberrations from the unit of work's original design needed to be made.

Preparatory phase. The first phase was conducted on the 4th of May 2018 in the assigned music classroom. First, I introduced myself and the unit of work to the class. Next, I explained each question in the pre-unit questionnaire they were about to complete. No questions were raised by any participant after giving them an opportunity to do so. Everyone individually completed the pre-unit questionnaire (including the participant music teacher) through a provided link on the computers stationed in the classroom. Pre-unit interviews were then conducted with the five selected students in their lunchtime that day. Next, one hour was spent with Zach afterschool to prepare him for the unit of work where we went over the practicalities and technical setup. Zach was told only to learn with and support students where he could. The pre-unit interview with Zach followed immediately after. All participants were present and all completed the measures for the first preparatory phase, which included the collection the following data:

- Student pre-unit questionnaire (n = 22)
- Five student semi-structured interviews (n = 5)
- Teacher (Zach) pre-unit questionnaire (n = 1)
- Teacher (Zach) semi-structured interview (n = 1).

Teaching phase. The second phase lasted from the 9th of May to the 25th of May 2018 with two scheduled lessons each week on Wednesday and Friday between 8:50am and 10:30am. No major technical issues or other interruptions were encountered throughout the whole unit of work. Two students were absent on Lesson 2 (James and Aiden), one on Lesson 3 (Sophia), and two on Lesson 4 (Aiden and Ava). The handling of the missing data for these students is explained under Section 3.4 in this Chapter. Absent students were able to catch-up missed lessons with the help from myself, Zach and other classmates.

Post-lesson reflections from myself were conducted for 30 minutes after each lesson, where observations were noted from the perspective of: (1) participant researcher;

(2) teacher of Music and Digital Technologies (DT); and (3) learning designer. These reflections observed any details about all participants relevant to answering the research questions in this study, for example: engagement levels, relevant discussions, and observed differences between students. These reflections contributed to answering the main research question and sub-question through providing qualitative perspectives of how the unit of work unfolded. All students that were present in each lesson completed the set work and had completed the set questionnaires, quizzes and reflections. This phase included collecting the following data:

- Post-lesson reflections (*n* = 22) *
- Student reflections (*n* = 22) *
- Student in-class quizzes (*n* = 22) *
- Student digital artefact folders consisting of all saved code (*n* = 22)

* Student absences meant that there were lower numbers for these measures in Lessons, 2 (*n* = 20), 3 (*n* = 21), and 4 (*n* = 20).

Reflection phase. The reflection phase was conducted on the day of the final lesson (25/05/2018), which involved questionnaires and interviews of all participants in the music classroom. All participants were present and completed the questionnaires and interviews for this phase, which included the collection of the following data:

- Student post-unit questionnaire (*n* = 22)
- Five student semi-structured interviews (*n* = 5)
- Teacher (Zach) post-unit questionnaire (*n* = 1)
- Teacher (Zach) semi-structured interview (*n* = 1)

As recommended by Cohen et al. (2011), member checks were also conducted with all interviewed participants. Once transcribed, interview transcripts were sent to each interviewee electronically by email. Participants had the option of requesting a meeting with me on an arranged date or sending through corrections or modifications within a week. No participant returned or requested any changes.

3.4 Analysis

A convergent parallel mixed-method analysis approach was employed in this project, where each measure was first analysed separately then merged together (Creswell, 2014). This systematic merging of data compares all results with a side-by-side comparison to see if they confirmed or disconfirmed each other. Once all data had been gathered, the process for analysis commenced in four stages for both research questions:

1. All raw data was prepared for analysis (transcribing interviews, marking quizzes, and grading projects).
2. An analysis process suited to each data type and research question was executed.
3. Data was systematically triangulated for similarities and differences (according to the convergent parallel mixed-method design for analysis).
4. The resulting findings were arranged according to their significance and selected to be presented in Chapter 4.

As stated by Cohen et al. (2011), these stages were non-linear and repetitive as regular checking between raw data and my interpretation was necessary to maintain a high degree of accuracy. After the first preparation of data phase was completed, the formal analysis of each measure for each research question began (stage 2).

3.4.1 Analysis of quantitative data. As recommended by Cohen et al. (2011), a quasi-experimental single-case design for the quantitative components was used for this project when studying a single-case its natural setting. This approach was chosen because this case study experiments with a new way of learning in a real teaching environment of a selected school (as opposed to a random sample in a controlled environment).

As mentioned, no missing data cases were discovered from participants who were present in lessons. However, two students were absent on Lesson 2 (James and Aiden), one on Lesson 3 (Sophia), and two on Lesson 4 (Aiden and Ava). According to McKnight, McKnight, Sidani and Figueredo (2007), numerical missing quiz data points can be averaged from a participant's previous and next data points. This strategy was employed for all student absences, which covered all missing quantitative data cases in this study (individual lesson quiz scores) as no student was absent for more than one lesson in a row. All missing qualitative data cases were left blank as recommended by Taylor et al. (2016).

After all numerical data was analysed and filled in for missing cases, it was imported into Microsoft Excel and SPSS for statistical analysis. Descriptive statistical calculations (mode, mean, median, minimum and maximum scores, range, variance, and standard deviation) were calculated according to each instrument's design. According to Cohen et al. (2011), correlations can also be made to understand interrelationships between numerical data. However, with only 22 student participants, quantitative analysis cannot make inferences and predictions. Where possible, correlations between different numerical measures were calculated to inform both research questions (described in the following sub-headings).

Main research question quantitative analysis. A one-way repeated-measures Analysis of Variance (ANOVA) was calculated for the quiz scores as recommended by Cohen et al. (2011). ANOVA tested if a null hypothesis indicated no change in participants' quiz scores in Lessons 1-5 and if there was statistical significance between different individual quiz scores.

Correlations between grades were also calculated between the music composition and programming grades. These calculations were made to see if the subject grades between music and programming were related. Correlations between quiz scores and all project grades were also calculated. These calculations were made to see if students who received high (or low) grades in their music compositions correlated with high (or low) quiz marks.

Sub-research question quantitative analysis. The sub-research question was measured with the quantitative data gathered in section 3 from the pre and post questionnaire items (18 total). Each questionnaire item in this section was assigned a numerical value from negative 3 to positive 3 for each option on the Likert scale. For each subscale, three questionnaire items were added together and divided by three to create an average score for each student. Thus, each student had a mean attitude score for each subscale (enjoyment, importance, and self-confidence) for both pre and post questionnaires in music and programming. These calculations created a total of 12 mean scores for each student in music and programming (see Appendix I for student results).

As recommended by Posten (1984), dependent samples *t*-tests are a robust way to test the null hypothesis that the pre and post questionnaires attitude means were equal. A dependent samples *t*-test was performed on each subscale (enjoyment, importance, and self-confidence) for programming and music. As recommended by Dowdy, Weardon and Chilko (2004), Cohen's *d* estimates were then calculated to compare if the effect sizes reinforced results from the dependent samples *t*-test.

Correlations for music and programming were calculated between overall attitude mean scores (between all subscales) and total quiz scores over the whole unit of work. Moreover, these means were correlated with individual project grades. The similarities and differences between these calculations were assessed to see if these attitude scores correlated with grades or quiz scores. According to Dowdy et al. (2004), a risk of multicollinearity can occur when there is a linear relationship among two or more independent variables with a variance inflation factor (VIF) above 5. Thus, multicollinearity were also calculated to see if correlations between subscales were affected.

3.4.2 Analysis of qualitative data. Hsieh and Shannon (2005) recommend a directed content analysis approach when qualitative data is categorised according to predetermined coding schemes. For both research questions, all qualitative data was analysed using a directed content analysis approach to align the findings with reliable frameworks for CT (main question) and attitude (sub-research question).

The same process for analysis was conducted for both research questions. First, the data was read multiple times and analysed for evidence on each predetermined code. Next, strings of text were identified and appropriately categorised under each predetermined code. As recommended by Hsieh and Shannon (2005), all qualitative data not able to be categorised with these codes were analysed to determine if any new codes emerged. However, no new categories were discovered for both research questions throughout this analysis process.

Once all raw qualitative data had been categorised, thematic analysis was used to identify and describe emerging patterns across the data under each code as themes. Thematic analysis allows researchers to find common themes for an interpretation of qualitative data to explain its significance (Cohen et al., 2011). In an iterative way, patterns across the categorised data were identified to generate themes across the data (Cohen et al., 2011). During this process, constant comparison with the raw data was employed so that the original context of each measure was not lost. The qualitative themes drawn using this approach were then compared with the quantitative data in accordance with the convergent parallel mixed-method design (Creswell, 2014).

Coding scheme for the main research question. The coding scheme for the main research question used the dimensions and their elements from the Brennan and Resnick (2012) CT assessment framework (see Table 2). This framework has been chosen because it was designed for young beginners with Scratch (a popular introductory programming language), which is similar to Sonic Pi with its low floor and high ceiling design (Arron, 2016). Moreover, this CT framework has been used to analyse CT in multiple studies at a school level by Lye and Koh (2014). Lye and Koh (2014) also recommend using a coding scheme to illuminate the different phases of the testing and debugging process, which was employed in the analysis for the testing and debugging element.

Table 2

Brennan and Resnick (2012) CT Assessment Framework

Computational concepts	Computational practice	Computational perspectives
• sequences • loops • conditionals • operators • data • parallelism	• incremental, iterative and adaptive • testing and debugging – with sub-coding scheme recommended by Lye and Koh (2014) • reusing and remixing • abstraction/modularisation	• expressing • connecting • reflecting

Brief descriptions of each CT dimension and their elements are provided as follows:

Computational concepts. The dimension of CT concepts by Brennan and Resnick (2012) highlighted *sequences*, *loops*, *events*, *parallelism*, *conditionals*, *operators*, and *data,* which are identified as core transferable programming concepts that can be applied to many programming languages. These CT concepts were also a focus in the analysis of all students' digital artefacts to help understand the extent and the complexity they were used.

Computational practices. Computational practices focus on *how* students think and learn computationally (Brennan & Resnick, 2012). There are four elements that make up this dimension:

1. Being incremental and iterative: This element is defined by Brennan and Resnick (2012) as an "adaptive process, one in which the plan might change in response to approaching a solution in small steps" (p. 7). In this unit of work, students were incremental through listening and reflecting on regular audio output in their project development. They also reflected in each lesson and gave feedback as a form of iteration.
2. Testing and debugging: Brennan and Resnick (2012) describe testing and debugging as "strategies for dealing with – and anticipating – problems" (p. 7).

Lye and Koh (2014) proposed using a modified framework which is based on general problem solving by Polya (1957) and further modified by Klahr and Carver (1988) as a coding scheme to help assess the testing and debugging element of the computational practice dimension. Lye and Koh's (2014, p. 58) four categories were employed as a coding scheme in my analysis, which consisted of: (1) understanding the problem, (2) devising a plan, (3) carrying out the plan, and (4) reviewing the plan.

3. Reusing and remixing: This element is defined by Brennan and Resnick (2012) as "building on other people's work" (p. 8). Students were encouraged to reuse and remix to creatively explore possibilities for their music compositions.
4. Abstracting and modularizing: Abstracting and modularising is defined by Brennan and Resnick (2012) as "building something large by putting together collections of smaller parts" (p. 9). Students regularly abstracted and modularized musical ideas through building up layers of sound and algorithms in the development of their music compositions.

Computational perspectives. Computational perspectives focus on the shifts in perspective observed in young people as they develop their computational thinking (Brennan & Resnick, 2012). Brennan and Resnick (2012) describe three elements that make up this dimension:

1. Expressing: the CT perspectives element of expressing is defined by Brennan and Resnick (2012) as "something they can use for design and self-expression" (p. 10). In this unit of work, students expressed themselves through the creative activity of developing music compositions with Sonic Pi.
2. Connecting: the CT perspectives element of connecting is defined as learning which is enriched through "interactions with others" (Brennan & Resnick, 2012, p. 10). In this unit of work, students regularly connect with each other through: the development of their group projects in pairs,

feedback sessions in each lesson, class and group discussions, and sharing of ideas.

3. Questioning: Brennan and Resnick (2012) define the CT perspectives element of questioning as how students “feel empowered to ask questions about and with technology” (p. 11). Students were regularly encouraged to ask questions through: class and group discussions, sharing of ideas, and group project work. Additionally, the teaching pedagogies employed in this study intended develop a student voice in students’ learning processes.

Coding scheme for the sub-research question. The sub-research question used the three attitude subscales employed by Teo (2007) as a coding scheme for the sub-research question (enjoyment, importance, and anxiety). These subscales are defined by Teo (2007) as follows: enjoyment means “enjoyment or liking” (p. 128); importance is “perceived usefulness” (p. 127); anxiety as “students' confidence” (p. 128). This framework to measure attitudes was employed in this study because: (a) there was a lack of reliable instruments to measure programming attitudes at a school level; (b) the attitudes subscales in the study by Teo (2007) were shown to be reliable to quantitatively measure them; and (c) the instrument could be easily adapted for a programming context at a school level. In this thesis, anxiety was modified to self-confidence because this focus was considered more appropriate to measure students’ attitudes towards programming.

3.4.3 Triangulation of data. According to Cohen et al. (2011), triangulation can be used to explain the complexity of human behaviour in a more descriptive and reliable way than what might result from a single research method or source. Multiple information sources and analyses can also be used to confirm or challenge the legitimacy of the findings (Creswell, 2014). Triangulation was embedded in the range of data collected, which was used to help both validate and describe answers to the research questions in a more comprehensive and robust manner.

In accordance with the convergent parallel mixed-method design employed in this study, triangulation was achieved through the systematic comparison of findings with all other relevant measures to inform, reinforce or challenge results. Additionally, methodological triangulation was embedded in the research design as recommended by Cohen et al. (2011) to help confirm and disconfirm data, which was utilised in a variety of ways, for example:

- end of lesson student reflections recorded the same questions on five different occasions (before the end of Lessons 1-5)
- five students were interviewed both before and after the unit of work to record a rich variety of responses based on the same interview protocols
- the music teacher pre and post interviews were triangulated with my post-lesson reflections to see what data were similar or different from another teacher's perspective
- all end of lesson quizzes were triangulated with the grades for students' final projects to strengthen findings for the main research question.

When the triangulation process resulted in comparisons that challenged each other, further analyses of other data types were undertaken to see if they could be resolved as recommended by Creswell (2014). However, when this further analyses yielded no resolve for differences, the results were recognised as limited and discussed in Chapter 5. The recognition of gaps and weaknesses is also particularly essential to recognise for robust and trustworthy case study research (Cohen et al., 2011). Gaps were identified through systematically comparing the intended purpose of each data type with the data collected to see if they met expectations.

3.4.4 Limitations. The data obtained in this single-case study is detail oriented and only provided a snapshot in time of the designed unit of work. Therefore, the findings cannot claim to be transferable, repeatable and dependable to other educational contexts (Cohen et al., 2011). A selective sampling strategy was also employed, where a prior professional contact was able to help the smooth inclusion of this study in their school. This sampling strategy is likely to have impacted results, thus, another researcher repeating this study could have different findings. However, the strength of this mixed-method case study is that it provides a rich description that increased understanding of the research questions (Creswell, 2014).

Additionally, I am a participant researcher and have a high influence over the data gathering process. According to Cohen et al. (2011), mitigating against this issue can include: post-lesson reflections of my different roles, reflexivity, respondent checks, and checks by external reviewers. These strategies have all been embedded in the methodology of this study as described in Section 3.5 in this Chapter.

A related limitation in answering the sub-question is the modelling effects I had as a male teacher, where reducing gender stereotypes for female participants had less impact. Being taught by a female teacher may promote more positive attitudes in female students towards computing subjects (Margolis & Fisher, 2002). Consequently, female interest and engagement was a factor I reflected on in my post-lesson reflections to help understand the sub-research question and illuminate potential differences between genders. However, findings on these gender differences only gave an indication of what occurred in this case study and cannot be generalisable due to the small amount of participants.

3.5 Trustworthiness – Validity and Reliability

Robust strategies to validate data reliability and its interpretation is necessary of trustworthy educational research (Creswell, 2014). On one hand, qualitative only research is often criticised due to the ambiguous nature of the data. On the other hand, quantitative only educational research is also criticised due to its limited ability to describe *why* results may have occurred when studying the complexities of human behaviour (Taylor et al., 2016). Thus, a mixed-method case study design has been employed where both qualitative and quantitative data can be triangulated to help strengthen the validity of findings (Cohen et al., 2011). A rich and thoughtful variety of data had been gathered appropriate for this mixed-method case study. Moreover, a range of appropriate methods for analysis to cross check findings made them more likely to be trustworthy. However, the designed unit of work will inevitably be taught and interpreted differently in other contexts. Therefore, this study is limited in its transferability and dependability because it only enables researchers and educators to understand what happened in this single-case study.

Triangulation. As mentioned in section 3.4.3, Triangulation has been accepted as a technique to validate data from two or more different types of sources to increase the likelihood that findings are reliable (Cohen et al., 2011). In accordance with the convergent parallel mixed-methodology employed in this study for both research questions, I analysed all measures individually, then systematically triangulated each with all other data (both qualitative and quantitative). This thorough method of triangulation helped to make my interpretation of the findings more reliable (see Section 3.4.3).

Validation of Instruments and Measurement Tool. All instruments and the measurement tool were first checked for acceptable face and content with my supervisors. As recommended by Cohen et al. (2011), they were also piloted with professional teacher contacts I had at several points of time before they were used for the full study. Feedback from piloting the instruments with these contacts helped to iron out inconsistencies and strengthened their validity to use with the participants in this research.

3.5.1 Reflections and involvement of others. Action research case studies have the researcher integrally involved (Cohen et al., 2011). To help recognise my own biases, Cohen et al. (2011) recommend that reflexivity, member checks and checks by external reviewers should be employed to ensure accurate analyses are drawn from the data.

Reflexivity. Reflexivity recognises that researchers are both participants and practitioners of their study, which impacts the interpretation of the results (Taylor et al., 2016). Reflexivity in this project was achieved through acknowledging and disclosing my involvement in the research (researcher and lead teacher) and how it may influence the study (see Chapter 1) as recommended by Herr and Anderson (2015). Each role I took on in this study included reflective post-lesson summaries after each lesson, which were: teacher of music, teacher of digital technologies, and participant researcher. I also kept a journal while developing the unit of work so the rationale behind design decisions are disclosed.

Member checks. The process of member checks involves asking participants to review the accuracy of findings drawn from data analysis (Taylor et al., 2016). In this study, all interviewees were given an opportunity to check their interview transcripts. Additionally, gathering interview and questionnaire data from the participant music teacher increased the reliability of my interpretations of the data from another teacher's perspective.

Checks by external reviewers. Checks at regular points through all stages of this research were from my supervisors at the University of Canterbury. These check-points helped to address issues to do with appropriate design, clarity, consistency, and interpretation at all stages in the research process.

3.6 Ethical Issues

As mentioned in section 3.3, I made sure I was following the University of Canterbury's Educational Research Human Ethics Committee's principles and guidelines for educational research with adolescents before this study commenced (official approval on the 14th of March 2018). To minimise any ethical risks, these principles ensured all participant rights were upheld and they gave informed and voluntary consent before participating. As a visiting teacher, I was in an authoritative position over the adolescent participants who are vulnerable to exploitation and coercion (Cohen et al., 2011). Therefore, I closely followed ethical principles at all stages of this research to protect student participants from potential harm.

Potential abuse of power relationships exist in the research process (Cohen et al., 2011), and efforts to prevent coercion of the school, parents, teachers, and students were regularly repeated and reviewed at every point. This was achieved through informing participants at every stage that they could withdraw from the study anytime. Moreover, they were provided full disclosure of: the scope, purpose, and the extent of the data gathered from them, and how this data was used. I was also available through email to answer any questions and listen to issues or concerns from any participant at any stage during the study (including student participants' parents).

Informed and voluntary consent requires all participants agree to participation with their individual free will and not pressured or coerced into giving this consent (Cohen et al., 2011). Signed consent forms from participants were obtained from the school, participant music teacher, participant students and their parents. Students were given information on the purpose of this study that was written appropriately for their age (Year 8). I disclosed information on the study in their regular music class with the participant music teacher present. In this time, students had a chance to ask questions and raise any concerns they might have had about the study. They were also given the opportunity to raise these questions or concerns after class through their parents or another adult. No concerns or questions were raised from students at this point.

Parental and student participant consent. Further permission was sought from the parents and students of the selected music class before the study took place. Before the unit of work started, Zach distributed the information sheets and permission forms to students and parents. All student participants and their parents gave permission to participate in the study. On these permission forms, students and parents had the opportunity to raise issues or questions about the study. No issues or questions were raised by the students and their parents.

Student interviews. All students were invited to participate in semi-structured pre and post teaching audio recorded interviews. Parents had the option to give permission for their son or daughter to participate in the study but not for interviews on all permission forms. Additionally, students could also agree or disagree to be interviewed. Only students who both themselves and their parents had agreed (through signing the permission forms), were considered for an interview. A total of ten students and their parents had given this permission to be interviewed.

Confidentiality. All data and information obtained from all participants remained confidential throughout the research process. Pseudonyms were used for all participants and the participating school so their privacy and confidentiality can be adhered to. I am the only person with access to the matching list of real names to pseudonyms. I collected data only relevant to the study based on the information sheets participants were given prior to the teaching phase starting. All data was kept digitally and secure with a password coded cloud storage system and laptop computer. The University of Canterbury Educational Research Human Ethics Committee requires data is kept for five years and then destroyed, which will be followed.

Chapter 4: Findings

This chapter aims to present the results aligned to the research questions in this study, which are:

Main research question: How can Computational Thinking (CT) support Year 8 students' learning outcomes in Programming and Music with the Sonic Pi platform?

Sub-research question: To what extent can the creative activity of composing music in the Sonic Pi platform help to promote positive attitudes towards programming?

Findings for the main research question are presented first, followed by the sub-research question. As recommended by Creswell (2014), the relatively robust quantitative results are presented before qualitative data according to the directed content analysis approach. Main sub-headings for each research question are divided according to the theoretical coding frameworks employed (outlined in Chapter 3 section 3.5.2). A summary of findings also concludes each research question.

4.1 Main Research Question

The main research question findings are presented through the three dimensions and their elements from the Brennan and Resnick's (2012) CT assessment framework under the following sub-headings:

4.1.1 CT concepts

- Quantitative findings - project grades and quiz scores
- Sequences
- Loops and parallelism
- Data
- Conditions and Operators

4.1.2 CT practices

- Being incremental and iterative
- Testing and Debugging
- Reuse and remixing
- Abstracting and modularising

4.1.3 CT perspectives:

- Quantitative findings
- Expressing
- Connecting
- Questioning

4.1.4 Summary of findings

4.1.1 CT concepts. The dimension of CT concepts identify *sequences*, *loops*, *events*, *parallelism*, *conditionals*, *operators*, and *data* as key transferable concepts applying to many programming languages (Brennan & Resnick, 2012). This section focuses on how the data coded and analysed under each element of the CT concepts dimension supported the set learning outcomes for programming and music composition. Project grades and end of class quizzes in both music and programming were the primary quantitative measures for the main research question, which are presented first.

Project grades. Each student handed in two music compositions (individual and groups in pairs), which were both graded by the researcher against a designed rubric (see Appendix B). These grades assessed the holistic integration of all CT concepts from music and programming perspectives (see Appendix I for individual student results). Each music composition (group and individual) was evaluated for both programming and music, making a total of four grades for each student. Grades from the Structure of Observed Learning Outcomes (SOLO) Taxonomy were assigned a numerical value to enable statistical analysis (see Table 3).

Table 3

SOLO Taxonomy grades assigned values to enable statistical analysis

SOLO Taxonomy Grade	Value assigned for statistical analysis
Extended abstract	5
Relational	4
Multi-structural	3
Uni-structural	2
Pre-structural	1

Table 4 shows all mean results for students' project grades are comparatively close (between $M = 3.45$ and $M = 3.59$). The standard deviations are also close (between $SD = .91$ and $SD = 1.18$). Most students achieved at least a multi-structural grade (3/5) for both projects in music and programming. Moreover, Table 5 shows no student received the lowest pre-structural grade for both projects. Because the rubric was designed to be reflective of the set learning outcomes (see Appendix B), these results collectively suggest CT concepts supported many learning outcomes in music and programming for most students in this study.

There are strong correlations between the group and individual project for programming ($r = .71$) and music ($r = .88$). These correlations suggest there is little difference in how students used CT concepts when collaborating in comparison to their individual work.

Table 4

Mean and standard deviation of final grades for individual and group projects out of five (see Table 3 for scaling)

	Individual project programming	Individual project music	Group project programming	Group project music
M	3.55	3.59	3.55	3.45
SD	1.06	1.10	.91	1.18

Table 5

Student counts for music and programming grades in their individual and group projects (see Appendix I)

Category of the rubric	Individual project grade for programming	Individual project grade for music	Group project grade for programming	Group project grade for music
Extended Abstract	5	6	4	6
Relational	6	5	6	4
Multi-structural	7	7	10	6
Uni-structural	4	4	2	6
Pre-structural	0	0	0	0

Quiz scores. The end of class quizzes from Lessons 1 to 5 in programming (six items per lesson) and music (four items per lesson) involved the use of CT concepts based on the set learning outcomes. Total mean results for all lessons indicate most students answered at least half the items correctly in programming (*M* = 18.27/30, *SD* = 5.57) and music (*M* = 15.5/20, *SD* = 3.20). These results reinforce the project grades that many set learning outcomes were understood by most students in both subjects.

Figure 1 and Figure 2 indicate the total quiz scores for programming and music in box and whisker charts. The interquartile range and upper whisker are above 50% for both subjects. However, the lower whisker is below 50% for programming (Figure 1) and the majority above 50% for music (Figure 2), which suggests less competent students found the programming quiz items more difficult. This difference in the lower whisker for both subjects is likely due to students' prior experience with music (as the unit of work was conducted with music students). However, there were two additional programming items per quiz, which increases the likelihood of incorrect answers.

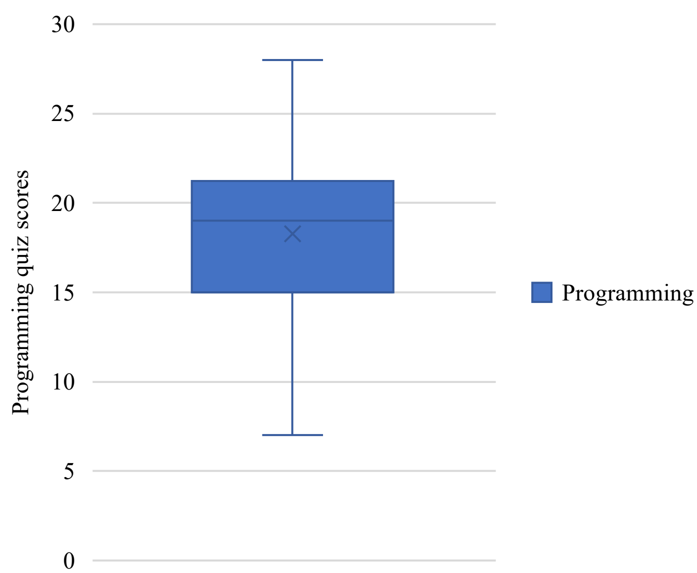

Figure 1. Total programming quiz scores (maximum 30).

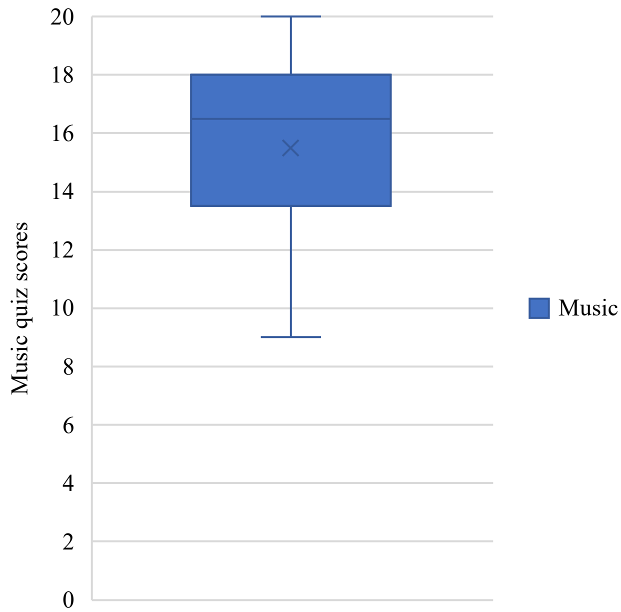

Figure 2. Total music quiz scores (maximum 20).

Differences between individual quizzes. A repeated-measures Analysis of Variance (ANOVA) rejects the null hypothesis of no significant changes in scores between lessons for programming ($F(4, 84) = 7.999$, $p < 0.001$) and music ($F(4, 84) = 8.147$, $p < .001$). Post-hoc tests using Tukey's honestly significant difference (HSD) reveal programming quiz results are statistically lower for Lesson 3 ($p = .018$ with a mean difference of $M = 0.682$) and Lesson 4 ($p = .001$ with a mean difference of $M = 1.182$) than Lesson 1. When compared to Lesson 1, these findings suggest many students struggled to understand the CT concepts in Lesson 3 (where the CT concepts of conditions and operators were introduced) and Lesson 4 (where the in-class quiz tested students on audio effects and making synthesisers in Sonic Pi) (see Appendix C for all quizzes). Figure 3 illustrates the range of scores across all lessons, indicating Lessons 3 and 4 had fewer students answering correct items within the interquartile range. Notably, Lessons 1, 2 and 5 are similar across all quartiles and whiskers in Figure 3.

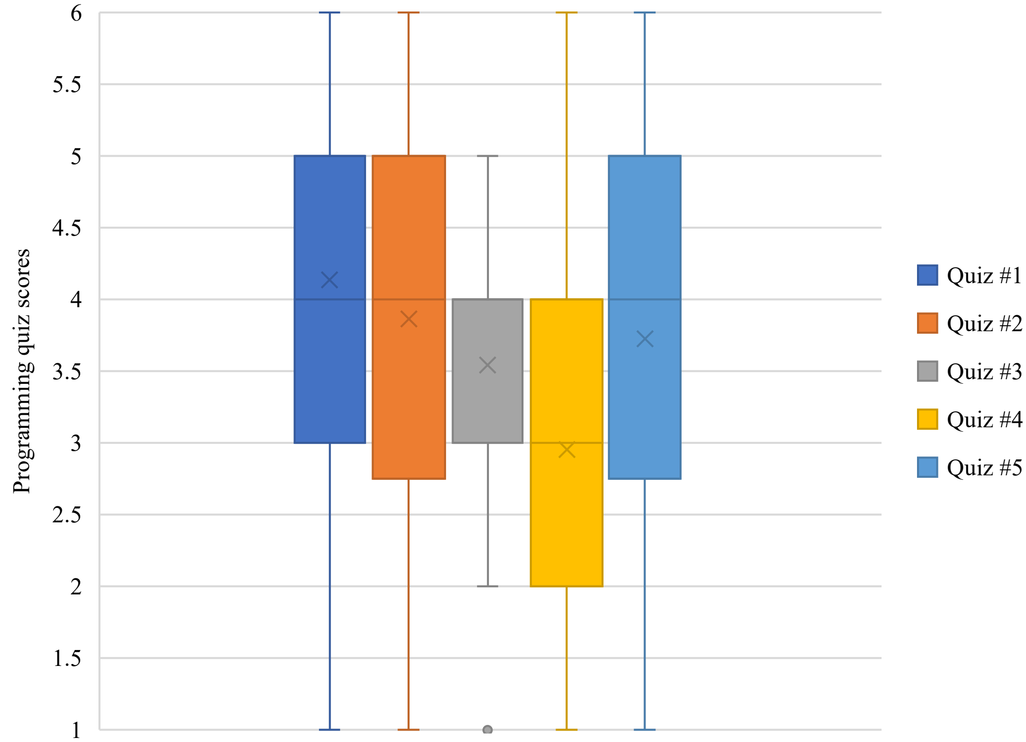

Figure 3. Programming quiz scores for each lesson (maximum 6).

For the music quiz items, post-hoc tests using Tukey's HSD reveal results are statistically significantly lower for Lesson 3 (p = .008 with a mean difference of M = 0.50) and Lesson 4 (p = .002 with a mean difference of M = .682) than for Lesson 2 (excluding Lesson 1). These significant differences indicate the quiz scores in Lesson 3 and 4 for music are overall statistically significantly lower than Lesson 2. Figure 4 presents the range of scores for music in all lessons, which visually confirms Lessons 3 and 4 have fewer correct marks for their interquartile results compared to all other lessons. Figure 4 also illustrates Lesson 1 and 2 were almost identical with all interquartile results above 50%. Like Figure 3 for the programming quiz scores, the interquartile results for music quiz scores drop in Lessons 3 and 4 when compared to all other lessons. The Lesson 3 and 4 music quiz items aurally tested students on identifying specified instruments, sounds, textures, and pitches.

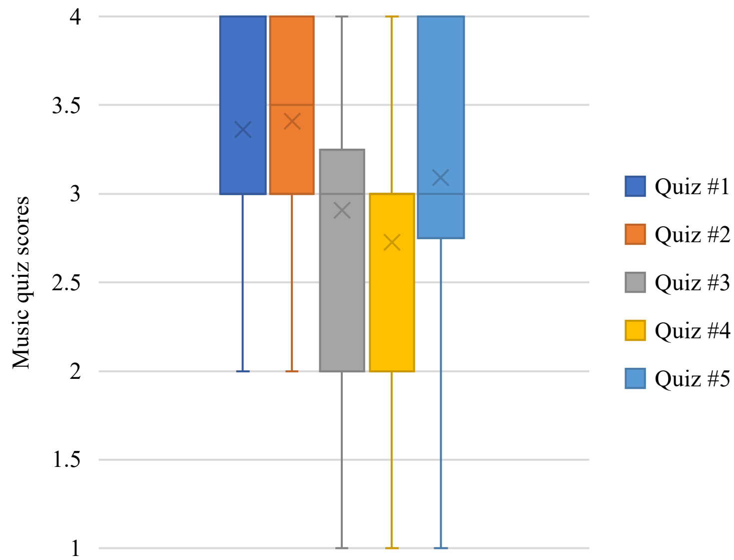

Figure 4. Music quiz scores for each lesson (maximum 4).

Thus, the music quiz items in Lesson 3 (on algorithmic music) and 4 (on making a synthesizer) are statistically significantly lower in both music and programming than previous lessons. These lower scores suggest the learning outcomes and the associated CT concepts in these lessons were more difficult for most students than Lessons 1 and 2. However, mean results still reveal most students were answering close to 50% correct for both programming (Lesson 3 with M = 3.55 and Lesson 4 with M = 2.95) and music (Lesson 3 M = 2.91 and Lesson 4 M = 2.73).

Correlations between quiz scores and final project grades. Strong correlations were found between quiz scores and final grades for students' individual projects in both programming (r = .92) and music (r = .78). Moreover, there are strong correlations between quiz results and their group project for programming (r = .67) and music (r = .82). These findings collectively suggest grades for both final projects (group and individual) and quiz scores for music and programming are likely to be similar for each student. For example, students with low quiz scores for programming likely have low programming grades (also likely for average and high performing students). However, these correlations are sensitive to the small sample size in this study (n = 22).

Sequences. All data indicated an understanding of sequences for all students throughout the unit of work. However, the depth and nuance students used to integrate sequences to a collective musical composition varied according to their level of competence. This variation in sequence use is reflected in students' overall grades; for example, lower grades generally indicated less evidence of musical refinement and blending with other sequences than those with higher grades in their music compositions.

A simplistic use of sequences is illustrated in Appendix F with Lucas's code. For example, on lines 23-27 (see Figure 5), there are three play commands with no manipulation.

```
22  loop do
23    play 22.444
24    sleep 2
25    play 53
26
27    play 55
28  end
```

Figure 5. Excerpt of Lucas's final individual project code.

This lack of refinement contributed to Lucas receiving a pre-structural grade (2/5) in both music and programming for this project (see Appendix F for Lucas's complete final project code). Lucas's total quiz scores confirm he did not understand many CT concepts with only 7/30 programming quiz items correctly answered. In contrast, Emma's exemplar from lines 1-7 (see Figure 6) selects a synthesiser more carefully and manipulates a sequence of notes with reverb effect and many different parameters.

```
1  live_loop :rad do
2    use_synth :prophet
3    with_fx :reverb, mix: 0.3 do
4      play choose([:C3,:D3, :A3, :G5]), attack: 5, release: 5, sustain: rrand(0.1, 3), decay: rrand(0.1, 3)
5      sleep 2
6    end
7  end
```

Figure 6. Excerpt of Emma's final individual project code.

Emma's sequence contributed to an extended abstract grade been given in music (4/5) and a relational grade in programming (5/5) (see Appendix F for Emma's complete final project code). Her total quiz scores also reinforce she was able to use the CT concept of sequences to answer programming quiz items with 19/30 correctly answered. Figure 6 shows a more nuanced and complex sequence, which demonstrates greater awareness on how sounds blend with her overall music composition in comparison to Lucas's example (Figure 5). Thus, evidence of refinement and sophistication in the use of sequences contributes significantly to the final grades given.

Post interviews indicate how a few students were able to verbally describe sequences within a looping code block. For example, Ben was asked to reflect on the section of code (see Figure 7) in his post-teaching interview (Ben, si, 25/05/2018):

> CP: Tell me about what is going on in this code block on lines 8 to 14 in your group project. Step through each line of code for me.
>
> Ben: Ok, I think we were going for some sort of bass line because Norman said he thought we should have bass in our project. But I wanted it to not repeat so much so we wrote an array on line 10 that had different numbers. Then on line 13 we use that for the sleep time so the time changes.
>
> CP: What about line 12? What is going on there?
>
> Ben: Um, basically it plays whatever note is in the random variable. That is actually made random on line 11 with a different scale. Then we use that random array [points to line 10] to control release and another random thing to control the cutoff. Then on line 13 we used a random sleep for each time it plays.

This interview excerpt demonstrates that Ben is able to accurately describe his sequence within the looping code block in Figure 7. The musical effect of the code in Figure 7 is it algorithmically generates melodies based on an E minor pentatonic scale with random variations in notes, release, cutoff and sleep times contained within a loop. Ben's total quiz scores also reinforce he was able to use the CT concept of sequences to answer programming quiz items with 28/30 correctly answered. Other post-teaching interviews reveal more examples of students describing sequences to a similar degree of accuracy and descriptiveness. Thus, evidence across data indicates all students used the CT concept of sequences to support learning outcomes in music and programming to varying degrees of complexity and refinement.

```
live_loop :bass do
  use_synth :beep
  random = [0.125, 0.25, 0.5].choose
  notes = scale(:e1, :minor_pentatonic).choose
  play notes, release: random * 1.125, cutoff: rrand(75, 110)
  sleep random
end
```

Figure 7. Excerpt of Ben and Norman's final group project code.

Misconception with melody writing in Sonic Pi. A notable misconception unique to Sonic Pi occurred when introducing sequences in Lesson 1. Students unanticipated a sleep command is needed to create a time delay between notes to sequence a melody (and not have notes sound simultaneously). This unique characteristic in Sonic Pi highlighted the essential role of time (or the spaces) between notes to create rhythm, which supported learning outcomes in music.

Loops and parallelism. Loops and parallelism were introduced in Lesson 1 and 2. Upon counting all loop code blocks in final project code for all students, each project had an average of four loops. All music compositions generally have a different sound or sequence within their loops, indicating an average of four sounds or sequences that feature in each music composition.

Many students were initially confused that a sleep command is required in a loop when using Sonic Pi (rr, 09/05/2018; rr, 11/05/2018). If no sleep command is coded in a Sonic Pi sequence, students receive the error, "Thread death! Loop did not sleep or sync!". The sleep command is needed because a loop would repeat each sound almost simultaneously otherwise, which Sonic Pi stops before it creates a cacophony of sound. Because the resulting error message gives the user the problem's cause with "Loop did not sleep", the majority of students had little trouble solving these errors. This unique characteristic highlights a notable difference Sonic Pi has in comparison to loops in other programming languages, which helped to support learning outcomes relating to parallelism in programming and the importance of time within musical repetition.

Many students were heard using the terms “loop” or “repeat” interchangeably throughout the unit of work (rr, 11/05/2018; rr, 25/05/2018). For example, Sarah commented on Kate's project after listening “I really liked how your live loops were slightly out of time with each other” (rr, 25/05/2018). Zach (the participant music teacher) reinforced this observation in his post-unit of work questionnaire, where he said “students were communicating through Sonic Pi syntax and music concepts” (post tq, 25/05/2018). Five examples of student reflections in various lessons were also found using the word “loop” and “repeat” to describe what they planned to work on in following lessons. For example, “our song repeats too much” (Charlotte, sr, 16/05/2018) and “we need to take out some loops” (Liam, sr, 18/05/2018). This wide use of the term ‘loop’ indicates that students were able to communicate and reflect with the CT concept of loops to support the refinement of their final music compositions.

Particularly noteworthy, was a planned group listening exercise with Hans Zimmer's music composition called *Time* (Zimmer, 2010) from the film *Inception* in Lesson 2. In this piece, almost all instruments loop with repeating sequences (or ostinatos in music). All groups were observed to correctly identify and write down a few loops they heard (rr, 11/05/2018). A few students were able to describe the nature of a loop within the piece accurately in the class discussion after listening. For example, William said: “the guitar is really simple, it is just looping two notes over and over” (rr, 11/09/2018). No student contributed incorrect answers in this class discussion, which may suggest only those that felt confident contributing did so. This evidence indicates the activity supported the music learning outcome “*All* students will *recognise basic composition repetition/variation in ‘Time’* by *Hans Zimmer*” to understand the CT concepts of loops and parallelism.

No significant differences were found between all students in the way they used loops. This finding is most evident in students’ saved code where all code blocks almost exclusively use the *live* loop. Live loops in Sonic Pi enable users to layer sounds and sequences, which also helped to illuminate the CT concept of parallelism in programming and layering sounds in music. All students used the CT concept of loops and parallelism to layer sounds in their music compositions, which supported the set learning outcomes in Lessons 1 and 2 in music and programming (see Appendix A).

However, two other forms of Sonic Pi syntax involving loops exist: (1) a user can specify the number of times a code block loops (e.g. "4.times do" loops a code block four times) and (2) another simple loop command (e.g. "loop do" repeats a code block forever). These other loops were only found twice from Laura, Oliver and Ava in their final projects. Additionally, there were only a few examples of nested loops (loops inside of loops) in all final project code. These findings suggest the use of looping in Sonic Pi presents less rigour when compared to using a *for* or *while* loop in other text-based programming languages like Python (where students need to consider the condition(s) upon which the loop is entered and exited more carefully).

Data. The CT concept of using data was introduced as a task in Lesson 2, where students were asked to store a collection of notes (as data) in a variable. This collection of notes were then programmed to be played in a manner of each student's preference. The intended musical outcome was to create a harmonic or melodic palate that reflected a personalised musical feeling (suiting their chosen video according to the project brief in Appendix B). In this way, students' use of lists and variables were able to support learning outcomes in music and programming in Lesson 2.

Figure 8 is an example of Sam using data in his final individual project, which uses two separate lists to store two collections of notes on lines 17 and 18 (called "myscale" and "myscale1"). With a different sleep time on lines 22 (0.2) and 27 (0.3), the code blocks from lines 20-23 and 25-28 are slightly out of time with each other. This small .1 second of time difference means the two loops slowly phase in and out with each other through the music notes in "myscale" (line 17) and "myscale1" (line 18), which results in musical harmonic movement between the two lists of notes. Additionally, these two code blocks make a texture of sound through the layering of different synthesisers (“square” on line 21 and “mod_beep” on line 26).

```
17 myscale = [:A7, :C7, :B7, :C3, :A3]
18 myscale1 = [:G3, :D3, :B3]
19
20 live_loop :synth do
21   synth :square, note: myscale.choose, cutoff: rrand(0.2, 40), amp: rrand(0.01, 0.3), pan: rrand(-1, 1)
22   sleep 0.2
23 end
24
25 live_loop :synth do
26   synth :mod_beep, note: myscale1.choose, cutoff: rrand(0.2, 40), amp: rrand(0.01, 0.3), pan: rrand(-1, 1)
27   sleep 0.3
28 end
```

Figure 8. Excerpt of Sam’s final individual project code.

Sam was asked to reflect on this section of code in Figure 8 in his post-unit of work interview:

> CP: Tell me about this code block from lines 17 to 28.
>
> Sam: Um, basically lines 17 and 18 have two different scales I made. I wanted to use them in a loop that went in different times with each other.
>
> CP: Can you tell me about the loops themselves? Are they similar or different?
>
> Sam: I think basically they are the same. But actually, I have different sounds for each of them, and like I said, they were out of time with each other, but basically, they're the same. (Sam, si, 25/05/2018)

Sam’s reflection on this code block in Figure 8 is accurate and indicates the musical effect is intentional. This level of nuanced experimentation contributed to Sam receiving the highest grade of *extended abstract* (5/5) in programming and music.

However, the development of lists and variables varied in-depth and complexity for other students. For example, Figure 9 shows William’s use of lists on line 17 that stores number values in the variable “n”, which is called on line 20 to manipulate sleep on each iteration of the loop. This example demonstrates a high degree of experimentation in William’s use of lists, which contributed to a *relational* grade (4/5) for programming.

```
live_loop :link do
  n = [0.25, 0.022, 0.2].choose
  3.times do
    sample :ambi_piano, sustain: 0, release: 1
    sleep n
  end
end
```

Figure 9. Excerpt from William's final individual project code.

In Figure 10, Charlotte stores a number in a random number generator (between .5 and 1) on line 13, which is used for a sleep time on line 15. Similar to William's example in Figure 9, this example demonstrates a high degree of experimentation that contributed to a *relational* (4/5) grade for Charlotte's overall project in music and programming.

```
12 live_loop :amp do
13   r = rrand(0.5, 1)
14   sample :ambi_glass_hum, attack: 5, sustain: 2, decay: 1, rate: rrand(0.2, 0.4)
15   sleep r
16 end
```

Figure 10. Excerpt from Charlotte's final individual project code.

In contrast, Figure 11 is Daniel's example of using a list, which has less complexity in the use of data compared to the examples in Figures 8, 9, and 10. A list called "notes" is created on line 11, which is executed on line 14 in a loop that iterates with the use of Sonic Pi's ".tick" method. The code in Figure 11 indicates a lack of musical variation and evidence of refinement in comparison with Sam's (see Figure 9) and Charlotte's (see Figure 10) examples, which contributed to Daniel receiving a *uni-structural* (2/5) grade in both music and programming.

```
11 notes = [:A7, :C7, :B7]
12
13 live_loop :noy do
14   play notes.tick
15   sleep 1
16 end
```

Figure 11. Excerpt from Daniel's final individual project code.

Students were mixed in the descriptiveness of their variable names. Of the 28 variables used across all final project code, more than half (16) were evaluated as inappropriately named. For example, Emma's variable (see in Appendix F on line 13) is named “yep” to store notes in a scale; it would be more appropriate to name this variable “sadScale” or another descriptive name. The analysis of students’ code also reveals that 20 variables are only called upon once, and the remaining eight instances were found to be called upon twice. Because variables are minimally used in students’ code, the need to give them descriptive names is less obvious to novice programmers (as any changes made result in fewer potential errors across their whole project). This finding is similar to students’ naming of functions, which is presented under the sub-heading Abstracting and Modularising in this Chapter (see Section 4.1.2). These findings highlight an area where students were not effectively developing conventional programming habits, which may create a barrier for future growth in programming.

Other forms of data were also used by students that came pre-installed in Sonic Pi, such as: audio samples, synthesiser sounds, and pre-programmed scales and chords. Analysis of student code reveals all students had evidence for the use and manipulation of these data forms. For example, Figure 12 from Liam’s project shows the use of chord one and five from the A minor tonality on line 20. Figure 9 on line 19 and Figure 8 on line 14 (above) are examples of how audio samples were used often in students’ code. Similar to the findings on storing notes and numbers in variables in Figures 8, 9, 10, and 11, those who received higher grades used more sensitive manipulation and refinement of this pre-installed data in the musical development of their projects than students with lower grades.

```
19  live_loop :chords do
20    c = (chord_degree, [:i, :v].ring.tick, :A3, :minor, 4)
21    play chord_invert( c, [-1, 0, 1].choose), amp: 0.5, release: 1, attack: rrand(0.1, 2)
22    sleep rrand(0.1, 3)
23  end
```

Figure 12. Excerpt from Liam’s final individual project code.

Conditions and operators. After a class demonstration in Lesson 3, students were observed to successfully implement conditions and operators in Sonic Pi (rr, 16/05/2018). However, the quiz results from testing conditions and operators in Lesson 3 countered this observation because results were statistically significantly lower than previous lessons in programming (p = .018 with a mean difference of M = 0.682). This contradiction indicates conditions and operators were both more difficult to understand than was first observed and not well understood by many students.

Only three students (Ben, Laura, and Emma) used conditions and operators in their final projects. For example, Figure 13 shows a code block from Laura's final individual project that uses a condition and a less than operator (<). She has music notes selected from an "egyptian" scale within the first branch of her condition (lines 33 and 34), and the "else" branch playing inversions of chords one and five within an E minor tonality (lines 37 and 38).

```
26 live_loop :chords do
27   #use_random_seed 2435
28   sample :guit_em9
29   nlen = [1,2,3].choose
30   cho = (chord_degree, [:i, :v].ring.tick, :e3, :minor, 4)
31   if rand() > 0.7 then
32     3.times do
33       play scale([:e6, :e6].choose, :egyptian).choose, amp: 0.2, release: nlen / 2.0
34       sleep nlen / 4.0
35     end
36   else
37     play chord_invert( cho, [-1, 0, 1].choose), amp: 0.5, release: nlen
38     sleep nlen
39   end
40 end
```

Figure 13. Excerpt of Laura's final individual project code.

Laura was asked to describe this code in her post-unit of work interview:

CP: Can you please describe for me what is going on in lines 26 to 40 here?

Laura: Umm, basically it plays chords and a melody kind of randomly.

CP: Can you tell me what lines play chords and what lines play the melody?

Laura: Yep, so lines 31 to 35 plays the melody and then lines 36 to 39 plays chords.

CP: Can you tell me what condition needs to be true in order for the melody to be played?

Laura: Umm, so I think a random number on line 31 needs to be over 0.7. Then that plays. Otherwise the chords get played. (Laura, si, 25/05/2018)

This interview excerpt demonstrates Laura was able to accurately describe the logic of the condition and operator she used in her final project. The musical effect of the code block in Figure 13 is it algorithmically generates an almost infinite amount of variation within the set bounds of the "egyptian" scale and minor chords (one and five in E minor). This example demonstrates the CT concepts of conditions and operators are able to support learning outcomes in music and programming through the Sonic Pi platform. However, Laura was likely to be already familiar with these CT concepts through prior programming experience in Python before the unit of work took place.

The post-unit of work interviews with Ben and Emma did not cover the conditions and operators they used in their final projects, but they integrated these concepts with a similar complexity to Laura's example shown in Figure 13. Ben and Emma also reported prior programming experience, which indicates that Ben, Laura, and Emma were likely transferring knowledge from their experiences in Scratch and Python (T1, 09/05/19). These three students' quiz results for testing conditions and operators in Lesson 3 confirm that they mostly understood how to solve bugs involving these CT concepts (Ben 5/6; Laura, 5/6; Emma 6/6). Nevertheless, these results collectively suggest the unit of work is significantly limited for most students in the way the CT concepts of conditions and operators can support learning outcomes with the designed unit of work and project brief (see Appendices A and B).

4.1.2 CT practices. The computational practices dimension focus on *how* students think and learn computationally (Brennan & Resnick, 2012), which has the following four elements: (1) being *incremental and iterative*, (2) *testing and debugging*, (3) *reusing and remixing*, (4) and *abstracting and modularising*. This section focuses on how the data coded and analysed under each element of the CT practices dimension supported the set learning outcomes for programming and music.

Being incremental and iterative. Brennan and Resnick (2012) define being incremental and iterative as an "adaptive process, one in which the plan might change in response to approaching a solution in small steps" (p. 7). Students showed evidence for this CT practice through saving their code under a new file name before the end of each lesson. Most music compositions had three or four saved files, which helped to provide snapshots of students being incremental and iterative.

Comparing similar code blocks side-by-side revealed instances of incremental and iterative development between two or more lessons in all students' music compositions. For example, Figure 14 presents an excerpt from Ben and Norman's group project highlighting incremental and iterative development over three lessons. Analysis of Figure 14 exposes three major changes made over this time:

- In Lesson 4, new code from lines 43-47 was copied exactly from a code block on lines 49-53 (with minor differences in the release and sleep parameters)
- In Lesson 5, the amplitude and panning parameters were added on lines 45 and 50
- Also in Lesson 5: the "notes" variable changed its stored values, and the naming of both live loops were updated to "hocket1" (line 43) and "hocket2" (line 48).

Similar refinements in other parts of this project contributed to extended abstract grades (5/5) given in both music and programming for this music composition.

Saved code end of lesson #2

```
notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4)

live_loop :myscale do
  use_synth :hollow
  play notes.tick, release: 0.8
  sleep 0.2
end
```

Saved code end of lesson #4

```
notes = (ring :D5, :Fs4,
         :E4, :Cs5, :B4, :Fs4)

live_loop :scale do
  use_synth :hollow
  play notes.tick, release: 0.3
  sleep 0.5
end

live_loop :scale1 do
  use_synth :hollow
  play notes.tick, release: 0.6
  sleep 0.45
end
```

Saved code end of lesson #5 (final)

```
notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4,
         :E4, :Cs5, :B4, :Fs4, :D5, :Cs5)

live_loop :hocket1 do
  use_synth :hollow
  play notes.tick, release: 0.5, amp: 0.2, pan: 0.8
  sleep 0.33
end
live_loop :hocket2 do
  use_synth :hollow
  play notes.tick, release: 0.7, amp: 0.1, pan: -0.75
  sleep 0.294
end
```

Figure 14. Excerpts from Ben and Norman's group project code.

Ben was questioned about this code in his post-unit of work interview:

> CP: Can you describe for me your revisions on this code block over the lessons you were making it?
>
> Ben: Um, yeah, so it started with just playing the notes we wanted then it kind of changed to two sounds going out of time with each other.
>
> CP: Can you remember what triggered this idea? It seems like you came up with it in Lesson 4.
>
> Ben: I think maybe it was just because we were playing around and then we realised we can make it sound like one loop was going faster than the other.
>
> CP: Can you describe what triggered that realisation?
>
> Ben: I think we just wanted something more cool than notes just repeating in a scale so we were trying ideas.
>
> CP: Can you remember how many ideas you tried and why did you picked this one?

> Ben: I think maybe we went through about 3 ideas. I think we picked this one because we think it fitted best with the other sounds, like we really liked the sound we were using but we thought it was boring by itself here [points to code block written in Lesson 2 in Figure 14]. (Ben, si, 25/05/2018)

Ben's interview excerpt accurately highlights the rationale for the code developments featured in Figure 14. Notably, the musical blend shaped the rationale for selecting sounds with the following quote: "I think maybe we went through about 3 ideas. I think we picked this one because we think it fitted best with the other sounds" (Ben, si, 25/05/2018). Thus, Figure 14 is an example where being incremental and iterative supported a higher quality of meeting learning outcomes in music and programming. However, prior programming experience reported by both Ben and Norman most likely contributed to the level of sophistication in this example.

In contrast, Figure 15 presents an example of less exploration and refinement in being incremental and iterative from Ava and Sophia's music composition. Five modifications are observed over Lessons 4 and 5:

- lines 18 and 19 were removed from Lesson 4 in the Lesson 5 code
- the collection of notes changed to two notes in line 16 in Lesson 5
- a sustain parameter is added on line 16
- the synthesiser sound changed from "beep" to "prophet" on line 14
- the attack and sleep values were modified on lines 16 and 17.

All qualitative data revealed no rationale for these changes in Figure 15. However, this example demonstrates less refinement and exploration when compared to the changes made in Figure 14. The code block in Figure 15 also musically blended less effectively in the context of the full music composition (for example, the two notes drown out other sounds), which contributed to an overall uni-structural grade in music and programming (2/5). Thus, being incremental and iterative contributed to the quality of refinement and musical blend students achieved in their final music compositions.

Saved code end of lesson #4

```
use_synth :beep
loop do
  play choose([:d3, :b3, :g3]), attack: 1
  sleep 2
  play choose([:a3, :c3]), attack: 1
  sleep 1
end
```

Saved code end of lesson #5 (final)

```
use_synth :prophet
loop do
  play choose([:d3, :g4]), attack: 2, sustain: 3
  sleep 4
end
```

Figure 15. Excerpts from Ava and Sophia's group project code.

All qualitative data presented only vague glimpses of students being incremental and iterative. This is because the evidence gathered focused on a different aspect of their music compositions in each reflection. For example, some students reflected on what they thought needed work in the next lesson in a general way: "maybe add a piano next time" (Sam, sr, 16/05/2018), and Oliver reflected briefly "add more sounds" (sr, 11/05/2018). Therefore, the design of student and post-lesson researcher reflection data largely failed to provide the desired descriptive evidence of students being incremental and iterative.

Testing and debugging. Brennan and Resnick (2012) define testing and debugging as "strategies for dealing with – and anticipating – problems" (p. 7). All class quizzes for programming were designed to test debugging skills (see Appendix C). A box and whisker chart (see Figure 5 in this Chapter) for these quizzes indicate the interquartile range and upper whisker are all above 50% for all lessons. These results suggest the majority of students were successful in deploying testing and debugging strategies over the whole unit of work. However, post-hoc tests using Turkey's HSD exposed the quiz results for Lesson 3 (algorithms in music) and 4 (making a synthesiser) were statistically significantly lower than all other lessons. This significant reduction in quiz scores suggest students had less ability to employ effective testing and debugging strategies to support the learning outcomes in Lessons 3 and 4.

These quantitative results are limited in understanding how students used testing and debugging to answer quiz items and solve problems in their music compositions. To analyse the qualitative data for this CT element, a coding scheme recommended by Lye and Koh (2014) is employed to illuminate strengths and weaknesses of the different phases in testing and debugging. This coding scheme consists of four categories: (1) understanding the problem, (2) devising a plan, (3) carrying out the plan, (4) and reviewing the plan (Lye & Koh, 2014, p. 58).

The methods capturing evidence for the *devising a plan* and *carrying out the plan* phases were discovered to be significantly limited because they required students to reflect and remember processes they were not consciously monitoring. For example, some interviewees said they could not remember when asked about their testing and debugging processes:

CP: Did you try any other ways to solve the same problem?

Laura: I can't really remember sorry. (Laura, si, 25/05/2018)

Consequently, only a small fraction of insight was gained in how students used testing and debugging to support learning outcomes. This difficulty of capturing evidence in CT practices reinforces limitations identified in the CT studies (Brennan & Resnick, 2012; Lye & Koh, 2014) and the development of metacognitive skills in adolescents (Flavell, 1979).

It was discovered that there was no code commenting upon analysis in all final project code. Comments enable others to read and make modifications to code more easily (Lye & Koh, 2014). In the delivery of the unit of work, code commenting was not taught to students. Therefore, all students were likely not aware that code commenting is a common and necessary practice in programming, which could help students to solve bugs they encounter. This finding indicates that if students do not encounter this concept, they will not use it.

Understanding the problem. Throughout lessons, many students reacted to unexpected audio output through verbally describing problems they encountered. For example, Jacob said out loud in Lesson 3: “My scale is too loud, and I need it to be quieter" (rr, 16/05/2018). Written reflections also revealed instances of students describing specific problems: for example, Emma wrote: "Our drum didn't fit in timing with the melody" (Emma, sr, 18/05/2018). These examples indicate students were understanding the problem they encountered through verbal or written descriptions of the audio output and the desired musical effect.

The majority of students' problems were a consequence of incorrect spelling of Sonic Pi commands (rr, L1 to 6, 09/05/2018 to 25/05/2018). For example, Norman said he struggled most over the whole unit of work with "fixing things when they didn't work, like the program wouldn't run" (Norman, si, 25/05/2018). These syntax errors were often understood and resolved from simply reading and making inferences from the resulting console error messages. For example, when the user forgets to add a sleep command inside a loop, Sonic Pi displays "Thread death! Loop did not sleep or sync!". Thus, typing accurately and recognising syntax errors were a regular barrier for Norman and other students.

Devising a plan. Few examples of qualitative evidence were found of students devising a plan for problems. Moreover, descriptions were often general and non-specific. For example, Sarah described a bug she encountered when her chords sounded out of time:

> CP: Can you describe a time when your code didn't run as expected?
>
> Sarah: Umm, there was a time when it was playing the chords in the wrong place.
>
> CP: Can you describe how you devised a plan to solve this problem?
>
> Sarah: I think we just changed it so it was in the right timing
>
> CP: How did you find the solution?
>
> Sarah: I think I just tried different things out till it fitted. (Sarah, si, 25/05/2018)

Sarah recalled no plan to debug the problem she described beyond "tried different things till it fitted" (Sarah, si, 25/05/2018), which suggests she primarily employed a trial and error approach. Other evidence of students adopting a trial and error approach were also confirmed in their reflection diaries, for example: “we just played around till it worked” (Sophia, sr, 16/05/2018) and “we just changed it, so it was right then it worked” (Sam, sr, 23/05/2018). Thus, students were likely limited to trial and error when devising a plan to solve bugs in their code. If a trial and error approach was used exclusively, then this indicates the strategy of ‘thinking before doing’ approach emphasised in the teaching on the unit of work (see Chapter 3) was not being adopted by some students. However, other approaches may have been revealed if there were more effective methods for gathering data for this CT practice.

Carry out the plan. A consequence of students primarily using a trial and error approach in planning to solve problems is that no qualitative data was found for the *carrying out a plan* phase. Therefore, they were likely restricted in supporting learning outcomes with this phase of the debugging process.

Reviewing the solution. Students often expressed their excitement or disappointment when reviewing the success or failure of a trialled solution. This occurred in reaction by students primarily when audio output sounded (or not sounded) as intended. For example, Kate said: “that sounded right that time” (rr, 16/05/2018) after comparing a combination of notes for a melody. At times, solving problems spurred new creative ideas: for example, “We should try a higher note as well” (rr, 11/05/2018). These qualitative examples suggest students were reviewing the success of a solution according to personal preferences of both what enabled the code to run without errors and what sounded more pleasing.

Reuse and remixing. Brennan and Resnick (2012) define reuse and remixing as “building on other people's work” (p. 8). Throughout the unit of work, students were encouraged to collaborate, share and learn from one another. Over Lessons 1-5, reuse and remixing were frequently observed to occur between students, which supported the completion of final projects (rr, L1-5, 09/05/2018 to 23/05/2018).

Students were often motivated to reuse and remix from what sounded appealing upon reviewing each other’s work. For example, in Lesson 4 Ben and Norman shared their work after a request was asked to do so from James (rr, 18/05/2018). Thus, reuse and remixing helped to facilitate learning between students, which supported learning outcomes in music and programming through the sharing of skills and knowledge.

On close analysis of saved files across all lessons, a few examples of near identical code blocks revealed some evidence of reuse and remixing. For example, Figure 16 and Figure 17 are examples where both Liam's and Michael's code block are almost identical in Lesson 3.

```
myscale = [:A7, :D7, :C7]

live_loop :synth do
  synth :square, note: myscale.choose, cutoff: rrand(0.2, 40)
  sleep 0.2
end
```

Figure 16. Excerpt from Liam’s individual project saved code the end of Lesson 3.

```
4 myscale = [:G3, :F3, :E3, :A3]
5
6 live_loop :synth do
7   synth :square, note: myscale.choose, cutoff: rrand(0.2, 60)
8   sleep 0.2
9 end
```

Figure 17. Excerpt from Michael's individual project saved code the end of Lesson 4.

However, if we look at Liam’s project code saved in Lesson 5 (see Figure 18), Michael *remixed* Liam’s original idea to be unrecognisable from the code block in Figure 16, whereas Liam removed this code block from his project. For example, on line 19 in Figure 17 in Michael’s example, a random number is generated and stored in the variable ‘r’, which is used as a sleep time on line 21. These changes result in an irregular rhythm on each iteration the loop sounds. Furthermore, the original code on line 7 in Figure 17 changed further, which is seen in Figure 18 on line 20 with many changed and added parameters. Despite the obvious copying of code blocks in Michael’s code in Figure 17, he subsequently remixed and personalised this idea significantly. Thus, the level of remixing in this example helped Liam to achieve a relational programming grade (4/5) for his individual project.

```
myscale = [:G3, :F3, :E3, :A3, :B3]

live_loop :synth do
  r = rrand_i(0, 4)
  synth :hollow, note: myscale[r], amp: rrand(0.2, 0.6), attack: rrand(0.3, 3), decay: 0.2, release: 2, pan: rrand(-1, 1), sustain: rrand(0.5, 2)
  sleep rrand(0.2, 4)
end
```

Figure 18. Excerpt from Liam's individual project saved code at the end of Lesson 5.

However, no qualitative evidence was found where students described how they reused and remixed code. For example, Norman's example in Figure 19 and interview excerpt below reveal a lack of understanding of the code he was remixing from Ben's work:

```
12 with_fx :panslicer, phase: 0.21, wave: 0, mix: 1.0 do
13   loop do
14     sample :guit_harmonics, amp: rrand(0.2, 1.5)
15     sleep rrand(5, 11)
16   end
17 end
```

Figure 19. Excerpt of Norman's final individual project code.

> CP: Can you point an idea that was reused and remixed from code other than your own?
>
> Norman: Yeah, this code block [points to lines 12-17 in his final project in Figure 19] was used from Ben's code in lesson 3.
>
> CP: Can you explain what you changed and how you changed Ben's original idea?
>
> Norman: Um, I think I just changed the sounds and numbers and stuff. I can't remember exactly what I did sorry.
>
> CP: Can you describe how line 12 works? What is a *panslicer* and how is it changing the sound?
>
> Norman: Umm, I don't really know what a *panslicer* does exactly – I just thought it sounded cool. But I was just playing around and using Ben's code for my project. (Norman, si, 25/05/2018)

This interview excerpt suggests evidence from final project code is not sufficient to indicate understanding of learning outcomes. Norman's lack of awareness also highlighted an issue of authorship with individual assessment, where the practice of reuse and remixing made less clear what was understood by each student. Thus, the practice of reuse and remixing made music composition grades less reliable as a measure for what learning outcomes individual students achieved.

Abstracting and modularizing. Abstracting and modularising is defined by Brennan and Resnick (2012) as "building something large by putting together collections of smaller parts" (p. 9). On close inspection of all final code for each music composition, the Sonic Pi syntax of live loops were almost exclusively used as code blocks that abstracted musical sequences, which made a modular structure. Live loops enabled students to layer repetitive and musical sequences of sounds. Thus, the support of learning outcomes with this CT strategy was made possible mostly through Sonic Pi's design, where students arranged many code blocks containing live loops.

There were no obvious differences in the way students used a modular structure in their final code from the grade they received. All five interviewed students were able to successfully describe the output of each code block in their individual projects. For example, Emma said (see Appendix F to see Emma's final project code):

> CP: Can you talk about what is going on in each of your four code blocks at the end of your saved project for lesson 5?
> Emma: Umm, ok. Like the first one just plays random notes. The second also just plays random notes but from a scale in the program. This one plays a phone sound but we slowed it down so it makes it sound a bit scary. The last plays some chords kind of randomly as well. (Emma, si, 25/05/2019)

This interview excerpt indicates Emma accurately summarised the resulting output of each code block (or module) in her final project. Emma's explanation of her final code suggests she was thinking of each module as a different instrument or musical sequence that made up a texture of sound for the whole music composition. However, due to time constraints, post interviews did not go into details on how and why Emma arranged each module the way she did. Nevertheless, Emma's interview excerpt suggests she was aware of how each code block contributed to her whole music composition.

However, on analysis of all 33 final projects, only 63 of 121 (52%) functions (or modules that needed naming) were evaluated as using descriptive names. Appropriate names could include a description of the audio output for easier code readability. For example, Emma's exemplar (see Appendix F) on line 1 uses "rad" and on line 12 "th", which is not descriptive; a possible descriptive name for Emma's code block could have been "melody1" on line 1 and "melody2" on line 12.

There are two likely explanations for the lack of descriptive function or module names found in students' code. Firstly, some students reported they did not have enough time to finish their music compositions (see sub-heading Expressing in this Chapter in Section 4.1.3). Second, the functions were never called in other parts of students' code. The connection of modules is one important reason to describe the structure of code through descriptive naming (Brennan & Resnick, 2012). For example, the logic of how functions connect to one another is more easily followed and understood with descriptive names, which decreases the chance of errors and bugs when changes are made. Therefore, students had limited time and reasons to implement descriptive function names in their projects with Sonic Pi.

4.1.3 CT perspectives. Computational perspectives focus on the shifts in perspective observed in young people as they develop their computational thinking (Brennan & Resnick, 2012). Brennan and Resnick (2012) describe three elements that make up this dimension: *expressing*, *connecting*, and *questioning*. This section focuses on how the data coded and analysed under each element of the CT perspectives dimension supported the set learning outcomes for programming and music.

Expressing. The CT perspectives element of expressing is defined by Brennan and Resnick (2012) as "something they [students] can use for design and self-expression" (p. 10). In this unit of work, the majority of students successfully expressed themselves through the creative activity of composing music through the Sonic Pi platform for the project brief given. See student complete exemplars of final project code from four students that received different grades in Appendix F. For example:

- In Lesson 2, all students explored making their own scale as a collection of notes that expressed how they felt about the topic of their chosen video (see Ben's exemplar in Appendix F lines 35-47 for a student example)
- In Lesson 3, many students explored the use of random number generators to algorithmically generate music within set parameters (see Emma's exemplar in Appendix F lines 11-17 for an example)
- In Lesson 4, students successfully sculpted waveforms with code to make their own electronic instrument (see lines 13-16 in Grace's exemplar in Appendix F).

Zach confirmed that the expressive freedom in Sonic Pi for the project brief was appealing to students:

> It is nice for students to start from a point where you free yourself from this constraint and simply deal with how do I feel about this sound? and how do these two notes put together make me feel in relation to this topic depicted in this video?... I was also surprised how much students cared about these issues and were thinking creatively about each project's merits in response to the issue. (Zach, ti, 25/05/2018)

Zach's evaluation was confirmed by Norman through providing some insight into how he expressed himself: "I think basically I was trying things out and trying to get sounds that suited the video… I was looking for kind of slow sounds or depressing sounds because the film was about how people don't care enough about global warming" (Norman, si, 25/05/2018). These participant reflections suggest many students found expressing themselves in accordance to the project's brief positively influenced their level of engagement, which likely helped to support the set learning outcomes. However, as evidenced in the analysis of saved code in Sections 4.1.1 and 4.1.2 in this Chapter, students who developed more skills and knowledge were able to express themselves with more scope in Sonic Pi (see Appendix F for student exemplars).

Limited time to complete projects. Notably, some students reported struggling with the completion of both projects. For example: "it was really difficult to get both projects in on time" (Ben, sr, 25/05/2018) and "I wish we had more time" (Oliver, sr, 25/05/2018). Observations also noted similar comments made out loud by students (rr, 23/05/2018). These reflections suggest some students were not able to fully express themselves with a lack of time, which is likely to have limited their ability to meet the set learning outcomes.

Programming in Sonic Pi versus Python. Ben, Laura, and Sam commented in their post-unit of work interviews on what aspects they thought programming in Sonic Pi is different to programming in Python. For example, Ben said:

> I think in some ways it was more free than making a game or something… like because music is maybe just sounds, in Python you have to make something work with logic and stuff like a score or something and it takes a while to figure it out if you don't know what you're doing. (Ben, si, 25/05/2018)

Sam and Laura confirmed Ben's assessment on these different aspects between Sonic Pi to Python. Thus, some students who had prior experience with learning Python thought programming in Sonic Pi was comparatively easier to express themselves in.

Music genre. In contrast to the positive reflections with the freedom given in the project brief, three students reported they would have preferred to learn about making familiar forms of music. For example, Grace said "[I] didn't like some of the music – I'd rather learn about pop or more modern music we listen to" (Grace, sr, 25/5/2018), and Sophia said "be cool to learn how to put in drums and vocals and stuff and different styles" (Sophia, sr, 25/5/2018). Zach did not confirm this objection from his post unit of work interview. Nevertheless, Grace and Sophia's comments suggest some students would have preferred to express themselves through being taught how to compose in styles of music they were familiar with.

*Creative identity***.** A few students had indicated pre-existing mental barriers in their creative abilities. Despite agreeing that programming can be creative in their pre and post questionnaires, Sarah and Sam revealed that they did not think of themselves as creative individuals: “I'm like just not very good at art and stuff” (Sarah, si, 4/5/2018) and “I have no idea how to make a song so I don't think I'll be good at it” (Sam, si, 4/5/2018). For Sarah and Sam, viewing their creative abilities as limited may have contributed the extent they expressed themselves to their full potential in their music compositions. This suggests some students have self-limiting beliefs and assumptions on the subjects they are learning when engaging with creative activities.

Connecting. The CT perspectives element of connecting is defined as “interactions with others” (Brennan & Resnick, 2012, p. 10). One primary way students connected in this unit of work was through the development of their project in pairs. As reported under the CT concepts sub-heading in this Chapter (see Section 4.1.1), the individual marks and group marks were close with a strong correlation between both projects (programming r = .71 and music r = .88). Additionally, no student received the lowest pre-structural grade (see Appendix I for all student grades). Thus, the analysis of final grades suggest students shared knowledge and skills between their group and individual projects.

Seven students confirmed they preferred to work in pairs in their post-lesson reflections: for example, “I didn't like just working by myself all the time it was boring” (Emma, sr, 16/05/2018) and “it was maybe a bit hard to start again and do something by ourselves without help” (William, sr, 16/05/2018). Laura reinforced these reflections in her post-interview because: “... you get more ideas and it is more interesting, we could work together, and I didn’t get so bored.” (Laura, si, 25/05/2018). Similarly, Norman said he liked working in pairs because “you have someone there to talk to and figure things out with.” (Norman, si, 25/05/2018). These reflections suggest collaboration was cooperative for many students, which likely supported their understanding of learning outcomes. However, Sam revealed he preferred working individually because: “... I think sometimes [William] would be really dumb or he didn’t know as much as me, and so I ended up doing most of it.” (Sam, si, 25/05/2018). Thus, these reflections indicate the success of group work relies upon cooperation and participation of all group members, otherwise, groups projects may negatively impact the achievement of learning outcomes.

All interviewed students were also asked whether they had disagreements and how they were resolved. Sarah, Norman, Laura, and Ben reflected they had to compromise on their idea at times and try another they both agreed upon instead. For example, Ben reflected, “Like sometimes I thought what he thought sounded good sounded really bad, so sometimes I had to just go with it.” (Ben, si, 25/05/2018). Negotiating these differences likely supported learning outcomes because another student’s perspective inevitably had a different set of knowledge and skills to expose their partner to.

Throughout the unit of work, students often gave feedback across groups and individuals, which influenced how they made creative decisions. Many students often spontaneously asked others to listen to progress: for example, “Hey, come and listen to this!” (rr, Oliver to Emma, 11/05/2018) or “Can I have a listen?” (rr, Charlotte to Emma, 18/05/2018). Occasionally, these impromptu reviews led to the reuse and remixing of each other’s code, which indicated students were generally observed to be open to sharing their work with others (also see the Reuse and Remixing sub-heading in this Chapter in Section 4.1.2). Upon listening, many students were also overheard to give general feedback based on progress made; for example, “I liked it more like it was before” (Sarah, rr, 23/05/2018). At times, specific suggestions were made as a result: “I think that drum is too loud” (Liam to James, rr, 16/05.2018) or “It needs a bass or something low” (Laura to Emma, rr, 23/05/2018). These interactions directly contributed to the development of students’ music compositions, which positively supported learning outcomes. Zach reinforced these observations in his post-interview, where he reported “students were communicating through music concepts and ideas...” (Zach, ti, 25/05/2018). These reflections and observations suggest students connecting with each other regularly supported learning outcomes through giving and receiving feedback on their music compositions.

Questioning. Brennan and Resnick (2012) define the CT perspectives element of questioning as how students "feel empowered to ask questions about and with technology " (Brennan & Resnick, 2012, p. 11). Most observed student questioning occurred between students when they were listening to other projects and giving feedback (rr, 09/05/2018 – 25/05/2018). For example, it was observed that Michael said to Sophia in Lesson 2 "how did you make that sound" (rr, 11/05/2018). This suggests students were asking questions to help solve problems and realise ideas to support the development of their music compositions.

Post-lesson reflection notes indicate the students who received uni-structural grades (Ava, Aiden, Daniel, and Lucas) needed help with syntax related problems more often than other students (rr, 09/05/2018 – 25/05/2018). Instead of asking a targeted question, these students often voiced comments like "it doesn't work" (rr, 18/05/2018). This observation suggests students with limited knowledge and skills need more support in formulating targeted questions to help support learning outcomes.

Brennan and Resnick (2012) state questioning in CT is also about debunking assumptions with and about technology. Many assumptions about programming and music composition from students were challenged with Sonic Pi in this unit of work. For example, six students commented they were surprised that making music with code was possible in the post-lesson reflections in Lesson 6: e.g. "I was really surprised you can code music!" (Elizabeth, sr, 25/05/2018). However, assumptions were most often debunked not through questioning generated from a student's voice. Rather, they were debunked because it was required in understanding how to use some Sonic Pi commands (e.g. play commands need a sleep time to create rhythm in music). These observations suggest students were limited in formulating questions that challenged their assumptions about the behaviour of Sonic Pi.

4.1.4 Summary. Overall, the results across all dimensions from the CT framework by Brennan and Resnick (2012) indicate they were all able to support learning outcomes in music and programming. However, they were utilised to mixed degrees in their scope and rigour. For example, trends in the data across each dimension consistently reinforced that students who received higher grades and quiz scores had explored, experimented and refined ideas to a greater extent than those with lower grades and scores. Those students with prior programming experience revealed some evidence of knowledge and skill transfer from other programming languages, which likely contributed to higher grades and quiz scores for these students. A summary of findings for each CT dimension is presented as follows:

CT concepts. Mean grades indicate most students were able to achieve at least a multi-structural grade (3/5) for projects in music and programming, which suggested students knew how to apply many CT concepts. The end of class quizzes also reinforced this skill with the majority of students having correctly answered each quiz at least half of the time in both subjects. However, quiz items in Lesson 3 (on algorithmic music) and 4 (on making a synthesiser) were statistically lower in music and programming when compared to previous lessons. These findings suggested students struggled with the CT concepts of conditions and operators in particular. Still, mean quiz scores were found to be all close to 50% for Lessons 3 and 4. Additionally, effective overlapping CT concepts that supported learning outcomes in programming and music composition were found to be between:

- sequences and melodies
- loops and repetition (additionally, ostinatos and riffs)
- parallelism and the layering of sounds
- lists (data) and making a musical scale.

The live loop in Sonic Pi enabled the layering of sounds, which was almost exclusively used as the only loop employed by all students. Saved code across all music compositions indicated there were an average of four live loops per project.

Over half of the variables used across students' projects (16/28) had non-descriptive naming, which may present a barrier for future growth in programming. 20/28 of these variables were called upon only once in each project, suggesting students had little reason to use descriptive variable names.

Students were first observed to understand conditions and operators, but could not answer quiz questions and integrate these CT concepts into their final project. Additionally, the three students who integrated conditions and operators in their music compositions all had prior programming experiences. These findings indicate conditions and operators did not support learning outcomes for almost all students.

CT practices. The findings for the CT practices dimension are considered limited because the data gathered only provided fragmented snapshots. Most notably, all phases of Lye and Koh's (2014) debugging phases indicated a limitation in both data gathering and the range of approaches students employed for the testing and debugging CT practice. For example, a trial and error approach was likely used as a primary debugging strategy by many students. Additionally, syntax errors were reported as a major struggle for many students. In contrast, the CT practice of being incremental and iterative was found to support the development of music compositions effectively for many students. However, being incremental and iterative may not be as appropriate in a live-coding context, which is promoted as the key activity in Sonic Pi over music composition (Aaron, 2016).

The reuse of Sonic Pi code from other students occurred frequently, which facilitated sharing of skills and knowledge between students to support the development of their music compositions. However, the CT practice of reuse and remixing made the assessment of individual learning outcomes less reliable. For example, students could not remember how they remixed code, which highlighted an issue of authorship with individual assessment.

An average of four loops were found per music composition (suggesting an average of four sounds or sequences in each). However, the connecting of modules (also known as functions) programmatically (and not musically) were found to be lacking in their code (through the calling of functions or modules). This lack of connection may have contributed to the finding that 52% of modules had non-descriptive names, which was a similar finding to students' naming of variables in CT concepts dimension.

Functions were generally not connected in students' code and never found to be calling each other, which suggests a reason for non-descriptive names being used.

CT perspectives. The creative activity of music composition with code for a purposeful outcome was a common aspect found to be engaging for the expressing element. Notably, interviewed students who had prior experience with learning Python said programming in Sonic Pi was comparatively less restrictive and more expressive. However, three students (that were not interviewed) said they would have preferred to be taught how to make music within a genres they were familiar with (like pop music). Two students also indicated they had pre-existing mental barriers in their creative abilities, which may have limited the extent students expressed themselves to their full potential.

The group project, planned feedback activities, and final presentations in Lesson 6 were found to be particularly beneficial for the Connecting element, which helped to provide ideas and develop students' music compositions. Most students stated they liked working in pairs, but Sam reported he liked working individually. This difference in student preference for working on projects suggests working in pairs relies on cooperation and can either support or negatively impact learning outcomes.

Sonic Pi challenged assumptions about the behaviour of programming languages, which led to a greater understanding in music and programming. For example, six students highlighted they were surprised that coding music compositions was possible. However, greater understanding was generally not as a result of questioning from a student voice, rather from the teacher questioning students. This lack of student voice in questioning suggests the they were limited in knowing how to ask questions that questioned their assumptions. Although, many students were often observed to ask questions between each other when giving and receiving feedback on their music compositions.

4.2 Sub-Research Question: Attitudes towards Programming

The attitude subscales by Teo (2007) were used as a framework and modified for the sub-research question (*enjoyment*, *importance*, and *self-confidence*). As mentioned, the statistical analysis of the pre and post questionnaire attitude items are presented first and followed by sub-headings that focus on the findings under each subscale.

4.2.1 Pre- and post-Questionnaire Attitude Items. A pre (T1) and post (T2) questionnaire (see Appendix D) were taken by all student participants to help measure changes in attitudes in programming and music for the sub-research question. Based on Cohen's (1992) guidelines (strong > .65; moderate between 0.35 and 0.65; and weak between .2 and .35), Table 6 indicate *t*-test and Cohen's *d* values for programming and music have strong effect sizes with minimal *p*-values of < .001. All programming attitude subscales indicate higher *t*-values than music, suggesting the unit of work had a more significant difference in all attitude subscales for programming in comparison to all attitude subscales for music. The greatest positive increase is the attitude programming subscales for importance ($t = 12.28$, Cohen's $d = 1.72$, and $p < .001$) and self-confidence ($t = 12.28$, Cohen's $d = 1.72$, and $p < .001$).

Table 6

T-test, p-value, and Cohen's d calculations for programming and music attitude subscales (T1 and T2 questionnaire; n = 22)

	Programming enjoyment	Music composition enjoyment	Programming importance	Music composition importance	Programming self-confidence	Music composition self-confidence
t-value	8.39	6.07	12.28	6.49	8.71	7.21
Cohen's *d*	1.31	1.39	1.72	1.48	1.54	.70
p-value	$p < .001$	$p < .001$	$p < .001$	$p < .001$	$p < .001$	$p < .001$

Notably, the smallest Cohen's d effect size from T1 and T2 is the music self-confidence attitude subscale ($d = .70$). Students may have had a greater prior self-confidence in music because the unit of work was conducted with music students. This explanation is confirmed with positive T1 responses for music (*M* = .39) in comparison to programming (*M* = -.98) for the self-confidence attitude subscale (a 1.37 difference).

Strong correlations are indicated when comparing all T2 subscales between programming and music (enjoyment r = .82, importance r = .68, self-confidence r = .83). According to Dowdy et al. (2004), a risk of multicollinearity can occur when there is a linear relationship among two or more independent variables with a variance inflation factor (VIF) above 5. Tests indicated a low level of multicollinearity for all attitude subscale correlations were present (enjoyment VIF = 3.28, importance VIF = 1.72, and self-confidence VIF = 1.89). As a result, these strong correlations may suggest students' attitudes may have increased because of: (a) exposure to music composition and programming; (b) the unit of work; (c) or the Sonic Pi platform. However, because there was no study to compare results with, the degree possible explanations may have been factors for these correlation results could not be determined.

Zach's attitude results. Table 7 show the results for the attitude questionnaire items Zach answered in T1 and T2. Except for programming enjoyment, findings indicate a positive shift in all subscales for music and programming. However, enjoyment for both subjects were closely positive with a .5 difference before and after the unit of work. The most significant differences from T1 and T2 indicate Zach's perception of programming increased in the importance subscale (a +3 difference); his self-confidence in programming (a +2 difference) and music composition (a +2.33 difference) also increased significantly.

Table 7

Zach's T1 and T2 attitude results scaled from -3 (negative) to +3 (positive)

	Programming enjoyment	Music composition enjoyment	Programming importance	Music composition importance	Programming self-confidence	Music composition self-confidence
T1	2.5	2.5	0	2.5	-.75	0
T2	2	3	3	3	1.75	2.33

Perceptions of creativity in programming. The T1 and T2 questionnaires asked: "Do you think programming can be creative?" to measure potential changes in students' perception of the role of creativity in programming. T1 indicated 12 out of 22 students answered this question "yes", which increased to 20/22 in T2. Only Ava and Aiden answered "no" in T2; no explanations were found why in all qualitative data gathered. Nevertheless, this increase between T1 and T2 suggests eight students reconsidered the role of creativity in programming through participating in the unit of work. As this study employed an quasi-experimental single-case design, it is difficult to conclude if exposure to programming specifically with Sonic Pi and this unit of work would have different results with another programming language or another unit of work. This notion is tentatively confirmed when seeing the T1 measure responses for the ten students having prior experience already all thought of programming as creative. Thus, exposure is potentially a major contributing factor to this increase in students' perception as to whether programming can be creative.

4.2.2 Enjoyment. Figure 20 provides a box and whisker chart as a visual representation for the enjoyment subscale in programming. This chart indicates the interquartiles and upper whisker increased to above zero in T2, which suggests positive attitudes for 75% of students. The mean also confirmed an increased from T1 (M = -.45) to T2 (M = 1.08) by 1.53. Standard deviations remained comparatively similar T1 (SD = 1.35) to T2 (SD = 1.21). With the high t-test value and Cohen's d effect size (t = 8.39, Cohen's d = 1.31, and p < .001), these results strongly suggest the unit of work increased the enjoyment subscale for most students in programming significantly.

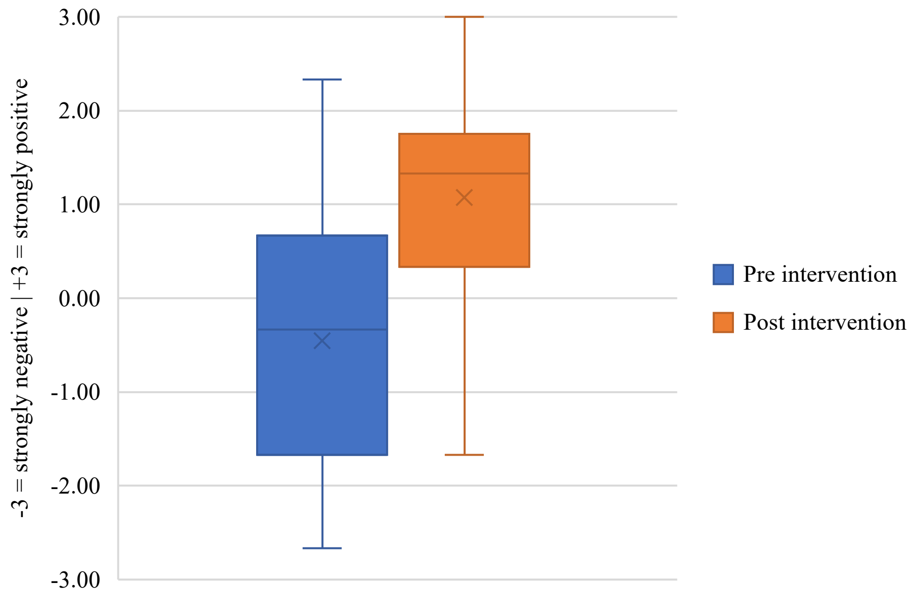

Figure 20. Box and whisker plot chart for the enjoyment subscale in programming.

A weak correlation was found between the enjoyment subscale in programming (T2) and students' total quiz scores for programming (r = .18). This correlation suggests students who achieved higher quiz scores had a weak chance of finding more enjoyment towards programming. However, a moderate correlation between this subscale and final individual project grades for programming was found (r = .32), which suggests students with higher individual programming grades had a moderate chance of finding more enjoyment towards programming.

Qualitative data. Post-lesson observation notes reveal all students were generally on task and engaged in all lessons (rr, 09/05/2018 – 25/05/2018). Additionally, Zach confirmed this positive observation of students' enjoyment in his interview: "students responded very positively to making music in this way, and for the most part, they were engaged and positive" (Zach, ti, 25/05/2018). These reflective observations from both participant teachers reinforce the quantitative results that the majority of students enjoyed programming by the end of the unit of work.

Questionnaire data revealed students' enjoyment of programming in response to the short answer question "Write a few short sentences about your feelings towards learning more Programming or Coding" in the pre and post student questionnaire (see Appendix D). These responses were evaluated as either positive or negative: 14 of 22 T1 responses were evaluated as positive, which increased to 20/22 positive responses in T2. Three notable contrasting changes from T1 to T2 in this short answer question were found from Jacob, Charlotte, and William:

Jacob: "boring" (Jacob, T1, 4/05/2018)
Jacob: "coding is rel[l]y fun and interesting, I liked working with my fr[i]end on cool sounds" (Jacob, T2, 25/05/2018).

And:

Charlotte: "not sure sorry" (Charlotte, T1, 4/05/2018)
Charlotte: "I feel quite good, it was quite fun thanks!" (Charlotte, T2, 25/05/2018).

And:

William: "OKK" (William, T1, 4/05/2018)
William: "coding can be rel[l]y cool, I feel rel[l]y good" (William, T2, 25/05/2018).

These three responses reinforce the quantitative results that significant positive changes occurred in students' perception of programming before and after the unit of work. The two negative responses emerged from the T2 responses were: "it did get a bit hard" (Sophia, T2, 25/05/2018) and Ava "[con]fusing" (Ava, T2, 25/05/2018), which suggest the difficulty of the unit of work presented the most significant barrier to enjoyment for Ava and Sophia. Nevertheless, the majority of data confirms that most students enjoyed programming after the unit of work.

End of class reflections. The end of class reflection in Lesson 6 asked students to evaluate the unit of work, which were summarised as follows:

- Eight students commented they found the novelty of coding music in music class the most enjoyable aspect. For example: "coding music was actually great. I was surprised how into it I was" (Sam, sr, 25/05/2018).
- Similarly, five students specified they particularly enjoyed coding music. For example: "i was surprised how fun it was. it is re[al]ly cool coding music!!!!! i re[al]ly liked learning about it" (Ben, sr, 25/05/2018).
- Four students commented specifically they liked the creative aspect of programming. For example: "I liked working on making and creating stuff" (Charlotte, sr, 25/05/2018).
- However, four responses were evaluated as incomprehensible. For example: "dkjvnksn", (Ava, sr, 25/05/2018).

Similar to all other findings, the end of class reflection in Lesson 6 mostly confirms that the majority of student participants enjoyed programming in Sonic Pi. These responses also helped to identify two themes that emerged from the qualitative data: (1) *novelty of making music with code* and (2) *working in pairs*.

Novelty of making music with code. A theme around students enjoying the novelty of making music with code was also common across interview data. For example:

- Laura said "I really like making my own music and it was cool to learn to make stuff with Sonic Pi" (Laura, si, 25/05/2018);
- Ben said he was surprised about how you could code music, "I think it is a different way of coding that I never thought about before. I think it is really cool" (Ben, si, 25/05/2018);

- Sam said he was mostly surprised that he "didn't think you could code music" and said that he liked coding in Sonic Pi more than he previous experiences with Python (Sam, si, 25/05/2018).

With support from the end of unit reflections in Lesson 6, the novelty of making music with code was found to be the most common theme that emerged explaining why students enjoyed the unit of work.

Working in pairs. Working in pairs emerged as one other of the notable themes explaining why students enjoyed programming in Sonic Pi (also see the sub-heading Connecting in this Chapter in Section 4.1.3). For example, "I liked working on the group project more" (Oliver, sr, 25/05/2018) and "it was cool to work with my friend" (Charlotte, sr, 25/05/2018). Post-unit of work student interviews reinforced this preference for working as Laura, Ben, and Sarah specifically commented they liked working in pairs more (si, 25/05/2018). However, Sam was one student countering this data, because he found his partner difficult to work with (see Connecting sub-heading in this Chapter in Section 4.1.3), which suggests working in pairs may have not have been enjoyable for all students.

4.2.3 Importance. Figure 21 box and whisker chart indicate the interquartiles and both whiskers increased to above zero in T2 for the programming importance subscale. However, three outlier cases (Grace, Sophia, and Ava) indicated below zero results for this subscale in T2. Comparing these individual three outlier cases with T1 results indicate a positive or negative average change within a range of 1 (see Appendix I), suggesting these students had slightly more negative attitudes in their perceived importance of programming after the unit of work. Nevertheless, mean calculations for all students indicated an increase of 1.88 (from T1 $M = -.65$ to T2 $M = 1.23$). Standard deviation results were similar but became narrower from T1 ($SD = 1.23$) to T2 ($SD = .95$). With supportive evidence from the high *t*-test value and Cohen's d effect size ($t = 8.39$, Cohen's $d = 1.31$, and $p < .001$), this quantitative analysis collectively suggests the unit of work increased the importance subscale for programming significantly for most students.

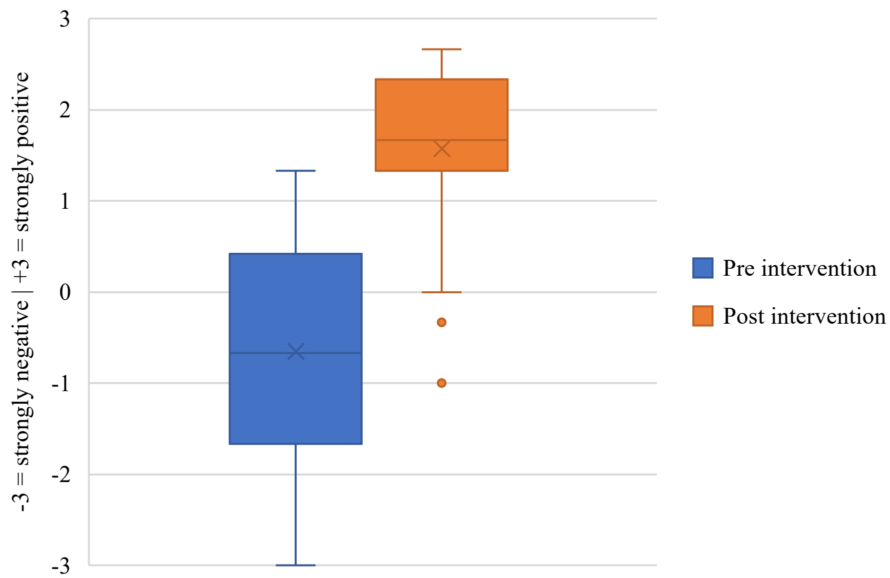

Figure 21. Box and whisker plot chart for the importance attitude subscale for programming.

A moderate correlation was indicated between results for this subscale and students' total quiz scores ($r = .32$). Additionally, a moderate correlation was also found between this subscale and all individual project grades for programming ($r = .45$). Thus, both correlations suggest a moderate chance that students who achieved higher quiz scores and grades for programming think of programming as more important.

Qualitative data. No post-lesson observation notes specifically reported on examples where students expressed their feelings on the importance of programming. However, the participant music teacher (Zach) thought:

> This is a really exciting way of linking two disciplines, which could help to influence the perception of music and creative disciplines in schools to be directly relevant to developing 21st century skills, thereby raising the profile of music in schools (increasing funding maybe?) and the

> perception of music as an essential subject in the minds of students, teachers, management and parents. (Zach, ti, 25/05/2018)

The potential for Sonic Pi to improve the perception that music can help students prepare for the future is seen by Zach as a notable benefit. This indicates that he thinks teaching skills in programming is valuable and sees Sonic Pi as a way to participate with the integration of the new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC). This interview excerpt also confirms Zach's questionnaire results, which indicated an increase in the importance subscale for programming (see Table 6 with a +2 difference).

Programming skills helps students prepare for the modern workforce. The idea that programming could help prepare students for the future emerged as the only common theme reflected in students' reflections and post unit of work student interviews for this subscale. For example:

- Norman thought he would study programming in high school and or university because "you could get a job that pays lots of money" (Norman, si, 25/05/2018).
- Laura perceived that programming will be an important skill to learn in the future because "we all use technology a lot" (Laura, si, 04/05/2018);
- Ben thought programming as valuable because "I think maybe everything is going to get taken over by robots, so I'll want to program robots to do things for us." (Ben, si, 25/05/2018).

Six similar comments were identified in the end of class reflection data from other students. A notable example of this perception by Olivia was expressed as "learning about this will probably be good for a job, so I think it woul[d] be good to do more of this" (Olivia, sr, 23/05/2018). Thus, programming was commonly reported as a valuable skill to learn by all participants to prepare for future employment.

4.2.4 Self-confidence. Figure 22 box and whisker chart provides a visual representation of the interquartiles and upper whisker had increased to above zero in the T2 questionnaire for the self-confidence programming subscale. A mean increase of 1.89 was indicated between the T1 mean (M = -.98) and T2 mean (M = .91). Standard deviation remained comparatively similar between T1 (SD = 1.09) and T2 (SD = .89). With the high t-test value and Cohen's d effect size for this subscale (t = 8.39, Cohen's d = 1.31, and p < .001), all quantitative evidence strongly suggests the unit of work increased self-confidence in programming significantly for the student participants.

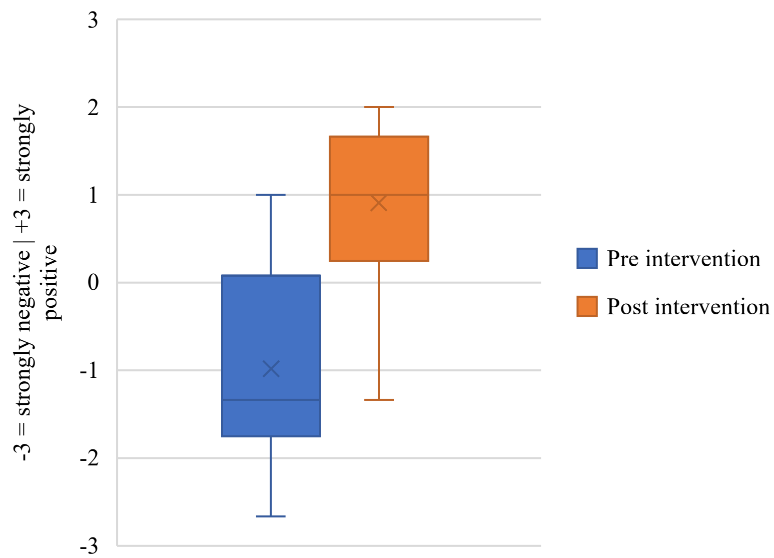

Figure 22. Box and whisker chart for the self-confidence subscale of programming.

Moderate correlations were found between this subscale and students' total quiz scores (r = .39). Additionally, moderate correlations were found between this subscale and individual project grades for programming (r = .43). Both correlations suggest a moderate chance that students who achieved higher programming quiz scores and grades had more self-confidence in programming.

In the pre-unit of work questionnaire, ten students reported prior programming experience in either Python or Scratch. The subsequent final programming grades for students' individual projects indicate the majority of relational (4/5) and extended abstract (5/5) grades had students with prior programming experience (see Appendix I):

- 80% extended abstract grade (5/5)
- 83% relational grade (4/5)
- 14% multi-structural grade (3/5)

- 0% uni-structural grade (2/5)
- No student received a pre-structural grade (1/5).

This implied relationship between prior experience to programming ability was reinforced in post unit of work student interviews. For example, Ben wrote, “I am excited to learn more about coding because I do a lot of it.” (Ben, si, 4/05/2018), and Norman “I’ve done a little bit of it before so I think I know that I might enjoy it” (Norman, si, 4/05/2018). These findings also confirm Zach’s prediction in his pre interview that this unit of work may work particularly well for those already who “had an interest in coding and technology in the class” (Zach, ti, 25/05/2018). Thus, these findings strongly suggest that students with prior programming experience had more self-confidence than those without.

Those students who had no prior programming experience commonly described their unfamiliarity towards programming prior to the unit of work starting. For example, Michael wrote: “I’ve never done it so dunno” (Michael, T1, 04/05/2018). The moderate correlations between this subscale and both students’ individual project grades for programming (r = .43) and quiz scores (r = .39) for the programming items confirm Zach’s pre-unit of work prediction that highlighted Sonic Pi will be “new for the majority of students, which will be challenging” (Zach, ti, 25/05/2018). However, Zach reflected that “students did seem uneasy at first but were quick to feel at home and feel familiar” (Zach, ti, 25/05/2018), which reinforces the T2 results that these students had gained self-confidence through participating in the unit of work. Some students also confirmed Zach’s observation in other qualitative data gathered, reflecting they had become a more confident programmer. For example, Sarah (with no prior programming experience) said, “I think because I feel better about it like I know more now” (Sarah, si, 25/05/2018). Thus, while students who were inexperienced in programming indicated a relatively low T1 self-confidence, the majority indicated an increase in this subscale by the end of the unit of work.

Programming perceived as difficult. A notable qualitative theme that emerged on students' self-confidence was the perceived difficulty of programming in student reflections:

- "Coding with Sonic Pi was quite fun, but it did get a bit hard" (Emma, T2, 25/05/2018)
- "feeling good, it w[a]s easy but some p[a]rts were quite hard" (Sophia, T2, 25/05/2018)
- "I liked it, but it was hard tho" (Daniel, T2, 25/05/2018).

The idea of programming as a difficult skill to learn is also reinforced in post-unit of work interviews. For example, Norman thought learning how to program in high school or university is more difficult than other subjects because "you need to be really really good at computers, like know so much stuff about them" (Norman, si, 25/05/2018) and Laura thought high school or university programming classes are "really hard [because] I think it seems really hard, you have to be good at typing a lot and solving things that are hard." (Laura, si, 25/05/2018). While the quantitative results indicated an increase in this subscale, these reflections suggest students' perception of programming as a difficult skill to learn may have limited their sense of self-confidence. Notably, no instances were recorded of students who thought programming is easy or comparatively difficult to other subjects.

Student creative identity. As discussed in the Expressing sub-heading in this Chapter (see Section 4.1.3), Sarah and Sam revealed they did not think of themselves as creative. This finding suggests students come with assumptions and biases about themselves and the subjects they are learning, which may inhibit their self-confidence to do creative tasks. The degree to which students viewed themselves as creative may have limited their programming self-confidence. This issue is of increased importance because the unit of work focused on the creative activity of music composition through programming in Sonic Pi.

*Zach's perception as a music teacher new to programming***.** Zach reported in the pre-interview he did not feel confident with both digital devices and music composition.

Due to inexperience, he admitted there were limitations in the extent to which he could help students. For example, he "felt it was limiting [that I didn't know how to help some students]... especially when they haven't really been given the tools to self-problem solve in this context" (Zach, ti, 25/05/2018). Despite this, Zach reported he is now more confident in implementing a unit on Sonic Pi in the future: "I think now that I have been through this unit of work with you, I think I now know what to do in order to develop my skills and to teach this in Sonic Pi" (Zach, ti, 25/05/2018) and later said without hesitation he would "strongly consider planning a unit with Sonic Pi in the future" (Zach, ti, 25/05/2018). These interview excerpts reinforced the quantitative results that Zach increased his self-confidence in teaching programming with Sonic Pi.

4.2.5 Summary. Overall, strong effect sizes indicated more positive attitudes towards programming in all subscales (enjoyment, importance, and self-confidence), which were mostly confirmed across results from qualitative analysis. The most significant increases found were seen in the programming importance and self-confidence subscales, suggesting the unit of work helped students to perceive programming as an accessible skill and more influential in their lives. Perceptions of creativity in programming also increased from T1 to T2 by eight (the majority in T2 thought programming can be creative 20 out of 22 students), indicating the unit of work helped students to see the role of creativity in this skill. Additionally, moderate correlations were discovered between individual project grades for programming and T2 results for all attitude subscales, suggesting high programming competence has a moderate chance to indicate relatively positive attitudes towards programming.

Findings about the participant music teacher (Zach) also indicated increases particularly in the importance and self-confidence subscales, which were confirmed across the qualitative data gathered. These results suggest Zach was able to overcome a lack of awareness and feelings of anxiety towards computing (indicated in the pre-unit of work interview) through facilitating the teaching with me.

The following notable insights were also discovered for each subscale from the qualitative data, which mostly reinforced the quantitative data:

Enjoyment. The novelty of making music with code and working in pairs was found to be the main reasons students enjoyed the unit of work. However, Sam reported he did not like working in pairs, suggesting not all students preferred to work collaboratively.

Importance. Zach and many students thought programming could help students prepare for the modern workforce. However, there was a lack of students reporting intrinsic motivations for learning programming.

Self-confidence:

- Students perceived programming as a difficult skill to learn regardless of their level of competence. However, quantitative data suggests students can increase their self-confidence despite students holding this perception.
- Those with no prior programming experience commonly described their unfamiliarity towards programming to describe their pre-unit of work feelings towards programming.
- Student self-perception on being creative may have reduced Grace and Sophia's self-confidence in programming, which could have been a limiting factor for other students.
- Zach increased his self-confidence in programming to the point he would like to include a unit on Sonic Pi in his future planning.

Chapter 5: Discussion

The aim of this Chapter is to provide an in-depth discussion of the findings and how they relate to the reviewed literature for each research question. The limitations of this study and suggestions for further research will then conclude this thesis.

5.1 Main Research Question Discussion

The main research question asked: *How can Computational Thinking (CT) support learning outcomes in Programming and Music with the Sonic Pi platform?* This question is answered through the CT assessment framework conceptualized by Brennan and Resnick (2012) with the dimensions of CT concepts, practices, and perspectives. First, this section discusses each CT dimension in detail and then concludes with overall implications and recommendations.

A trend applying to all CT dimensions from the findings indicated that the scope, refinement and depth of each were generally reflected in their grades and quiz scores. An explanation of this finding was found when discovering that the majority of students with the top two grade categories were dominated with most having had prior programming experience in Scratch and Python. This finding reinforces the notion from Brennan and Resnick (2012) that CT is transferable between programming languages. Thus, knowledge transfer between programming languages from prior experiences likely contributed heavily to students' CT competence to support learning outcomes. The implications for educators signals this issue may further add to the demanding interdisciplinary challenges in supporting a wide range of abilities in both disciplines. Thus, it is recommended educators consider more time may be needed to support students with less programming experience in their planning.

5.1.1 CT concepts. For all students in this study, findings suggest sequences and loops supported learning outcomes in music and programming. These findings are consistent with results in case studies at a school level in the programming platforms Scratch (Allsop, 2018; Brennan & Resnick, 2012; Burke, 2012; Lee, 2010) and Stagecast

Creator (Denner et al., 2012). However, Sonic Pi has the added benefit of highlighting overlapping concepts in both music and programming with sequences and loops. For example, melodies in music were easily grasped by all students while linking to the programming concept of sequences, and repetitive music composition concepts (e.g. ostinatos and riffs) also linked effectively while learning about looping in programming. Moreover, music listening activities, where the objective was to find repetitions (or loops) in a piece of music, were found to be effective for aiding students' initial understanding in both disciplines. However, it was surprising their use of the three looping syntaxes available in Sonic Pi were underutilised. This finding suggests loops were implemented by all students in a limited way. In particular, conditional looping (which combine the concepts of loops, conditions and operators) are not possible with a single command in Sonic Pi. The implications for educators are students' understanding of looping required less rigour when compared to traditional programming languages aiming to build software. While this implication suggests these concepts in Sonic Pi benefit beginners for an easier introduction to these concepts, it limits students' future growth in their CT to apply transferable learning from Sonic Pi to other programming languages (like Python) for building software.

Sonic Pi also helps to illuminate the CT concept of parallelism for beginners because each line of code simultaneously sounds unless otherwise programmed not to. Parallelism can often be a CT concept reserved later for non-beginners in programming to understand how computers run several computational processes at the same time (Aaron & Blackwell, 2013). Consequently, students did not anticipate that they needed to program a sleep command when composing melodies to ensure music notes did not play together as chords. This misconception emphasised the need to consciously program time (or the spaces) between notes, which supported music learning outcomes with the core concept of rhythm. The results in this study confirm Aaron and Blackwell's (2013) claim that Sonic Pi enables this concept to be more easily grasped by beginner students through coding music. This confirmation is also consistent with Brennan and Resnick's (2012) study, where they have found the CT concept of parallelism can also be understood by beginners in Scratch. Thus, Sonic Pi offers pedagogical benefits for educators on the CT concept of parallelism for beginner students with overlapping music concepts like rhythm, melodies and layering sounds.

The CT concept of data successfully supported learning outcomes for all students when they were tasked with selecting a collection of notes in a personalised musical scale (taught in Lesson 2). However, the majority of variables were given non-descriptive names and were only called upon a few times in their code. Other studies in Scratch have also found storing and using data in variables not to be challenging for beginners in programming but did not report on participants' naming of variables (Allsop, 2018; Brennan & Resnick, 2012; Burke, 2012; Lee, 2010). If the role of variables cannot be evaluated from their name, then the readability of their code is compromised, which may unnecessarily complicate the problem solving process when errors and bugs occur (Brennan & Resnick, 2012). The lack of information in the literature on this issue suggests further studies are needed to compare findings on naming variables for different contexts. Despite this concern, the results suggest this approach is an effective way for educators to adopt because it enables beginners to experiment and manipulate data in Sonic Pi with the concept of scales in music. However, educators should consider emphasising the habit of descriptive variable naming in their instruction to aid students' development of solving bugs and readability of their code.

Despite initial success from my observations when first introducing conditions and operators, quiz scores and analysis of students' code had disappointing findings for supporting learning outcomes. For example, only three students integrated these concepts into their final music compositions. These findings are consistent with evidence of concepts students find challenging in Scratch (Brennan & Resnick, 2012; Burke, 2012; Lee, 2010) and Stagecast Creator (Denner et al., 2012). However, Allsop (2018) countered these findings where she found only a few students having trouble implementing conditional logic with a game design task in Scratch. This contradiction in the literature suggests students' understanding of conditions and operators can vary significantly between studies. This issue also highlights that Sonic Pi does not have syntax that allows for conditional loops (e.g. '*for*' loops in Python), which combine the CT concepts of looping, conditions and operators. Nevertheless, the extent students in this study did not integrate conditions and operators into their final music compositions was not anticipated because this issue was not reported in the case studies on Sonic Pi to date (Burnard et al., 2014; Burnard et al., 2016; Cheng, 2018). Thus, implications for these findings suggest educators need to be especially aware of this limitation when planning interdisciplinary learning outcomes involving conditions and operators with Sonic Pi.

Conclusion. The overall implications are CT concepts offer beginner students insight into the overlapping concepts that support and do not support learning outcomes in both music and programming with Sonic Pi. In particular, the CT concepts of sequences, loops, parallelism and data yielded noteworthy evidence for interdisciplinary skills and knowledge transfer in music. However, the lack of descriptive variable names indicate a challenge for educators to consider carefully. In contrast, the lack of evidence gathered on the CT concepts of conditions and operators illuminate a concerning gap for beginner programmers with Sonic Pi. Despite Edwards (2011) highlighting that conditions and operators have existed for millennia across cultures in music, the way beginner students in programming and music composition implement these two CT concepts is not obvious. This major weakness has not been reported in existing case studies on Sonic Pi (Burnard et al., 2014; & Cheng, 2018). Thus, the implications for educators indicate that conditions and operators are challenging CT concepts to integrate into music compositions with Sonic Pi, which illuminates a likely gap educators need to anticipate in their planning. Therefore, it is recommended that educators:

- Consider utilising the overlapping CT concepts that were found to be successful (sequences, loops, parallelism, and data) in the way they were presented in this unit of work for beginner students.
- Emphasise the use descriptive variable names.
- Recognise conditional loops are not possible in Sonic Pi, which is a limitation to anticipate in planning learning outcomes.

- Anticipate the CT concepts of conditions and operators will likely be challenging to support learning outcomes in programming and music for beginners in Sonic Pi. Therefore, plan to teach these CT concepts with another programming language.

5.1.2 CT practices. *Being incremental and iterative.* All student participants in this study showed some evidence of being incremental and iterative, which helped to support learning outcomes through the development of their music compositions. This finding confirms the literature on this CT practice, where being incremental and iterative was found to be effective for supporting learning in other subjects, besides computer science (Brennan & Resnick, 2012; Fronza et al., 2017; Kafai et al., 2014; Lin & Liu, 2012). The studies on Sonic Pi by Burnard et al. (2014) and Cheng (2018) focused on live-coding performance, where the objective was to use Sonic Pi like a music performance instrument. It is suggested that being incremental and iterative is less appropriate in the live performance context because it involves *preparing* for a performance through practice instead of *developing* a music composition to be presented. For educators, the implications of these findings are that being incremental and iterative is an appropriate and effective way to support learning outcomes in both music and programming through the development of music compositions in Sonic Pi.

Testing and debugging. The findings for the testing and debugging CT practice were disappointing because the data gathered gave little insight into the four phases of the debugging process outlined by Lye and Koh (2014, p. 58). While there was evidence of some success with the first phase of *understanding the problem* in Lye and Koh's (2014) framework, the remaining three phases (*devising a plan*, *carrying out a plan* and *testing the plan*) were not illuminated effectively from the data gathered. However, the average quiz results (that quizzed students with debugging problems) in this study for all lessons indicated students experienced success when debugging. This contradiction in the data suggested that students were able to successfully solve debugging problems, but found it difficult to *describe* the process they adopted. One possible explanation could be that student participants simply struggled to remember details on exactly what they did when being asked afterwards in interviews and written reflections. If this is the case, more explicit teaching of debugging processes could have been helpful to help solve this problem. However, this methodological issue to record meaningful and robust data on testing and debugging practices mirrors many studies on CT (Brennan and Resnick, 2012; Lye & Koh, 2014). For example, Kafai et al. (2014) found that students struggled with debugging in a crafts-oriented approach when programming e-textiles, but provided no details to on this process beyond general researcher observations. Similarly, Denner et al. (2012) also did not describe the debugging processes students adopted beyond general observations that students found it challenging. The implications of these findings are they contribute little about how to best teach effective testing and debugging processes for beginners in programming. Consequently, further investigation on how this CT practice can be monitored and recorded in an effective and robust way is needed for educational research.

In addition, syntax errors were reported to be a major struggle for many students in this study, which also reflects some participants' experiences in a study by Kafai et al. (2014). The findings alluded only to a trial and error approach to solve bugs and syntax errors, which was also found to be the case in Lin and Liu's (2012) study of parent-child collaborations with beginner programmers. Lister (2011) emphasises a trial and error approach should be discouraged with novice programmers so they can develop effective abilities in problem solving practices. The implications for educators are that students in this study tended to employ a trial and error approach to solving syntax errors, which ineffectively supported learning outcomes.

The lack of code commenting in all the student code gathered could have contributed to a trial and error approach, which may have partly explained why some students did not show evidence for a 'thinking then doing' approach to programming. However, it is difficult to conclude the extent students may or may not have used a 'thinking then doing' approach because the data gathering strategies were not able to capture non-verbal evidence effectively. No case study from the reviewed literature revealed how code commenting were utilised, which exposes a gap in the literature on

CT. However, Brennan and Resnick (2012) suggest code commenting may be a way to aid students' CT practices (e.g. devising a plan for debugging). The implications for educators reveal students do not comment their code in Sonic Pi if they are not taught this practice as to aim problem solving. Moreover, it also indicates that the student participants were not developing their code commenting abilities, which may limit how CT practices can support learning outcomes with the designed unit of work.

Reuse and remixing. Reuse and remixing effectively facilitated the sharing of skills and knowledge between students for the development of their music compositions. However, similar to data collected for the testing and debugging element, students could not remember how they remixed code, which also brought up issues around individual student authorship and assessment practices raised by Brennan and Resnick (2012). These issues are reinforced by Van Aalst (2013), where individual contributions in group work have been evaluated as challenging to track and assess. However, authentic learning in a professional programming setting includes code that is often shared and remixed (Brennan and Resnick, 2012). Therefore, a disconnect and tension exists between traditional assessment in schools and the new and revised Digital Technologies content (DTC) in the Technology learning area of the New Zealand Curriculum (NZC), which promotes authentic learning (Ministry of Education [MoE], n.d.a). While reuse and remixing compromises the clarity in knowing what learning outcomes are achieved by individual students, the implication of the findings for educators suggest this CT practice benefits learning through allowing students to collaborate and share ideas, which supports learning outcomes (but not the individual assessment of them).

Abstracting and modularising. All students' final music compositions were discovered to abstract short sequences of sounds that were contained in separate modules (or code blocks) predominantly through the use of live loops. The way in which students structured these modules were motivated by the musical blend with other modules within their music compositions. Other case studies on programming at a school level have not reported and analysed details on the way students' code is modularised (Allsop, 2018; Burke, 2012; Denner et al., 2012; Kafai et al., 2014; Lee, 2010), which indicates this CT practice needs further research. Abstracting and modularising highlights a major difference between the modules in Sonic Pi found and how they are typically used in a programming languages designed to build software. For example, in the music composition context in this study, modules were *musically* connected (e.g., a module for chords and another module for the melody), however, these modules were not *programmatically* connected because their code did not call each other to trigger behaviours like in the study by Brennan and Resnick (2012). Because the findings indicated that students were not exposed to Sonic Pi code being programmatically connected, it may explain why the majority of live loops had non-descriptive names. For educators, the implications are that programmatically abstracting and modularising code will likely not be developed with beginners in Sonic Pi to support learning outcomes in programming. However, this CT practice does support music learning outcomes through the musical blending and layering of different sounds and sequences in each module.

Conclusion. The overall implications for educators are that employing all elements of CT practices support learning outcomes in music and programming through the development of music compositions. However, 'testing and debugging' and 'reuse and remixing' had limited qualitative evidence describing the extent to which students employed each CT practice, which mirrored data gathering issues in literature on CT (Brennan & Resnick, 2012; Lye & Koh, 2014). Students' lack of memory yielded little data for each CT practice from the methods and instruments used in this research, which may also reinforce the studies by Flavell (1979) and Hu (2011) on adolescents' metacognition could be limited and still emerging at this age. However, the lack of ability to remember and articulate processes may also be due to the way in which the students were taught and the pedagogical approach used. With more explicit teaching and scaffolding to help communicate computational practices that were not based on a constructionism pedagogy, the student participants may have been more successful to remember and describe their processes. With the consideration of these unresolved issues, the most effective way to teach, monitor and assess CT practices needs further research to understand what is appropriately challenging for beginners at this level. Thus, based on the outlined implications for each CT practice, the following recommendations are made for educators:

- Being incremental and iterative is recommended for the development of music compositions over a series of lessons (rather than a unit of work aimed at live-coding with Sonic Pi).
- A trial and error approach to solving errors and bugs should be discouraged through providing effective alternatives to solving these. Modelling examples from a variety of effective approaches is suggested as a possible way help illuminate these alternative methods for students.
- As recommended by Brennan and Resnick (2012), the unit of work should be modified to teach and emphasise the value of code commenting in order to aid the development of CT practices (particularly testing and debugging).
- Beginners new to text-based programming will likely struggle with syntax errors commonly from the incorrect spelling of Sonic Pi commands. Therefore, educators should emphasise how to quickly overcome these errors through teaching how to read and understand the console error messages as well as effective ways to look up documentation to compare example code.
- Reuse and remixing is recommended as a highly beneficial way to share and build awareness of a range of creative approaches to music composition with Sonic Pi between students. However, educators need to be aware that encouraging this CT strategy will take away the reliability of evidence to show individual understanding of learning outcomes.
- Findings on abstracting and modularising dimension highlighted that students were not developing abilities to programmatically connect their code together to trigger behaviours with Sonic Pi. Therefore, it is recommended that educators to plan to develop this CT practice in another programming platform to support learning outcomes in programming.

5.1.3 CT perspectives. *Expressing.* Brennan and Resnick (2012) suggest that learning is more engaging and effective if students are able to express themselves through their intrinsic interests. This claim is confirmed in the results of this thesis, where it was discovered all students successfully expressed themselves to support learning outcomes in music and programming. An explanation of these findings is likely attributed through the design of the given project brief and lesson activities to develop two music compositions (see Appendix A and B). Additionally, some students also reflected the freedom to express themselves with Sonic Pi was easier and more engaging than their experiences of programming in Python. Previous studies on Sonic Pi for a live-coding performance context also mirror these findings that creating music was an engaging way for students to express themselves while learning about programming (Burnard et al., 2014: Cheng, 2018). These findings also reinforce studies that interest driven projects maximise student expression for middle school students, which are effective for supporting learning outcomes in both a programming context (Peppler & Kafai, 2009) and a music composition context (Leung, 2004). Surprisingly, there were no case studies found in the literature countering the notion that students expressing themselves negatively affect learning outcomes. However, there were three common themes discovered that were barriers for students to fully express themselves in this study:

1. A lack of time to complete music compositions, which confirmed Leung's (2004) warning that creative projects do not necessarily fit well into time constraints;
2. Some students said they would have preferred to learn music composition techniques on music genres they were familiar with (like pop music);
3. A few students expressed that they did not think of themselves as creative.

These barriers highlight tensions between a balance of teaching knowledge and skills while allowing creative freedom and enough time for students to express themselves. Collectively, the implications of these findings suggest that while the unit of work supported learning outcomes through the development of music compositions in Sonic Pi, these barriers impacted the extent to which students were able to express themselves to their full potential. However, it is likely these issues will remain in other contexts in a school environment, which are challenges for the learner and teacher to reflect on regularly.

Connecting. Brennan and Resnick (2012) argue learning is more effective if students learn and share both with and for each other. The findings in this study confirmed connecting in a variety of ways supported learning outcomes, which may be explained through the combined connecting activities in this study (for example: group compositions, class discussions, presenting music compositions in a final presentation, feedback between students, and reuse and remixing). Burnard et al. (2016) and Cheng (2018) further confirm learning outcomes in music and programming were supported through students connecting in similar ways for a live-coding context with Sonic Pi. Furthermore, this notion that ‘connecting’ benefits learning between students is confirmed by Werner and Denning (2009) in a case study on pair programming at middle school, and a group music composition context by Thorpe (2017). However, not all forms of collaboration were universally reported to be beneficial as a few students in this study said they preferred working by themselves. Surprisingly, accounts of students preferring to work individually did not appear in the reviewed literature. This preference for individual versus group learning indicates an important consideration for educators that activities connecting students together do not necessarily support learning outcomes effectively for all students. Therefore, the implications of these findings suggest that educators allow the option for students to work collaboratively in order to align with student preferences for learning.

Questioning. Brennan and Resnick (2012) suggest students should be asking questions that could be answered through computation to increase understanding of themselves and the world around them. While the findings in this study confirmed some students could formulate questions to support learning outcomes in music and programming, there was a lack of student voice in initiating this questioning. This finding implies that students may not have: (a) understood the benefits of effective questioning, and (b) been unsure of what an effective question entails. For example, while Sonic Pi challenged assumptions about the behaviour of programming languages that led to a greater understanding about music and programming, these assumptions were not understood as a result of questioning from the student. However, this finding may also reinforce the studies on adolescents' metacognition raised by Flavell (1979) and Hu (2011) as the development of metacognition are found to have only started at this age (Flavell, 1979). The reviewed literature did not report on this CT perspective, which is surprising particularly in the studies using this framework by Brennan and Resnick (2012) to teach and assess CT (Kafai et al., 2014; Kong et al., 2018). The implications of these findings for educators indicate that it remains unclear the extent beginner students at a Year 8 school level can employ effective questioning from their own voice to support learning outcomes.

Conclusion. Brennan and Resnick (2012) state this CT dimension highlights the shifts in perspectives students have when developing their CT. Overall, the findings for this dimension were mixed. In particular, the expressing and connecting elements were confirmed to be effective for supporting learning outcomes in music and programming through the designed unit of work. A clear trend across the reviewed literature in Chapter 2 suggest other studies in both a music and programming contexts leaves these findings uncontested. However, it was discovered unavoidable barriers and student preferences for learning indicated ways these CT perspectives do not support learning outcomes for some students. Furthermore, the questioning element was discovered to be disappointing to support learning outcomes because there was a lack of a student voice in asking questions, which may further reinforce limitations in metacognitive abilities for young adolescents (Flavell, 1979). While the findings in this dimension largely confirmed the reviewed literature for all elements, the successes and barriers illuminate important considerations for educators on how CT can support learning outcomes. The implications for educators indicate the expressing and connecting CT elements can be effective for supporting learning outcomes through increased student engagement. However, student led questioning needs to be further scaffolded to support learning outcomes for beginner students at this level. Thus, the recommendations for educators are:

- The activities promoting CT in the designed unit of work (See Appendix A) with Sonic Pi are recommended because they potentially enable beginner students at a Year 8 level to effectively express themselves. However, the limiting aspects of student preferences for learning and a lack of time need to be considered as potential barriers to enable the 'expressing' CT element to be fully realised.
- The connecting activities of group compositions, class discussions, presenting music compositions in a final presentation, feedback between students, and reuse and remixing are all recommended to support learning outcomes. However, educators need to be aware that some students prefer working individually.
- It is unclear the extent to which Year 8 students can effectively employ the questioning CT perspective because there may be limitations in metacognitive abilities in adolescents (Flavell, 1979). Nevertheless, it is recommended to design activities that encourage student led questioning through providing examples that beginner students at this level can easily use.

5.1.4 Overall conclusion for the main research question. Brennan and Resnick (2012) suggest CT is about the habits employed by computer scientists and programmers to solve computational problems. For the main research question, the major contribution of this study is that it illuminates many successes and challenges not reported in the literature on Sonic Pi to support learning outcomes in music and programming. Where possible, the findings mostly reinforced the reviewed literature across all CT dimensions and their elements. Additionally, they often expanded or exposed many gaps not in the literature for a more detailed understanding of how CT can support learning outcomes in music and programming. Thus, this study starts a detailed understanding on the ways in which CT can support learning outcomes in music and programming with the Sonic Pi platform, which could enhance the successful integration of the DTC in New Zealand (NZ).

More specifically, the findings particularly illuminated how programming music compositions in Sonic Pi is different to programming in traditional languages designed to build software (e.g. Python and Scratch). For example: most music is time bound, deals with audio output exclusively, and has no inputs required during its execution (Edwards, 2011). In contrast, programming could include: any type of data output, most often has 'inputs' to process, and frequently has some degree of dynamic user interaction (Shute et al., 2017). Despite reporting that learning overlaps occurred between music and programming, these fundamental differences between both disciplines were not highlighted in prior studies on Sonic Pi (Burnard et al., 2014; Burnard et al., 2016; Cheng, 2018). For example, programming concepts like parallelism and sequences in Sonic Pi overlapped in a way that was beneficial for emphasising the importance of silence or time between sounds in music. However, significant challenges were discovered when closely analysing the integration of many CT elements for beginners with Sonic Pi, which have major limitations to develop learning outcomes in programming and music (e.g. there was minimal evidence of students integrating conditions and operators into their projects). Thus, the overall recommendations for educators are:

- Sonic Pi and the designed unit of work in this study are recommended as a creative way to introduce programming for interested beginners at a Year 8 level to support learning outcomes in music and programming – provided educators plan to compensate for the challenges outlined under each element of the three CT dimensions in this Chapter.
- In particular, it is recommended that the challenges of programming in the Sonic Pi platform need to be especially considered for the CT concepts of conditions and operators and the CT practice of abstracting and modularising.
- When teaching with Sonic Pi, emphasise music composition concepts that are similar to CT concepts. For example, sleep commands are required to sequence sounds, which highlights the role of time and space as an essential component of rhythm in music.

Research. This study expands the literature on interdisciplinary CT through Brennan and Resnick's (2012) framework for the Sonic Pi programming environment. Using this framework to code the data using a directed content analysis approach helped to illuminate the strengths, challenges, and gaps associated with the unique aspects of programming in Sonic Pi when compared to traditional programming environments designed to develop software. However, there were limitations in the data gathered in this study to give reliable insight into what occurred in the CT practices dimension. For example, the results for testing and debugging revealed students were successfully solving bugs, but little insight was gained on how students were achieving or not achieving in this CT practice with Lye and Koh's (2014) coding scheme. This gap may confirm literature that students only start to develop their metacognitive abilities in their adolescent years found by Flavell (1979). However, this challenge may also be explained only because students found it difficult to remember what processes and decisions they followed. Thus, it is difficult to draw conclusions on the effectiveness of students' processes in each CT practice. Therefore, this gap in understanding mostly reinforces the challenges in gathering data that illuminate CT practices described by Brennan and Resnick (2012), Lye and Koh (2014) and Allsop (2018). The implication for researchers are that this study contributed little to the literature on the established challenges in gathering data on CT practices, but added to the literature on using Brennan and Resnick's (2012) framework with the Sonic Pi platform to illuminate its unique characteristics. Thus, the recommendations for researchers are:

- The Brennan and Resnick (2012) CT assessment framework is recommended to code data using a directed content analysis approach to illuminate a holistic understanding of strengths, gaps and challenges of CT at a school level.
- Methods and instruments that require recalling from memory the processes and decisions participants make in the CT practices dimensions are not recommended as a data gathering strategy for Year 8 beginners in

programming (see Section 5.4 for recommendations for possible ways to overcome this limitation).

Policy. The findings in this study highlight the potential benefits of an interdisciplinary approach to CT that can support learning outcomes in music and programming. Moreover, the participant music teacher (Zach) was willing and able to start integrating these concepts into his future teaching. Recently, there has been a growing argument that CT has limited efficacy in supporting interdisciplinary learning outcomes in non-computing subjects (Denning, 2017; Tedre & Denning, 2016), suggesting CT should be only taught in Digital Technologies (DT) classes. In this study, it was discovered that those with prior programming experience were more successful in achieving learning outcomes. A possible implication from this finding could be that having specialist learning opportunities greatly compliments interdisciplinary learning (whether in or out of school). However, if the implementation of the DTC is presented as exclusive to computing subjects, there is a danger of missed opportunities like the ones found in this study to enrich the interdisciplinary learning experiences of students developing CT at a school level. Thus, the implications suggest CT may be a way to recognise that computation can be used as an effective skill to be taught and developed for learning in domains other than DT, which potentially supports the successful integration of the DTC in NZ. The recommendations at a policy level are:

- Ensure there are specialist learning opportunities accessible to all students in programming and music
- Ensure professional development opportunities are available for non-DT teachers so that they can take a role in CT teaching and learning to aid the successful integration of the DTC in NZ schools.
- Fund further research into ways CT can (and cannot) effectively overlap and integrate into other non-computing subjects.

- Design and implement education policies that incentivise and enable teachers to collaborate and share knowledge between staff and students for the effective development of interdisciplinary CT.

5.2 Sub-research Question Discussion

The sub-research question aimed to answer: *To what extent can the creative activity of composing music with the Sonic Pi platform help to promote more positive attitudes towards programming?* This question is answered through the subscales of a modified framework by (Teo, 2007), which has the three elements of enjoyment, importance, and self-confidence. First, overall findings across the three subscales are discussed, which is followed by an in-depth discussion of qualitative themes identified for each subscale. This section then concludes with implications and recommendations for the sub-research question.

5.2.1 Results across all attitude subscales. Overall, the results indicated all students' subscales (enjoyment, importance, and self-confidence) for attitude increased significantly in programming from T1 (pre-unit of work) to T2 (post-unit of work). In particular, the importance and self-confidence subscales yielded the largest effect sizes in programming. These large effect sizes are encouraging because they suggest the designed unit of work with the Sonic Pi platform particularly promotes that both programming is an important skill to learn and an increased self-confidence in this skill. Thus, this approach may be an effective way to introduce programming to students at a Year 8 school level to aid the successful integration of the DTC in NZ. The previous studies on Sonic Pi by Burnard et al. (2014) and Cheng (2018) reinforce more generally that Sonic Pi promotes an increased engagement with programming. However, both of these studies did not quantitatively measure attitudes, nor did they record data prior to the unit of work starting to measure potential differences with post unit of work data. Therefore, it is difficult to compare results beyond a general confirmation that both studies found the student participants reacted positively to the unit of work. Nevertheless, the findings in this thesis could be attributed to the design of its unit of work because it enables the creative freedom for students to explore and experiment with code in Sonic Pi, which could be aligned and linked with their intrinsic preferences in music. The T1 and T2 questionnaire results support this idea because there was an increase in student perception on whether they thought programming can be a creative activity. The reviewed literature with case studies using a variety of programming platforms and tasks also unanimously support the notion that students react positively to creative programming tasks (Allsop, 2018; Brennan & Resnick, 2012; Burke, 2012; Denner et al., 2012; Lee, 2010; Kaifai et al, 2014; Kong et al., 2018; Vekiri, 2010). The overall implication of these findings for educators is that the creative activity of music composition with the Sonic Pi and the designed unit of work could be a way to promote more positive attitudes towards programming for other contexts.

However, many studies have discovered that increased exposure to general computer use can also promote more positive attitudes (Barron et al., 2010; Bovée et al., 2007; Cazan et al., 2016; Teo, 2006;2008). The strong correlations found between the music and programming T2 results may also suggest that exposure generally leads to more positive responses in attitude tests. Therefore, it is difficult to specifically attribute the large increases in attitudes found in this study with the creative activity of music composition using Sonic Pi and the design of the unit of work. However, the qualitative evidence was able to support aspects of activities associated with music composition in Sonic Pi and the design of the unit of work with emerging themes under each subscale (to be discussed later in this Chapter).

Moderate correlations were discovered between quiz scores and programming grades for all subscales (except for a weak correlation found between the enjoyment programming subscale and individual student programming grades). These findings suggest a predominantly moderate relationship between programming skill level and attitude. Supporting this notion that a correlation exists between these factors, Başer (2013) found significant correlations between attitude and academic performance in a programming context. Cazan et al. (2016) also discovered a lower self-confidence in general computer use leads to students using computers less as a learning tool, which is likely to decrease academic performance when learning programming skills. Thus, the findings in this thesis confirms this literature that student attitudes are correlated to their level of competence in a skill. The implication of this finding for educators is approaches and factors that relate to promoting more positive attitudes are important to adopt to increase student academic performance.

The participant music teacher's (Zach) attitude subscales either increased or remained the same from T1 to T2. No subscale indicated a negative attitude in T2, which was supported in all qualitative data gathered from Zach. Similar to the results for the student participants, Zach's subscales for programming importance and self-confidence saw the most significant increases. This result is encouraging because his programming self-confidence was initially lacking in T1 due to inexperience. A major finding in the study by Burnard et al. (2014) reports generally that participant teacher attitudes to be positive towards Sonic Pi. Aaron et al. (2016) pushes this further and argues that educational partnerships for learning computer programming offer rich and effective opportunities and experiences for learning with teachers and students. Findings in this thesis suggest the chance to experience and learn programming with Sonic Pi was a re-invigorating experience for Zach, which supports the idea that educational partnerships between teachers in different disciplines can be beneficial for teacher engagement. The implication of these findings indicates a way to help overcome the lack of teacher capability with the implementation of the DTC. This approach can be fostered through inclusive and collaborative opportunities for non-computing specialist teachers to develop CT skills.

5.2.2 Enjoyment. The novelty of making music with Sonic Pi was found to be a common explanation as to why students enjoyed the unit of work. This theme mirrors interviews conducted from the reviewed literature on Sonic Pi by Burnard et al. (2014) and Cheng (2018) with student participants. Sonic Pi is unique in the way it combines two disciplines into one activity, which was a new experience for all student participants in this study. Therefore, the positive results may be also related to the idea that they were learning about a skill they had never experienced before. Other case studies involving creative activities that support student interest also reinforce the notion they are enjoyable for students (Brennan & Resnick, 2012; Davies et al., 2013; Kong et al., 2018; Vekiri, 2010). Thus, the implication for educators indicate that the new and novel creative activity of the Sonic Pi platform and the designed unit of work may be a potential way to increase student enjoyment in other contexts.

Working on music compositions in pairs was a second common theme, which was also reflected as a positive factor in the study by Burnard et al. (2014). This theme relates to the findings for the Connecting CT perspective dimension (see Section 4.1.3). However, two students reported they preferred to work individually, which counters the notion that collaborative activities are supported by all students. Kong et al. (2018) found those student participants who wanted to collaborate more had more positive attitudes than those who wanted to work individually. However, there was not sufficient evidence to confirm this from the data gathered in this thesis. The designed unit of work enabled both collaborative and individual work, which may explain why students did not report their preference for working as they were able to engage in both. Thus, the implication for this theme suggests educators allowing students to choose collaborative or individual work may maximise their enjoyment.

5.2.3 Importance. The only theme emerging from qualitative data coded under the importance subscale indicated that students perceive programming skills important because it helps them prepare for the modern workforce. The DTC is aligned towards this goal (MoE, n.d.c), which may explain one reason why many students see programming in this way. While it is important students are aware of the potential extrinsic benefits of learning programming skills for their future employment, Kong et al. (2018) and Verkiri (2010) found learning activities that are intrinsically motivating was a significant predictor of students' interest in computing. Music composition may be also intrinsically motivating for middle school students (Leung, 2004; Menard, 2013), therefore, the combination of music composition and programming skills with Sonic Pi potentially provides a link for students to become intrinsically interested in programming. However, the lack of themes from the findings in this study that suggest students had intrinsic interest in computer programming is concerning despite the opportunity to engage with both subjects in Sonic Pi. Kay (1991) warned against too much emphasis on promoting computing as a literacy without an appreciation for the craft through relating it to music, saying that "... if teachers do not nourish the romance of learning and expressing, any external mandate for a new 'literacy' becomes as much a crushing burden as being forced to perform Beethoven's sonatas while having no sense of their beauty" (p. 138). Thus, the implications of findings for educators indicate many students may be already aware of the potential employability benefits of programming through the promotion of the DTC. However, students may be less aware of the potential intrinsic benefits, which may be more motivating and rewarding.

5.2.4 Self-confidence. A major theme related to the self-confidence subscale found that many students perceive programming as a difficult skill to learn. All five interviewed students who all received the top two grades and relatively high quiz scores also thought programming as difficult, suggesting this perception exists regardless of the level competence in programming skills. This theme is consistent across the reviewed literature, despite males often reporting more self-confidence than females (Anderson et al., 2008; Margolis & Fisher, 2002). However, the perception that programming is difficult is surprising because the quantitative results for this subscale found a large effect size. Thus, these results are also encouraging because they indicate students increased their self-confidence significantly despite many perceiving programming as a difficult skill to learn.

A few students expressed that they do not see themselves as creative, which may also explain why they reported programming with Sonic Pi as difficult. Glăveanu (2018) and Edwards (2011) warn against idealist and romantic notions of creativity that promote the idea you need to be born with creative abilities. Moreover, learning programming has also been reported in the literature as having a similar stereotype for only those born with this skill (Margolis & Fisher, 2002; Abbiss, 2008). Thus, the implications of findings for educators suggest creative programming activities may decrease self-confidence in some students. However, findings indicated that the creative activity of music composition with

Sonic Pi may not necessarily pose a barrier to promote more self-confidence as this study found significant increases of this subscale for most students.

5.2.5 Conclusion. Overall, the findings indicated the unit of work significantly promoted more positive attitudes towards programming for all participants. These results add to a more detailed and robust understanding from existing literature on how the creative activity of music composition through Sonic Pi can promote more positive attitudes towards programming. The quantitative themes that emerged for each subscale largely supported the reviewed literature and helped to illuminate challenges in each subscale for educators. However, there was a notable lack of critical responses from students in their reflections on all subscales, which may suggest the instruments and methods were not effective in encouraging students to reflect in this way. All interviewed students received the highest grades and relatively high quiz scores for music and programming. Bovée et al. (2007) found that academic performance is significantly related to attitude, which may explain the lack of a critical voice from all the interviewed students. Nevertheless, the major implication is the creative activity of music composition with Sonic Pi can promote more positive attitudes towards programming. Thus, the recommendations for educators are:

- Overall, it is recommended that educators use Sonic Pi and the designed unit of work for those beginner programming students interested in music composition to potentially promote more positive attitudes towards programming.
- *Enjoyment*: It is recommended to allow for both individual and group projects, which may promote students' enjoyment according to their preferences for working.
- *Importance*: While it is recommended to inform students that programming skills can offer bright employment opportunities in the future, it is more important to emphasise potential intrinsic benefits for learning programming skills as an empowering activity to express oneself.

Intrinsic interests in programming may be achieved through interest driven projects like music composition with Sonic Pi.

- *Self-confidence*: It is recommended to emphasise that programming is a skill any beginner student can learn. Sonic Pi is recommended as an accessible way for beginner students at a school level to effectively develop their self-confidence in programming.

Research. The research instruments used for the sub-research question had no apparent issues. Additionally, no obvious challenges were discovered when using the directed content analysis approach with a modified theoretical framework by Teo (2007) for attitude consisting of the subscales: enjoyment, importance, and self-confidence. However, it was surprising to discover there was a notable lack of negative responses across all qualitative data. This mirrored existing studies on Sonic Pi where participants reported positive experiences (Burnard et al., 2014; Cheng, 2018). There may be three main factors that could explain reasons for this:

- All interviewees received top grades and relatively high quiz scores.
- Students were supported more than usual with two teachers throughout the unit of work.
- The novelty of making music with Sonic Pi was a theme as to why this unit of work was exciting for students.

Therefore, the data gathered in this study were likely skewed towards positive findings. This bias indicates the instruments and methods should be modified to gather data from students with a range of programming competency levels. With these modifications, researchers could use the instruments and framework to measure student attitudes at a school level in other contexts. Provided these adjustments have been piloted, the recommendations for researchers are that the methods and instruments can likely be used to successfully measure student attitudes towards programming.

Policy. The positive results for the sub-research question suggest implications for the successful integration of the DTC in NZ. Many students thought programming can be beneficial for their future employability. Thus, the extrinsic aims of the DTC (MoE, n.d.c) can be a successful factor to promote the perception that programming is important. However, it is also concerning that intrinsic perceptions as to why programming is important to students were not reported. The participant music teacher was found to have a more positive attitude towards teaching programming and was willing to include a unit on Sonic Pi in the future as a result of participating in this research. Thus, the recommendations at a policy level are:

- Emphasise intrinsic motivations as to why learning programming is important in the DTC documents.
- Support teachers with professional development opportunities for those willing to develop their skills and deliver interdisciplinary units of work.
- Develop incentives for collaboration between teachers to team teach for sharing of knowledge and skills for interdisciplinary teaching and learning.

5.3 Limitations

Small sample size and single-case study. As this research was framed as a mixed-method single-case study, the data obtained was detail oriented and only provided a single snapshot of the designed unit of work. Therefore, the findings do not claim to be transferable, repeatable and dependable to other educational contexts (Cohen et al., 2011). The selective sampling strategy is likely to have impacted the results of this unit of work to be more positive, because the participant music teacher and I could collaborate effectively having had prior experiences working together. Despite this unknown impact on the results, the contribution of this mixed-method case study is it provided an in-depth description and analysis (Cohen et al., 2011) that may influence, researchers, educators and policy makers for the successful implementation of CT at a school level and the DTC in NZ.

Additionally, without a control group with an impossibly large sample size for statistical significance to compare data with, the extent the unit of work and the Sonic Pi platform were accountable for the results will remain unknown. For example, other studies on student attitudes towards computers have found those who have had more exposure to computers is correlated to academic performance and more positive attitudes (Başer, 2013). Thus, more studies in other contexts with Sonic Pi and the designed unit of work are needed to help understand if similar successes and challenges occur for others.

Lack of case studies at a school level on both research questions. A lack of similar case study research was discovered that had easily comparable results for both research questions. For example, the literature review found few case studies on the Sonic Pi platform (Burnard et al., 2014; Burnard et al., 2016; Cheng, 2018), which only reported on general student engagement and motivation. These studies only enabled shallow comparisons to be made. Furthermore, case studies on other programming languages at a school level could only broadly be compared with the results in this study. The reason for this is that music composition with code in Sonic Pi is different in character (as highlighted in Chapter 4) to the code produced in traditional programming languages' designed to make computer software. Therefore, it is difficult to draw robust and reliable comparisons between studies to understand the significance of the findings.

Participant researcher bias. Cohen et al. (2011) state participant researchers have a high influence over the data gathered in research, which are subject to confirmation bias. This bias has been indicated in previous educational research on the Logo programming language with the involvement of Seymour Papert (Klahr & Carver, 1988). Additionally, the case studies on Sonic Pi involved its creator (Burnard et al., 2014), which is also likely to have influenced its positive findings. Mitigating against this issue according to Cohen et al. (2011) involves post-lesson reflections of my different roles, reflexivity, respondent checks, and checks by external reviewers, all of which have been embedded in the methodology of this thesis (see Chapter 3). Despite these precautionary strategies, the results are still recognised to be prone to confirmation bias for positive findings because I designed the unit of work and was a participant researcher in this study.

Modelling effects as a male teacher. The modelling effects I had as a male teacher were likely to have had less impact for female participants to promote more positive attitudes towards programming (sub-research question). Previous studies on gender differences in learning programming have found that female teachers are more likely to positively influence and engage female students in computing subjects (Margolis & Fisher, 2002). While no apparent and observable differences were found between the male and female participants, the total sample size is statistically too small ($N = 22$) to compare any differences in gender. Therefore, it remains unclear the extent to which females may react differently to Sonic Pi and the designed unit of work.

Accuracy and descriptiveness of qualitative data. As mentioned in Section 5.1, the five selected students that were interviewed all were discovered to have received the top two grades for their music compositions and relatively high quiz score totals. Moderate correlations were found between programming competency and most subscales, which reinforced the literature that attitude correlates to academic performance in a programming context (Başer, 2013). Therefore, the in-depth descriptive interview data is likely to have influenced the findings to be more positive. Additionally, there was a notable lack of negative or critical responses discovered from all participants. As a result, it was recommended in this Chapter for researchers to modify the instruments and data gathering strategies to encourage responses from a range of student abilities.

Participant absences. Fortunately, there were no more than two students absent for any given lesson in this study, and numerical quiz score data were able to be estimated in for these gaps (see Chapter 3). However, qualitative data was left missing as recommended by Cohen et al. (2011). The impact these absences had on my findings were particularly considered when looking at individual student data. As there were relatively few student absences throughout the whole unit of work, the overall impact to the results of these missing cases is considered small.

Questionnaire and quiz results. The questionnaires and quizzes had multiple-choice and Likert style checkbox questions, which are prone to response biases. Particularly for adolescent participants, there is a danger with this style of data gathering that participants simply misinterpret questions and select answers without thinking (Cohen et al., 2011). However, the participant students may also have been limited in their metacognitive abilities found by Flavell (1979), which could have been a reason for not engaging with the questions asked. This limitation in metacognitive abilities was reinforced where students struggled to remember processes and decisions made for the findings under the CT practices dimension. The following strategies were employed to help prevent these issues:

- there were three questions in each subscale measure with one reverse coded so that a mean score for each subscale can be calculated;
- each item in the questionnaire were explained to students before commencing;
- the questionnaire and quizzes were checked for acceptable face and reading level for a Year 8 level from several teachers;
- students were told they could ask for clarification on items at any stage;
- mean scores for each attitude subscale were triangulated with other qualitative data where consistencies and inconsistencies were analysed.

While these strategies may have helped to mitigate against inaccurate data, it is not clear the extent students may have simply selected answers without thinking or misinterpreted the questions. Therefore, further research is needed with similar instruments and participants to compare the results to understand the extend this issue may have affected the data.

5.4 Further Research

This research raises critical questions around the educational value and integration of Sonic Pi in schools and the designed unit of work for the successful integration of the DTC in NZ. If the recommended modifications under each research question and their instruments outlined in this Chapter are effected, researchers can expand on the findings in many ways in further research. Some notable possibilities for further research include:

- In the course of conducting this research, gathering meaningful data was difficult for the CT practices dimension (main research question) and encouraging a critical student voice from a range of programming competency levels (sub-research question). Further research could include real-time observations and recordings from a balanced range of student competency levels in programming, which may sufficiently illuminate both a critical student voice and the way students engage with CT practices. This data gathering strategy has also been recommended in the literature on CT (Brennan & Resnick, 2012; Lye & Koh, 2014), which may help to gather meaningful data for the CT practices dimension and to reveal a critical student voice.
- In this study, it is recommended that Sonic Pi could be effective as a starting point for beginners (provided they are interested in music composition) for their progression to other programming languages. However, it is not clear to what extent CT knowledge and skill transfer can occur given the differences highlighted in this study between programming in Sonic Pi and traditional programming languages designed to develop software. Further research could investigate the extent to which skill and knowledge for novices at a school level transfers from Sonic Pi to traditional programming languages like Python.
- This case study could be repeated in different contexts and with different participants (e.g. all female or male classes, different cultures and backgrounds) to compare the results found in this study. Additionally, different contexts may also include studying teachers with little programming experience who lead the delivery of the unit of work, which will help to understand common successes and challenges teachers new to programming may face.

In NZ, the successful implementation of the DTC will largely depend on educators' understanding of CT (MoE, 2017). However, there is still little case study research on CT and programming attitudes, which is urgently needed at a school level to inform effective ways to help promote more positive attitudes towards programming on an international level. These suggestions for further research will further contribute both to an understanding of effective CT teaching methods and for the successful integration of DTC in NZ.

References

Aaron, S. (2016). Sonic Pi – performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, *12*(2), 171–178. https://doi.org/10.1080/14794713.2016.1227593

Aaron, S., & Blackwell, A. F. (2013). From sonic Pi to overtone: creative musical experiences with domain-specific and functional languages. *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design - FARM '13*, *Boston, Massachusetts, USA,* 35-46. https://doi.org/10.1145/2505341.2505346

Aaron, S., Blackwell, A. F., & Burnard, P. (2016). The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music, Technology and Education*, *9*(1), 75–94. https://doi.org/10.1386/jmte.9.1.75_1

Abbiss, J. (2008). Rethinking the 'problem' of gender and IT schooling: Discourses in literature. *Gender and Education, 20*(2), 153-165. doi:10.1080/09540250701805839

Abbiss, J. (2009). Gendering the ICT curriculum: The paradox of choice. *Computers & Education*, *53*(2), 343–354. https://doi.org/10.1016/j.compedu.2009.02.011

Abbiss, J. (2011). Boys and machines: gendered computer identities, regulation and resistance. *Gender and Education*, *23*(5), 601–617. https://doi.org/10.1080/09540253.2010.549108

Allsop, Y. (2018). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30–55. https://doi.org/10.1016/j.ijcci.2018.10.004

Anderson, N. D. (2016). A Call for Computational Thinking in Undergraduate Psychology. *Psychology Learning & Teaching*, *15*(3), 226–234. https://doi.org/10.1177/1475725716659252

Anderson, N., Lankshear, C., Timms, C., & Courtney, L. (2008). 'Because it's boring, irrelevant and I don't like computers': Why high school girls avoid professionally-oriented ICT subjects. *Computers & Education*, *50*(4), 1304–1318. https://doi.org/10.1016/j.compedu.2006.12.003

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads 2(1), 48-54.* http://dx.doi.org/10.1145/1929887.1929905

Barron, B., Walter, S. E., Martin, C. K., & Schatz, C. (2010). Predictors of creative computing participation and profiles of experience in two Silicon Valley middle schools. *Computers & Education*, *54*(1), 178–189. https://doi.org/10.1016/j.compedu.2009.07.017

Başer, M. (2013). Attitude, gender and achievement in computer programming. *Middle-East Journal of Scientific Research, 14*(2)*,* 248-255. doi:10.5829/idosi.mejsr.2013.14.2.2007

Bell, J., & Bell, T. (2018). Integrating computational thinking with a music education context. *Informatics in Education, 17*(2), 151-166. doi:10.15388/infedu.2018.09

Bell, T. (2015). Surprising Computer Science. In: Brodnik, A., & Vahrenhold, J. (eds.), *Informatics in Schools. Curricula, Competences, and Competitions, ISSEP 2015,* 9378, pp. 1–11. https://doi.org/10.1007/978-3-319-25396-1_1

Bell, T., Andreae, P., & Robins, A. (2014). A Case Study of the Introduction of Computer Science in NZ Schools. *ACM Transactions on Computing Education*, *14*(2), 1–31. https://doi.org/10.1145/2602485

Biggs, J. B., & Collis, K. F. (1982). Evaluating the quality of learning: The SOLO taxonomy (structure of the observed learning outcome). *New York: Academic Press.*

Blackwell, A., & Aaron, S. (2015). Craft Practices of Live-coding Language Design. In *Proceedings of the First International Conference on Live-coding*, (pp. 41–52). Leeds, UK: ICSRiM, University of Leeds. http://doi.org/10.5281/zenodo.19318

Blackwell, A. F., Church, L., & Green, T. (2008). The Abstract is an Enemy: Alternative Perspectives to Computational Thinking. Paper presented at the *PPIG 2008 – 20th Annual Workshop*. Retrieved from: http://www.ppig.org/library/paper/abstract-enemy-alternative-perspectives-computational-thinking

Blackwell, A., McLean, A., Noble, J., & Rohrhuber, J. (2014). Collaboration and learning through live-coding. In *Dagstuhl Reports, (Vol. 3, No. 9). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.* (pp. 131-168). https://doi.org/10.4230/dagrep.3.9.130

Bolton, J. (2008). Technologically mediated composition learning: Josh's story. *British Journal of Music Education*, *25*(1), 41–55. https://doi.org/10.1017/S0265051707007711

Bovée, C., Voogt, J., & Meelissen, M. (2007). Computer attitudes of primary and secondary students in South Africa. *Computers in Human Behavior*, *23*(4), 1762–1776. https://doi.org/10.1016/j.chb.2005.10.004

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (Vol. 1, p. 25). Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Brown, C., Czerniewicz, L., & Noakes, T. (2016). Online content creation: looking at students' social media practices through a Connected Learning lens. *Learning, Media and Technology*, *41*(1), 140–159. https://doi.org/10.1080/17439884.2015.1107097

Brown, Q. (2016). Broadening. *ACM Inroads, 7*(4), 46-48. doi:10.1145/3008664

Buolamwini, J. A. (2017). *Gender shades: intersectional phenotypic and demographic evaluation of face datasets and gender classifiers* (Doctoral dissertation, Massachusetts Institute of Technology, USA). Retrieved from https://dspace.mit.edu/handle/1721.1/114068

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency* (pp. 77-91). Retrieved from http://proceedings.mlr.press/v81/buolamwini18a.html?mod=article_inline

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as- writing in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121–135. Retrieved from https://eric.ed.gov/?id=EJ985683

Burnard, P., Brown, N., Florack, F., Major, L., Lavicza, Z., & Blackwell, A. (2014). *Sonic Pi: Live & Coding*. Retrieved from https://static1.squarespace.com/static/5433e132e4b0bc91614894be/t/5465e778e4b02ea3469103b0/1415964536482/research_report_dc_02.pdf

Burnard, P., Lavicza, Z., & Philbin, C. A. (2016). Strictly Coding: Connecting Mathematics and Music through Digital Making. In *Proceedings of Bridges 2016: Mathematics, Music, Art, Architecture, Education, Culture* (pp. 345–350). Phoenix, Arizona: Tessellations Publishing.

Cazan, A.-M., Cocoradă, E., & Maican, C. I. (2016). Computer anxiety and attitudes towards the computer and the internet with Romanian high-school and university students. *Computers in Human Behavior*, *55*, 258–267. https://doi.org/10.1016/j.chb.2015.09.001

Chan, C. C., Tsui, M. S., Chan, M. Y. C., & Hong, J. H. (2002). Applying the Structure of the Observed Learning Outcomes (SOLO) Taxonomy on Student's Learning Outcomes: An empirical study. *Assessment & Evaluation in Higher Education*, *27*(6), 511–527. https://doi.org/10.1080/0260293022000020282

Chomeya, R. (2010). Quality of Psychology Test Between Likert Scale 5 and 6 Points. *Journal of Social Sciences*, *6*(3), 399–403. https://doi.org/10.3844/jssp.2010.399.403

Cohen, L., Manion, L., & Morrison, K. (2011). *Research methods in education* (7th ed). London, New York: Routledge.

Cox, M. T. (2005). Metacognition in computation: A selected research review. *Artificial Intelligence*, *169*(2), 104–141. https://doi.org/10.1016/j.artint.2005.10.009

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed-methods approaches* (Fourth, international student ed.). Los Angeles, Calif: SAGE.

Curzon, P., Bell, T., Waite, J., & Dorling, M. (2019). Computational Thinking. In S. Fincher & A. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (Cambridge Handbooks in Psychology, pp. 513-546). Cambridge: Cambridge University Press. doi:10.1017/9781108654555.018

Davies, D., Jindal-Snape, D., Collier, C., Digby, R., Hay, P., & Howe, A. (2013). Creative learning environments in education—A systematic literature review. *Thinking Skills and Creativity*, *8*, 80–91. https://doi.org/10.1016/j.tsc.2012.07.004

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Denning, P., & Tedre, M. (2019). Computational Thinking. MIT Press Essential Knowledge series, Boston, MA.

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355–369. https://doi.org/10.1007/s40692-017-0090-9

Dowdy, S., Wearden, S., & Chilko, D. (2004). *Statistics for research*. New Jersey, USA: John Wiley & Sons.
Edwards, M. (2011). Algorithmic composition: computational thinking in music. *Communications of the ACM*, *54*(7), 58-67. https://doi.org/10.1145/1965724.1965742
Engelman, S., Magerko, B., McKlin, T., Miller, M., Edwards, D., & Freeman, J. (2017). Creativity in Authentic STEAM Education with EarSketch. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*, 183–188. https://doi.org/10.1145/3017680.3017763
Fishbein, M., & Ajzen, I. (1975). *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research.* MA: Addison-Wesley.
Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, *34*(10), 906.
Frey, C. B., & Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, *114*, 254–280. https://doi.org/10.1016/j.techfore.2016.08.019
Fronza, I., Ioini, N. E., & Corral, L. (2017). Teaching Computational Thinking Using Agile Software Engineering Methods: A Framework for Middle Schools. *ACM Transactions on Computing Education*, *17*(4), 1–28. https://doi.org/10.1145/3055258
Fux, J. J., Mann, A., & Edmunds, J. (1971). *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum* (Rev. ed.). New York: W. W. Norton.
Garcia, M. (2016). Racist in the Machine: The Disturbing Implications of Algorithmic Bias. *World Policy Journal*, *33*(4), 111–117. https://doi.org/10.1215/07402775-3813015
Glăveanu, V. P. (2018). Educating which creativity? *Thinking Skills and Creativity*, *27*, 25–32. https://doi.org/10.1016/j.tsc.2017.11.006
Google Inc. & Gallup Inc. (2016). *Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics*. Retrieved from http://goo.gl/PG34aH
Google. (2016, November 7). *Work at Google — Example Coding/Engineering Interview – YouTube* [Video file]. Retrieved from https://www.youtube.com/watch?v=XKu_SEDAykw&t=0s
Gough-Jones, V. J. (2008). *Girls' perceptions of secondary school specialist computer courses: A case study* (Thesis). University of Canterbury, Christchurch, New Zealand. Retrieved from https://ir.canterbury.ac.nz/bitstream/handle/10092/1749/thesis_fulltext.pdf?sequence=1&isAllowed=y
Grover, G. (2018, November 29). *Thinking About Computational Thinking - Dr. Shuchi Grover Keynote, ICCE 2018* [Video file]. Retrieved 28 December 2018, from https://www.youtube.com/watch?v=Zy5JW5IaMSc
Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Grover, S., Pea, R., & Cooper, S. (2016). Factors Influencing Computer Science Learning in Middle School. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*, 552–557. https://doi.org/10.1145/2839509.2844564

Guzdial, M. (2003). A media computation course for non-majors. *Paper presented at the 8th annual conference on Innovation and technology in computer science education – ITiCSE '03*, 104-108. doi:10.1145/961511.961542

Harel, I., & Papert, S. (1990). Software Design as a Learning Environment. *Interactive Learning Environments*, *1*(1), 1–32. https://doi.org/10.1080/1049482900010102

Herr, K. & Anderson, G. L. (2015). *The Action Research Dissertation: A Guide for Students and Faculty* (second ed.). New York, NY: Sage Publications Inc.

Hipkins, C. (2017, December 8). *New digital technologies for schools and kura* [Press release]. Retrieved from https://www.beehive.govt.nz/release/new-digital-technologies-schools-and-kura

Howell, E. (2016, November 10). *Harvard's 'Computers': The Women Who Measured the Stars. Space*. Retrieved from https://www.space.com/34675-harvard-computers.html

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education - ITiCSE '11*, 223-227. https://doi.org/10.1145/1999747.1999811

Kafai, Y. B., & Burke, Q. (2013). The social turn in K-12 programming: moving from computational thinking to computational participation. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education - SIGCSE '13*, 603-608. https://doi.org/10.1145/2445196.2445373

Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014). A Crafts-Oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. *ACM Transactions on Computing Education*, *14*(1), 1–20. https://doi.org/10.1145/2576874

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583-596. Retrieved from https://www.academia.edu/26126123/A_Framework_for_Computational_Thinking_Based_on_a_Systematic_Research_Review

Kallia, M. (2017). *Assessment in Computer Science courses: A Literature Review*. Retrieved from https://royalsociety.org/-/media/policy/projects/computing-education/assessment-literature-review.pdf

Karaci, A. (2016). Investigation of Attitudes Towards Computer Programming in Terms of Various Variables. *International Journal of Programming Languages and Applications*, *6*(1/2), 1–9. https://doi.org/10.5121/ijpla.2016.6201

Kay, A. (1991). Computers, Networks and Education. *Scientific American, 265*(3), 138-149. Retrieved from http://www.jstor.org/stable/24938722

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, *41*(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*(3), 362–404. https://doi.org/10.1016/0010-0285(88)90004-7

Knezek, G., & Christensen, R. (1998). *Validating the Computer Attitude Questionnaire (CAQ).* Retrieved from https://files.eric.ed.gov/fulltext/ED398243.pdf

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education*, *127*, 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

LaBouliere, J. J., Pelloth, A., Lu, C.-L., & Ng, J. (2015). An exploration of the attitudes of young girls towards the field of computer science. *2015 IEEE Frontiers in Education Conference (FIE)*, 1–6. https://doi.org/10.1109/FIE.2015.7344265

Lee, Y. J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia*, *19*(3), 307-326. Retrieved from https://www.learntechlib.org/primary/p/33300/

Leung, B. W. (2004). A Framework For Undertaking Creative Music-Making Activities In Hong Kong Secondary Schools. *Research Studies in Music Education*, *23*(1), 59–75. https://doi.org/10.1177/1321103X040230010901

Lin, J. M. C., & Liu, S. F. (2012). An investigation into parent-child collaboration in learning computer programming. *Journal of Educational Technology & Society*, *15*(1), 162-173. Retrieved from http://www.jstor.org.ezproxy.canterbury.ac.nz/stable/jeductechsoci.15.1.162

Lister, R. (2011). Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114* (pp. 9-18). Australian Computer Society, Inc. Retrieved from https://dl.acm.org/citation.cfm?id=2459938

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference - ITiCSE-WGR '14*, 1–29. https://doi.org/10.1145/2713609.2713610

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: women in computing*. Cambridge, Massachusetts, USA: MIT Press.

Margolis, J., & Goode, J. (2016). Ten Lessons for Computer Science for All. *ACM Inroads*, *7*(4), 52–56. https://doi.org/10.1145/2988236

Martin Jenkins. (2017). *Digital Technologies and Hangarau Matihiko Consultation Final Repor*. Retrieved from https://education.govt.nz/assets/Documents/dthm/Nov-17-update/Martin-Jenkins-DT-HM-consultation-report.pdf

McKnight, P. E., McKnight, K. M., Sidani, S., & Figueredo, A. J. (2007). *Missing data: A gentle introduction*. Guilford Press.

McPhail, G. (2012). From singular to over-crowded region: Curriculum change in senior secondary school music in New Zealand. *British Journal of Music Education*, *29*(03), 317–330. https://doi.org/10.1017/S0265051712000058

Menard, E. (2013). Creative Thinking in Music: Developing a Model for Meaningful Learning in Middle School General Music. *Music Educators Journal*, *100*(2), 61–67. https://doi.org/10.1177/0027432113500674

Ministry of Education. (2000). *The Arts in the New Zealand Curriculum*. Retrieved from https://nzcurriculum.tki.org.nz/content/download/74037/581665/file/TheArtsCurriculum.pdf

Ministry of Education [MoE]. (2017, October). *Curriculum Advisory Group Report*. Retrieved from https://education.govt.nz/assets/Documents/dthm/Nov-17-update/Digital-Curriculum-Consultation-CAG-Report.pdf

Ministry of Education [MoE]. (n.d.a). *Technology in the New Zealand Curriculum.* Retrieved from http://technology.tki.org.nz/Technology-in-the-NZC

Ministry of Education [MoE]. (n.d.b). *Musical Talk.* Retrieved from http://www.tki.org.nz/r/assessment/exemplars/arts/music/mu_3c_e.html

Ministry of Education [MoE]. (n.d.c). *Digital Technologies and the national curriculum.* Retrieved from http://elearning.tki.org.nz/Teaching/Curriculum-areas/Digital-Technologies-in-the-curriculum

Mohaghegh, M., & McCauley, M. (2016). Computational Thinking: The Skill Set of the 21st Century. International Journal of Computer Science and Information Technologies (IJCSIT), 7(3) ISSN: 0975-9646, pp.1524-1530. Retrieved from https://hdl.handle.net/10652/3422

Moore, D. (2014). Supporting students in music technology higher education to learn computer programming. *Journal of Music, Technology and Education*, *7*(1), 75–92. https://doi.org/10.1386/jmte.7.1.75_1

Mozart, W. A. (Composer). (1793). Musikalisches Würfelspiel, K.516f. Bonn: N. Simrock. Retrieved from https://imslp.org/wiki/Musikalisches_W%C3%BCrfelspiel,_K.516f_(Mozart,_Wolfgang_Amadeus)

Özyurt, H., & Özyurt, Ö. (2015). *A Study For Determining Computer Programming Students' Attitudes Towards Programming And Their Programming Self-Efficacy*. Retrieved from http://acikerisim.lib.comu.edu.tr:8080/xmlui/handle/COMU/1058

Papastergiou, M. (2008). Are Computer Science and Information Technology still masculine fields? High school students' perceptions and career choices. *Computers & Education*, *51*(2), 594–608. https://doi.org/10.1016/j.compedu.2007.06.009

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. Brighton: Harvester Press.

Peppler, K., & Kafai, Y. (2009). *Creative coding: Programming for personal expression.* Retrieved from http://kyliepeppler.com/Docs/2009_Peppler_Creative_Coding_Personal.pdf

Petrie, C. (2018, March 7). *Creative thinking and the new Digital Technologies curriculum [Blog post]*. Retrieved from https://nzareblog.wordpress.com/2018/03/07/creative-thinking-dt/

Polya, G. (1957). *How to Solve It*. New York, USA: Doubleday & Company Inc. Retrieved from https://math.hawaii.edu/home/pdf/putnam/PolyaHowToSolveIt.pdf

Posten, H. O. (1984). Robustness of the Two-Sample T-Test. In D. Rasch & M. L. Tiku (Eds.), *Robustness of Statistical Methods and Nonparametric Statistics* (pp. 92–99). https://doi.org/10.1007/978-94-009-6528-7_23

Resnick, M. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. Boston, USA: MIT Press.

Rubin, M. J. (2013). The effectiveness of live-coding to teach introductory programming. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education - SIGCSE '13*, 651-656. https://doi.org/10.1145/2445196.2445388

Runco, M. A., & Jaeger, G. J. (2012). The Standard Definition of Creativity. *Creativity Research Journal*, *24*(1), 92–96. https://doi.org/10.1080/10400419.2012.650092

Schwartz, D. L., & Bransford, J. D. (1998). A Time For Telling. *Cognition and Instruction*, *16*(4), 475–522. https://doi.org/10.1207/s1532690xci1604_4

Seiter, L. (2015). Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15*, 540–545. https://doi.org/10.1145/2676723.2677244

Selby, C., Dorling, M., & Woollard, J. (2014). *Evidence of assessing computational thinking*. Retrieved from https://eprints.soton.ac.uk/id/eprint/372409

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

United Nations. (n.d.). *Sustainable Development Goals.* Retrieved from https://www.undp.org/content/undp/en/home/sustainable-development-goals.html

Taylor, S. J., Bogdan, R., & DeVault, M. (2016). *Introduction to qualitative research methods: A guidebook and resource*. John Wiley & Sons.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16*, 120–129. https://doi.org/10.1145/2999541.2999542

Teo, T. (2006). Attitudes towards computers: A study of post-secondary students in Singapore. *Interactive Learning Environments*, *14*(1), 17–24. https://doi.org/10.1080/10494820600616406

Teo, T. (2007). Perceived Importance, Enjoyment, and Anxiety as Correlates of Computer Attitudes. *Psychological Reports*, *100*(1), 127–135. https://doi.org/10.2466/pr0.100.1.127-135

Teo, T. (2008). Assessing the computer attitudes of students: An Asian perspective. *Computers in Human Behavior*, *24*(4), 1634–1642. https://doi.org/10.1016/j.chb.2007.06.004

The Royal Society. (2017). *After the Reboot The State of Computing Education in UK Schools and Colleges*. Retrieved from https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf

The Royal Society. (2012). *Shut down or restart The way forward for computing in UK schools.pdf*. (n.d.). Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Thorpe, V. (2017). Assessing complexity. Group composing for a secondary school qualification. *British Journal of Music Education*, *34*(3), 305–320. https://doi.org/10.1017/S0265051717000092

Väkevä, L. (2010). Garage band or GarageBand®? Remixing musical futures. *British Journal of Music Education*, *27*(1), 59-70. https://doi.org/10.1017/S0265051709990209

Van Aalst, J. (2013) Assessment of collaborative learning. In C. Hmelo–Silver, C. A. Chinn, C. K. Chan & A. M. O'Donnell (Eds.), International Handbook of Collaborative Learning. (pp. 280–296). New York: Routledge.

Vekiri, I. (2010). Boys' and girls' ICT beliefs: Do teachers matter? *Computers & Education*, *55*(1), 16–23. https://doi.org/10.1016/j.compedu.2009.11.013

Vivian, R., Lozanovski, C., & Falkner, K. (2017). *Supporting teachers to assess F–10 Digital Technologies Literature review*. Retrieved from https://www.digitaltechnologieshub.edu.au/docs/default-source/default-document-library/esadeliverableassessmentreviewofliterature_publish.pdf?sfvrsn=0

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445–468. https://doi.org/10.1007/s10639-016-9493-x

Webster, P. R. (1990). Creativity as Creative Thinking. *Music Educators Journal*, *76*(9), 22–28. https://doi.org/10.2307/3401073

Weijters, B., Baumgartner, H., & Schillewaert, N. (2013). Reversed item bias: An integrative model. *Psychological Methods*, *18*(3), 320–334. https://doi.org/10.1037/a0032121

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Werner, L., & Denning, J. (2009). Pair Programming in Middle School: What Does It Look Like? *Journal of Research on Technology in Education*, *42*(1), 29–49. https://doi.org/10.1080/15391523.2009.10782540

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine*, 20-23. Retrieved from http://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Wise, S., Greenwood, J., & Davis, N. (2011). Teachers' use of digital technology in secondary music education: illustrations of changing classrooms. *British Journal of Music Education*, *28*(2), 117–134. https://doi.org/10.1017/S0265051711000039

World Economic Forum. (2018). *The Future of Jobs report 2018.pdf*. Retrieved from http://www3.weforum.org/docs/WEF_Future_of_Jobs_2018.pdf

Zimmer, H. (2010). Time. (Music from the Motion Picture Inception). On *the Soundtrack album Inception* [CD]. United States: Reprise.

Sonic Pi Unit Plan

Contents:

Unit plan overview:

Core subject(s)	Music (Composition); Digital Technologies (Programming)	
New Zealand Curriculum Level:	**Computer Science:** NZ Digital Technologies progress outcomes level 4; **Music:** Music–Sound arts achievement standards level 4	
Planned for:	Year 8 music class. Individual, group, and class activities/projects	
New Zealand Curriculum links	**Music** Practical Knowledge (PK): -Music theory focus: Timbre, Texture, Tonality, Rhythm, Layering, Pitch Developing ideas (DI): -Composition Techniques -Individual Composition Communicating and Interpreting (CI): -Pair composition project	**Digital Technologies: Hangarau – Matihiko** Computational Thinking (Brennan & Resnick, 2012): *CT concepts:* sequences, loops, data, parallelism, conditions *CT practices:* being incremental and iterative, testing and debugging, reuse and remixing, abstracting and modularising *CT perspectives:* expressing, connecting, and questioning

	-Writing music for videos with a brief (topical issues facing the world today. see Appendix B) Understanding Context (UC): -film music, algorithmic music, generative music -Listening from Mozart, John Cage, Steve Reich, and generative music by Brian Eno	Designing and Developing Digital Outcomes: -Two algorithmic projects that make music compositions for selected videos on topical issues facing the world today (see Appendix B)
Classroom time required:	Six 1 hour and 40-minute lessons	
Resources needed:	For the teacher: • Projector connected to a computer with high quality speakers loud enough to play music at high volumes and a wide range of frequencies • Many spare headphones for students • Internet connection for each computer • Headphone splitters (one between two students) For each student:	

	• Computer for each student with an internet connection
Notes	→ All lessons will start with a 5-minute recap of concepts covered in previous lessons → <u>For lessons 1-5: 20 minutes before each lesson finishing, students will:</u> a) Save progress (not overwriting previous saved files) and take screenshots of progress b) Answer quiz on concepts covered in Programming and Music, complete short written reflections (5 minutes) c) Participate in class discussion on what they found challenging and successful (5 minutes)

Preparation tasks:

- Confirm that computers are switched on, logged-in, connected to the internet and working with audio (speakers, headphones, headphone splitters) - Ensure students can save their work.	5 to 10 minutes

IN PREPARATION FOR LESSON #6 ONLY: - Ensure each project plays before this class starts; some students might have left bugs in their code that might be embarrassing for them if error messages appear when presenting their work	Over 1 hour depending on the number of students

Lesson #1 Get comfortable:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: What is music composition? What is programming? **15 minutes**	Music: -pitch (high and low) -timbre -synthesiser -repetition Programming:	**Activity #1** play, sleep **Activity #2** use_synth _times.do loop do	Debugging, abstraction, experimenting	Iteration, loops, simple functions, sequence, output	PK, UC	Music: a) All students will compose a melody in Sonic Pi b) All students will explore and experiment with at least three different

2. *Activity 1*: Explore Sonic Pi basic commands: play, sleep, run. **15 minutes** 3. *Activity 2*: Adjusting an algorithm to generate music. **30 minutes** 4. *Activity 3*: Group composition brief. **20 minutes** 5. *Wrap-up Activity*: Quiz and reflection. **10 minutes**	-sequence -loop -output -debugging	live_loop :myLoop do → end				synthesised sounds using Sonic Pi c) All students will use repetition in a composition Programming: a) All students will make a sequence of at least four steps b) All students will make a loop to abstract repetition in their composition
Hyperlink to Lesson #1 plan PDF						

Lesson #2 Scales:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: What did we cover in the last lesson? **5 minutes** 2. *Activity 1*: Recognise repetition and loops in 'Time' by Hans Zimmer from the film 'Inception'. **15 minutes**	Music: -tonality -scale -pentatonic -octave -melody Programming: -pattern recognition -functions -lists	**Activity #1** using note names e.g. :C4 **Activity #2** sample play [:A, :E, :D, :G] notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4, :E4, :Cs5, :B4, :Fs4, :D5, :Cs5)"	Debugging, iteration, abstraction, making algorithms, patterns	Functions, loops, arguments, data types, (lists), methods, naming conventions	PK, DI, UC	Music: a) All students will recognise basic composition repetition/variation in 'Time' by Hans Zimmer b) All students will layer three sounds on top of one another c) All students will create their own

3. *Activity 2*: Create your own music scale. **30 minutes** 4. *Activity 3*: Listening and reflecting. **25 minutes** 5. *Wrap-up Activity*: Quiz and reflection. **15 minutes**	-naming conventions	using the .tick, .choose and .shuffle methods				scale to find a tonality that suits their chosen video Programming: a) All students will develop strategies to make an algorithm more efficient through using iteration b) All students will create an algorithm for that plays a series of notes (scale) in a loop
Hyperlink to Lesson #2 plan PDF						

Lesson #3 Algorithms in music:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: What did we cover in the last lesson? **5 minutes** 2. *Activity 1*: Introduce brief for individual project. **30 minutes** 3. *Activity 2*: Introduction to generative	Music: -texture -mood -sampling -stretching layering cuttoff/highpass Programming: -random number generator -selection -arguments	**Activity #1** rate cutoff **Activity #2** rrand() .choose use_random_seed if else	Decomposition, conditional logic, making algorithms	Methods, selection, arguments, parameters	PK, DI, UC	Music: a) All students will identify and experiment with characteristics of algorithmic music and mood/timbre b) All students will be introduced to sampling

music with Sonic Pi and student time on individual projects. **30 minutes** 4. *Activity 3*: Pair/group/class reflection on progress so far. **15 minutes** 5. *Wrap-up Activity*: Quiz and reflection. **15 minutes**						and experiment with basic sample manipulation <u>Programming:</u> a) All students will use a random number generator (rrand) within Sonic Pi b) All students will experiment conditional

						logic (if/else) in Sonic Pi c) All students will use arguments to stretch and sculpt audio samples with .rate and .cutoff
Hyperlink to Lesson #3 plan PDF						

Lesson #4 Synthesisers and effects:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: What did we cover in the last lesson? **5 minutes** 2. *Activity 2*: Introduce new Sonic Pi commands (with sine, sawtooth, triangle waves). **30 minutes**	Music: -reverb, delay, distortion -attack, release sustain, decay -sine, sawtooth, triangle waves Programming: -decomposition (making synthesisers)	**Activity #1** (none) **Activity #2** with_fx: -reverb -echo -synth: (sine, square, saw, tri) -attack -release -sustain -decay **Activity #3 and #4** (none)	Abstraction, debugging, decomposition	Methods, parameters	PK, DI	Music: a) All students will create their own unique sounds using the square, triangle, and sine raw wave forms b) All students will use basic effects e.g. reverb, delay, cutoff

3. *Activity 3*: Reflection of progress so far and class discussion of selected listening examples. **10 minutes** 4. *Activity 4*: Student time for group projects. **20 minutes** 5. *Wrap-up Activity*: Quiz and reflection. **15 minutes**	-abstraction -methods					Programming: a) All students will create and use their own personalised synthesiser b) All students will explore attack, release and sustain of synthesising a unique and personalised electronic instrument
Hyperlink to Lesson #4 plan PDF						

Lesson #5 Finalising each project:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: What did we cover in the last lesson? **5 minutes** 2. *Activity 1*: Class discussion on examples of film music about topical issues. **10 minutes** 3. *Activity 2*: Recap of Sonic Pi	Music: -using 6 music elements (pitch, texture etc) to give feedback on each other's composition Programming: -recap of any gaps	(none)	Abstraction, making algorithms, debugging, efficiency/optimisation	As per lessons 1-4, debugging	DI	Music and programming: a) All students will optimise, refine and finalise musical ideas for both individual and group projects

commands and student time for group compositions. **30 minutes** 4. *Activity 3*: Student time for the development of both projects and final hand in. **30 minutes** 5. *Wrap-up Activity*: Quiz and reflection. **15 minutes**						
Hyperlink to Lesson #5 plan PDF						

Lesson #6 Showcase of all projects:

Suggested learning sequence	Key concepts	Sonic Pi syntax to be taught this lesson	Curriculum Links			Learning Outcomes
			Computational Thinking	Programming	Music (strands)	
1. *Introduction*: Making constructive comments musically and computationally. **10 minutes** 2. *Activity 1*: Class demonstration and comments of all projects with film. **70 minutes**	Music: -constructive feedback -arc of composition -variation/repetition ratio Programming: -efficiency -optimisation -performance constraints (time limit)	(none)	Evaluation/ Reflection	As per lessons 1-4, debugging	DI, CI	Music and programming: a) All students will evaluate their projects for its fitness for purpose with their chosen video b) All students will have the opportunity to give constructive feedback

3. *Wrap-up Activity*: Quiz and reflection. **15 minutes**						
Hyperlink to Lesson #6 plan PDF						

Assessment plan:

Final Projects	See Appendix B
End of lesson quizzes and reflections	See Appendix C

Appendix B: Brief and assessment rubric for group and individual projects

Brief and Rubric for Group and Individual Projects

The purpose of your project is to write music suitable to the mood of a supplied short video of about 1 - 2 minutes in length both in pairs and individually. These videos have the following specific themes on current issues facing the world today: climate change, plastic in the oceans, and the refugee crisis. Your aim is to create what you think might be an appropriate musical background for your chosen film for both projects. There are no wrong answers and you are encouraged to creatively explore and experiment with different combinations of sounds. Please see the assessment rubric below for specific requirements for both final projects.

Assessing rubric for both projects

Subject	Pre-structural	Uni-structural	Multi-structural	Relational	Extended abstract
Programming (DT)	Substantially lacks programming constructs presented in lessons Code has bugs that prevent it from running	Minor flaws in basic brief requirements Minimal and basic use of programming concepts Program runs but could have bugs that prevent some parts from running A functioning synthesiser has been created but with little evidence of refinement and exploration	A valid solution according to the brief given Some code redundancy Program runs bug free or has bugs that only very minimally affect the output Uses sequences and loops Uses samples and effects A synthesiser has been created which changes a fundamental waveform in some way	Removed most code redundancy and has a clear logical structure Advanced use of sequences, loops, a random number generator, and a variety of Sonic Pi's commands to manipulate sound (including audio effects) Program is bug free and runs as expected according to brief requirements A synthesiser has been created which demonstrates creative sculpting of waveforms with attack, release, sustain and decay	*As per Relational with:* Skillfully uses constructs and concepts beyond those required in the exercise to provide an improved solution. Efficient and creative use of algorithms Obvious demonstration of creative exploration, refinement, and optimisation in final outcome

Music (DI and CI strand only)	Incomplete or no audio No engagement with the topic of the chosen film	Sounds and instrumentation used do not blend well Obvious unintentional tonal clashing/unrelated sounds Rhythms used lack variety, clarity, and clash between sounds Timbres demonstrate little creative exploration and experimentation Tonality demonstrate little creative exploration and experimentation	Some obvious engagement with the subject of the video chosen Sounds and instrumentation are used simply with some manipulation The sounds chosen blend Some rhythmic variety and demonstration of creative exploration Personalised tonality used (created a scale as per lesson #2) A range of compositional techniques used such as repetition, layering, and silence/space Thoughtful use of frequencies No sound is too overbearing to the overall mix	Effective and a highly creative response to the film chosen Sound/instrumentation chosen blend well A range of rhythms, frequencies, timbres and tonal colours demonstrate effective exploration and refinement Effective range of composition techniques used Sounds are mixed to a high standard	*As per Relational with:* Convincing, imaginative, and refined response to the video chosen The sounds chosen all complement each other Imaginative variety of musical elements used: for example, rhythms, frequencies, timbres, and tonal colours

Appendix C: End of class quizzes and reflections

Reflection Diaries for Lessons 1-5:

1. What did you struggle with the most today? If something didn't work, how did you get it working?
2. How do you think your piece could be further developed? How do you think you would go about this?
3. What did you like most about today? Was there anything you disliked or were surprised about?

Reflection Diary for Lesson 6 only:

1. What parts or aspects of your projects did you tinker with the most?
2. What did you like/dislike most about the whole unit of work? What were you most surprised about?

Lesson #1 Quiz items [correct answers are highlighted]

Instructions for all quizzes:

- *You can test code in Sonic Pi and look up documentation to find the right solution*
- *Circle only one letter for each question*
- *You are allowed to ask the teacher to help understand the question*
- *You are **not** allowed to communicate to other classmates during this quiz*

Programming questions:

1. **Which code block below repeats forever in Sonic Pi?**

(a)
```
repeat do
  play 60
  sleep 1
end
```

(b)
```
loop do
  play 60
  sleep 1
end
```
← (b) is the correct answer.

(c)
```
do again
  play 60
  sleep 1
end
```

2. How many syntax errors are in the following code? You can test each line in Sonic Pi to help find them.

```
Play 50
sleep 1
pley 64
slep 1
```

(a) 0
(b) 1
(c) 2
(d) 3

3. Which description is the most accurate of the following Sonic Pi code?

```
loop do
  play 60
  sleep 1
  play 56
  sleep 1
end
```

(a) Play the notes 60 and 56
(b) Forever loop two notes with one second sleep
(c) Play the notes 56 then 60 with one second in-between
(d) Forever loop the notes 60 then 56 with one second sleep between each note

4. This code is intended to play the first part of a well-known song. However, it has a wrong note and therefore has a bug:

```
play 70
sleep 0.5
play 68
sleep 0.5
play 67
sleep 0.5
play 68
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds; then test and debug the code to make it sound as intended.

Which code block of the following options sounds with the intended result?

(a)

```
play 70
sleep 0.5
play 68
sleep 0.5
play 65
sleep 0.5
play 68
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
```

(b)

```
play 70
sleep 0.5
play 68
sleep 0.5
play 66
sleep 0.5
play 68
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
```

← (b) is the correct answer

(c)

```
play 70
sleep 0.5
play 68
sleep 0.5
play 66
sleep 0.5
play 68
sleep 0.5
play 71
sleep 0.5
play 71
sleep 0.5
play 71
sleep 0.5
```

(d)

```
play 70
sleep 0.5
play 68
sleep 0.5
play 66
sleep 0.5
play 69
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
play 70
sleep 0.5
```

5. This code is intended to use the ‘dsaw’ synthesizer and forever loop the notes 60 and 63 with 0.2 seconds in-between each of these notes. However, it does not sound as intended and therefore has one or a few bug(s):

```
live_loop :myLoop do
  use_synth :dsaw
  play 60
  sleep 0.2
  play 63
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds.

Which code block of the following options sounds with the intended result?

(a)
```
live_loop :myLoop do
  use_synth :dsaw
  play 60
  play 63
end
```

(b)
```
live_loop :myLoop do
  use_synth :dsaw
  play 60
  play 63
  sleep 0.2
end
```

(c)
```
live_loop :myLoop do
  use_synth :dsaw
  play 60
  sleep 0.2
  play 63
  sleep 0.2
end
```
← (c) is the correct answer

(d)
```
live_loop :myLoop do
  use_synth :dsaw
  play 60
  sleep 0.2
  sleep 0.2
  play 63
end
```

6. **What order should this code be in to make a loop that plays a note three times with 1 second of sleep after each note? Note that there may be one or two lines of code that are not needed.**

```
1 sleep 1
2 play :a4
3 3.times do
4 live_loop :drums do
5 end
```

(a) 4, 3, 1, 2, 5
(b) 3, 2, 1, 5
(c) 2, 4, 1
(d) 3, 2, 1

Music questions:

1. Click here and listen to the linked audio extract. Which if the following options best describes this sound in music terms?

a) A melody
b) A chord
c) Drums
d) Both melody and chords together

2. Click here and listen to the linked audio extract. Which if the following options best describes this compositional device in music terms?

a) This audio is not a composition technique
b) Repetition
c) Sequence

3. Click here and listen to the linked audio extract. Which if the following options best describes the difference between these two sounds in music terms?

a) The first sound is soft and the second loud
b) Both sounds are roughly the same volume
c) There is no sound
d) The first sound is loud and the second is soft

4. Click here and listen to the linked audio extract. Which if the following options is the instrument that plays?

a) Drums
b) Piano
c) Guitar
d) Flute

Lesson #2 Quiz items [correct answers are highlighted]

Instructions for all quizzes:

- *You can test code in Sonic Pi and look up documentation to find the right solution*
- *Circle only one letter for each question*
- *You are allowed to ask the teacher to help understand the question*

- *You are **not** allowed to communicate to other classmates during this quiz*

Programming questions:

1. **Which code block below allows you to store notes in a variable with the name 'notes' in Sonic Pi?**

(a)
```
notes = :E4, :Fs4, :B4
```

(b)
```
notes = [:E4, :Fs4, :B4]
```
← (b) is the correct answer.

(c)
```
notes :E4, :Fs4, :B4
```

2. **How many syntax errors are in the following code block?**

```
notes = :F3, :Fs4, :C3, :E2]

loop do
  play notes.choose
  sleep 1
End
```

(a) 0
(b) 1
(c) 2
(d) 3

3. **Which description below is the most descriptive of the following Sonic Pi code?**

```
notes = [:F3, :Fs4, :C3, :E2]

live_loop :melody do
  use_synth :dsaw
  play notes.choose
  sleep 1
end
```

(a) With the 'dsaw' synthesiser, play a random element from the 'notes' variable in a loop with 1 second sleep in-between each iteration
(b) Play notes stored in the 'notes' variable with the 'dsaw' synthesiser
(c) Loop the 'melody' code block

(d) Use the ‘dsaw’ synth and loop with one second between each iteration

4. The code below is intended to iterate and play through each note stored in the ‘notes’ variable in the ‘melody’ live loop. However, it does not sound as intended and therefore has a bug:

```
notes = (ring :Fs4, :D5, :Cs5)

live_loop :melody do
  play notes
  sleep 0.33
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds; then test and debug the code to make it sound as intended.

Which code block of the following options sounds with the intended result?

(a)

```
notes = (ring :Fs4, :D5, :Cs5)

live_loop :melody do
  play notes.choose
  sleep 0.33
end
```

(b)

```
notes = (ring :Fs4, :D5, :Cs5)

live_loop :melody do
  play notes
  sleep 0.33
  play notes
  sleep 0.33
  play notes
  sleep 0.33
end
```

(c)

```
live_loop :melody do
  notes = (ring :Fs4, :D5, :Cs5)
  play notes
  sleep 0.33
end
```

(d)

```
notes = (ring :Fs4, :D5, :Cs5)

live_loop :melody do
  play notes.tick
  sleep 0.33
end
```

← (d) is the correct answer

5. The code below is intended to randomly choose notes from the variable called ‘notes’ in the ‘repeatedCs’ live loop. However, it does not sound as intended and therefore has one or a few bug(s):

```
notes = (ring :C3, :C5, :C4)

live_loop :repeatedCs do
  play notes.tick
  sleep 0.2
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds.

Which code block of the following options sounds with the intended result?

(a)

```
notes = (ring :C3, :C5, :C4)

live_loop :repeatedCs do
  play notes.choose
  sleep 0.2
end
```

← (a) is the correct answer

(b)

```
live_loop :repeatedCs do
  play notes.choose
  notes = (ring :C3, :C5, :C4)
  sleep 0.2
end
```

(c)

```
notes = (ring :C3, :C5, :C4)

live_loop :repeatedCs do
  play notes
  sleep 0.2
end
```

(d)

```
live_loop :repeatedCs do
  play notes.choose
  sleep 0.2
end

notes = (ring :C3, :C5, :C4)
```

6. **What order should this code be in to make a loop that of a cymbal repeating every two seconds and plays forever? There may be one or two lines of code that are not needed.**

```
1 sleep 2
2 sample :drum_cybal_open
3 end
4 2.times do
5 live_loop :cymbal do
```

(a) 4, 3, 2, 5
(b) 3, 2, 1
(c) 5, 2, 1, 3
(d) 2, 1, 5, 4

Music questions:

1. Click here and listen to the linked audio extract. Which if the following options best describes this sound in music terms?
a) A melody
b) A chord
c) Drums
d) Both melody and chords together

2. Click here and listen to the linked audio extract. Which if the following options best describes this compositional device in music terms?
a) This audio is not a composition technique
b) Layering sounds
c) Sequence

3. Click here and listen to the linked audio extract. Which if the following options best describes the difference between these two sounds in music terms?
a) The first sound is low and the second is high in pitch
b) The first sound is high and the second is low in pitch
c) Both sounds are roughly the same pitch
d) They are both different percussion instruments

4. Click here and listen to the linked audio extract. Which if the following options is the instrument that plays?
a) Drums
b) Piano
c) Guitar
d) Flute

Lesson #3 Quiz items [correct answers are highlighted]

Instructions for all quizzes:

- *You can test code in Sonic Pi and look up documentation to find the right solution*
- *Circle only one letter for each question*
- *You are allowed to ask the teacher to help understand the question*
- *You are **not** allowed to communicate to other classmates during this quiz*

Programming questions:

1. **Which code block below allows you to randomly select a sleep time between 0.5 and 1 seconds in Sonic Pi?**

(a)
```
sleep random .5 to 1
```

(b) This is not possible in Sonic Pi

(c)
```
sleep rrand(0.5, 1)
```
← (c) is the correct answer.

2. **How many syntax errors are in the following code block?**

```
live_loop :piano do
  play rrand(60 65)
  sleep rrand(0.3, 0.7)
end
```

(a) 0
(b) 1
(c) 2
(d) 3

3. **Which description below is the most descriptive of the following Sonic Pi code?**

```
1 loop do
2   if one_in(2)
3     play [:A, :E, :D, :G].choose
4     sleep 1
5   else
6     play chord(:E4, :minor)
7     sleep 0.7
8   end
9 end
```

(a) Loop lines 1-9 containing a condition (if/else). Lines 2-4 is executed if one is calculated in a random selection between one and two; line 3 randomly chooses one note from a list on this line, then a one second of sleep.

(b) Loop lines 1-9. Line 3 randomly chooses one note from a list on this line, then a one second of sleep. Lines 6-7 plays an E minor chord followed by a sleep time of 0.7.

(c) Lines 2-4 is executed if one is calculated in a random selection between one and two; line 3 randomly chooses one note from a list on this line, then a one second of sleep. However, if '2' is calculated from line 2, then line 6-7 get executed – which plays an E minor chord followed by a sleep time of 0.7.

(d) Loop lines 1-9 containing a condition (if/else). Lines 2-4 is executed if one is calculated in a random selection between one and two; line 3 randomly chooses one note from a list on this line, then a one second of sleep. However, if '2' is calculated from line 2, then line 6-7 get executed – which plays an E minor chord followed by a sleep time of 0.7.

4. The code below is intended to play random notes in a minor pentatonic scale from the note 50 and have a random sleep time between each note between 0.3 and 0.5 in a live loop using the 'dsaw' synthesizer. However, it does not sound as intended and therefore has a bug:

```
live_loop :melody do
  use_synth :saw
  play scale(:minor_pentatonic).choose
  sleep rrand(0.3, 0.5)
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds; then test and debug the code to make it sound as intended.

Which code block of the following options sounds with the intended result?

(a)
```
live_loop :melody do
  play scale(:minor_pentatonic)
  sleep rrand(0.3, 0.5)
end
```

(b)
```
live_loop :melody do
  use_synth :saw
  play scale(50, :minor_pentatonic).choose
  sleep rrand(0.3, 0.5)
end
```
← (b) is the correct answer

(c)
```
live_loop :melody do
  use_synth :saw
  play scale(50, :minor_pentatonic).choose
  sleep 0.3
end
```

5. The code below is intended to randomly choose a branch of a condition (if/else); each branch is intended to play the same collection of notes but with different synthesizers. However, it does not sound as intended and therefore has one or a few bug(s):

```
loop do
  if
    use_synth :beep
    play [:A, :E, :D, :G].choose
    sleep 0.5
  else
    use_synth :dpulse
    play [:A, :E, :D, :G].choose
    sleep 0.5
  end
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds.

Which code block of the following options sounds with the intended result?

(a)
```
loop do
  use_synth :beep
  if
    play [:A, :E, :D, :G].choose
    sleep 0.5
  else
    play [:A, :E, :D, :G].choose
    use_synth :dpulse
    sleep 0.5
  end
end
```

(b)

```
loop do
  if one_in(2)
    use_synth :beep
    play [:A, :E, :D, :G].choose
    sleep 0.5
  else
    use_synth :dpulse
    play [:A, :E, :D, :G].choose
    sleep 0.5
  end
end
```

← (b) is the correct answer

(c)

```
loop do
  use_synth :beep
  if
    play [:A, :E, :D, :G].choose
    sleep 0.5
  else
    use_synth :dpulse
    sleep 0.5
  end
end
```

(d)

```
loop do
  if
    play [:A, :E, :D, :G].choose
    sleep 0.5
  else
    play [:A, :E, :D, :G].choose
    sleep 0.5
  end
end
```

6. **What order should this code below be in to make a loop repeat 8 times and plays random notes between 50 and 95 with a sleep time of 2 seconds between each iteration? There may be one or two lines of code that are not needed.**

```
1 sleep 2
2 play rrand_i(50, 95)
3 end
4 2.times do
5 8.times do
```

(a) 4, 3, 2, 1
(b) 3, 2, 4
(c) 5, 2, 1, 3
(d) 2, 3, 5, 4

Music questions:

1. Click here and listen to the linked audio extract. Which if the following options best describes this sound in music terms?
a) A melody
b) A chord
c) Drums
d) Melody and chords together

2. Click here and listen to the linked audio extract. Which if the following options best describes this compositional device in music terms?
a) Base line
b) Arpeggiation
c) Drone

3. Click here and listen to the linked audio extract. Which if the following options best describes the difference between these two sounds in music terms?
a) The first sound is shorter than the second
b) The first sound is longer than the second
c) Both sounds are the same length

4. Click here and listen to the linked audio extract. Which if the following options is the instrument that plays?
a) Violin
b) Clarinet
c) Synthesiser
d) Saxophone

Lesson #4 Quiz items [correct answers are highlighted]

Instructions for all quizzes:

- *You can test code in Sonic Pi and look up documentation to find the right solution*
- *Circle only one letter for each question*
- *You are allowed to ask the teacher to help understand the question*
- *You are **not** allowed to communicate to other classmates during this quiz*

Programming questions:

1. **Which code block below correctly plays the note :C4 and manipulates the attack, release, sustain, and decay of a saw synthesiser in Sonic Pi?**

(a)
```
synth :saw, note: :C4, release: 3, sustain: 0.7, decay: 1
```

(b) This is not possible in Sonic Pi

(c)
```
synth :saw, attack: 2, release: 3, sustain: 0.7, decay: 1
```

(d)
```
synth :saw, note: :C4, attack: 2, release: 3, sustain: 0.7, decay: 1
```
← (d) is the correct answer.

2. **How many syntax errors are in the following code block?**

```
Synth :tri, attack: 2 release: 3, sustain 0.7, decay: 1
```

(a) 0
(b) 1
(c) 2
(d) 3

3. **Which description below is the most descriptive of the following Sonic Pi code?**

```
live_loop :synth do
  synth :fm, note: :d4, release: 8, cutoff: 100, amp: 0.7, attack: 1
  sleep 1
end
```

(a) In a live loop called ‘synth’, play an ‘fm’ synth with the note d4; on each iteration of the loop, have one second sleep
(b) In a live loop called ‘synth’, play an ‘fm’ synth that manipulates the release, cutoff, amplitude and attack.
(c) In a live loop called ‘synth’, play an ‘fm’ synth with the note d4 that manipulates the release, cutoff, amplitude and attack; and on each iteration of the loop, have one second sleep

(d) Play the note d4 that manipulates the release, cutoff, amplitude and attack; on each iteration of the live loop, have one second sleep

4. The code below is intended to play the 'guit-hamronics' sample in a loop with a random amplitude and sleep time on each iteration; it is also intended to use the audio effect 'bitcrusher'. However, it does not sound as intended and therefore has a bug:

```
loop do
  sample :guit_harmonics, amp: rrand(0.4, 2.4)
  with_fx :bitcrusher do
    sleep rrand(1, 3)
  end
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds; then test and debug the code to make it sound as intended.

Which code block of the following options sounds with the intended result?

(a)

```
loop do, with_fx :bitcrusher do
  sample :guit_harmonics, amp: rrand(0.4, 2.4)
  sleep rrand(1, 3)
end
```

(b)

```
loop do
  with_fx :bitcrusher do
  sample :guit_harmonics, amp: rrand(0.4, 2.4)
  sleep rrand(1, 3)
end
```

(c)

```
with_fx :bitcrusher do
  loop do
    sample :guit_harmonics, amp: rrand(0.4, 2.4)
    sleep rrand(1, 3)
  end
end
```

← (c) is the correct answer

5. The code below is intended to use the variable ‘r’ to randomly manipulate the ‘rate’ of which the sample on line 3 is played. However, it does not sound as intended and therefore has one or a few bug(s):

```
1 loop do
2   r = rrand(0.2, 3)
3   sample :ambi_piano
4   sleep 0.1
5 end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds.

Which code block of the following options sounds with the intended result?

(a)

```
loop do
  r = rrand(0.2, 3)
  sample :ambi_piano, amp: r
  sleep 0.1
end
```

(b)

```
r = rrand(0.2, 3)

loop do
  sample :ambi_piano, r
  sleep 0.1
end
```

(c)

```
loop do
  r = rrand(0.2, 3)
  sample :ambi_piano, rate: r
  sleep 0.1
end
```

← (c) is the correct answer

(d)

```
loop do
  use r
  r = rrand(0.2, 3)
  sample :ambi_piano, r
  sleep 0.1
end
```

6. **What order should this code below be in to play the note 60 with the echo audio effect and one second sleep? There may be one or two lines of code that are not needed.**

```
1 sleep 1
2 with_fx :echo do
3 play 60
4 end
5 8.times do
```

(a) 5, 2, 1, 3
(b) 2, 3, 1, 5
(c) 2, 1, 5, 4, 3
(d) 2, 3, 1, 4

Music questions:

1. Click here and listen to the linked audio extract. Which if the following options best describes this sound in music terms?

a) A melody
b) A chord
c) Drums/percussion
d) Both melody and chords together

2. Click here and listen to the linked audio extract. Which if the following options best describes this compositional device in music terms?

a) Base line
b) Arpeggiation
c) Drone

3. Click here and listen to the linked audio extract. Which if the following options best describes the difference between these two sounds in music terms?

a) The first sound is brighter than the second
b) The first sound is darker than the second
c) Both sounds are have roughly the same tonal colours

4. Click here and listen to the linked audio extract. Which if the following options is the instrument that plays?

a) Bass
b) Piano
c) Synthesiser
d) Flute

Lesson #5 Quiz items [correct answers are highlighted]

Instructions for all quizzes:

- *You can test code in Sonic Pi and look up documentation to find the right solution*
- *Circle only one letter for each question*
- *You are allowed to ask the teacher to help understand the question*
- *You are **not** allowed to communicate to other classmates during this quiz*

Programming questions:

1. **Which code block below correctly plays the 1st element of the notes variable in Sonic Pi?**

(a)
```
play notes[0], attack: 1, release: 2
notes = [:Fs4, :G4, :B4]
```

(b) This is not possible in Sonic Pi

(c)
```
notes = [:Fs4, :G4, :B4]
play notes[0], attack: 1, release: 2
```

(d)
```
notes = [:Fs4, :G4, :B4]
play notes[1], attack: 1, release: 2
```
← (d) is the correct answer.

2. **How many syntax errors are in the following code block?**

```
c = (chord_degree, :i, :c4, :minor
play C
```

(a) 0
(b) 1

(c) 2
(d) 3

3. Which description below is the most descriptive of the following Sonic Pi code?

```
loop do
  use_synth :dpulse
  play scale(:C4, :aeolian).choose
  sleep rrand(0.5, 0.7)
end
```

(a) Play random notes from the C aeolian scale in the 4th octave.
(b) In a loop, play random notes from the C aeolian scale in the 4th octave with the ‘dpulse’ synthesiser and a random sleep time between 0.5 and 0.7 seconds on each iteration.
(c) In a loop, play random notes from the C scale with a random sleep time between 0.5 and 0.7 seconds.
(d) Play random notes from the C aeolian scale in the 4th octave with a random sleep time between 0.5 and 0.7 seconds.

4. The code below is intended to play the ‘ambi_choir’ sample in a loop with the rate, attack and decay manipulated with a 1 second sleep on each iteration. However, it does not sound as intended and therefore has a bug:

```
live_loop :lowNote do
  sample :ambi_choir, amp: 0.1, attack: 22
  sleep 1
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds; then test and debug the code to make it sound as intended.

Which code block of the following options sounds with the intended result?

(a)
```
live_loop :lowNote do
  sample :ambi_choir, rate: 0.5
  sleep 1
end
```

(b)

```
live_loop :lowNote do
  sleep 1
end

sample :ambi_choir, rate: 0.5
```

(c)

```
live_loop :lowNote do
  sample :ambi_choir, rate: 0.5, attack: 2, decay: 4
  sleep 1
end
```

← (c) is the correct answer

5. The code below is intended to randomly play the notes C4, Db4, and G3 with the 'tri' synth in a loop with a random sleep time; it is also intended to systematically use the numerical values stored in the 'r' variable to manipulate the attack and sustain of each iteration through the '.tick' method. However, it does not sound as intended and therefore has one or a few bug(s):

```
r = [1, 2, 3, 4].tick
synth :tri, attack: r, sustain: r, note: [:c4, :db4, :g3]

live_loop :phase do
  sleep rrand(0.2, 5)
end
```

→ You can copy and paste this code into Sonic Pi to hear how it currently sounds.

Which code block of the following options sounds with the intended result?

(a)

```
r = [1, 2, 3, 4].tick

live_loop :phase do
  synth :tri, attack: 1, sustain: 1, note: [:c4, :db4, :g3].choose
  sleep rrand(0.2, 5)
end
```

(b)

```
r = [1, 2, 3, 4]

live_loop :phase do
  synth :tri, attack: r, sustain: r, note: [:c4, :db4, :g3].choose
  sleep rrand(0.2, 5)
end
```

(c)

```
r = [1, 2, 3, 4].tick

live_loop :phase do
  synth :tri, attack: r, sustain: r, note: [:c4, :db4, :g3]
  sleep rrand(0.2, 5)
end
```

(d)

```
r = [1, 2, 3, 4].tick

live_loop :phase do
  synth :tri, attack: r, sustain: r, note: [:c4, :db4, :g3].choose
  sleep rrand(0.2, 5)
end
```

← (d) is the correct answer

6. **What order should this code below be in to play the note 49 with the synth called 'beep' and repeat forever with 0.5 seconds rest in-between each iteration? There may be one or two lines of code that are not needed.**

```
1 synth :beep, note: 49
2 live_loop :beep do
3 sleep 0.5
4 end
5 10.times do
```

(a) 4, 3, 2, 5
(b) 2, 1, 3, 4
(c) 2, 1, 5, 4, 3
(d) 2, 3, 1, 4

Music questions:

1. Click here and listen to the linked audio extract. Which if the following options best describes this sound in music terms?
   a) A melody
   b) A chord
   c) Drums
   d) Both melody and chords together

2. Click here and listen to the linked audio extract. Which if the following options best describes this compositional device in music terms?
   a) Arpeggiation

b) Sequence
c) Drone

3. Click here and listen to the linked audio extract. Which if the following options best describes the difference between these two sounds in music terms?
a) The first sound fades out and the second sound fades in
b) The first sound fades in and the second sound fades out
c) Both sounds stay at the same volume throughout

4. Click here and listen to the linked audio extract. Which if the following options is the instrument that plays?
a) Drums
b) Synthesiser
c) Guitar
d) Flute

Appendix D: Pre and post questionnaires for students

Pre questionnaire for students:

Information and instructions:

All names will be kept confidential through using pseudonyms in all publications relating to the study. The school's name will also not be revealed in any publications. Chris Petrie (the researcher) and the music teacher will be the only people with access to the matched list of names to pseudonyms.

This study is voluntary. If you do not wish to continue at any time, please quietly inform a teacher.

This questionnaire asks background information on your experiences in music and programming (also known as coding), and questions relating to your attitude to music and programming.

Once you click 'submit' at the end of this questionnaire, your responses are recorded. This will indicate that you agree to your responses to be used for the purposes of this study.

Please make sure you answer ALL questions as truthfully as possible.

Student Pre-unit of work Questionnaire Part 1

1. Please write your full name: ____________________________

2. Have you previously learnt any programming (or coding) skills? If so, for how long? (Please circle one of the following)

   None | A few weeks | A month | A few months | More than 6 months | Over a year

3. If you have had some experience, what programming languages did you use?

   ____

   ____

4. If you have had some experiences programming, did you try to complete any projects? If so, briefly describe these below.

5. Do you play a musical instrument? If you do: what do you play? For how long?

6. Have you made your own music before? (Please circle one of the following)

YES | NO

7. Have you made your own music on computers before? (Please circle one of the following)

YES | NO

Student Pre-unit of work Questionnaire Part 2

8. Briefly describe your feelings towards learning to program (or code).

9. Briefly describe your feelings towards learning to make your own music.

10. Do you think programming (or coding) can be creative? (Please circle one of the following)

YES | NO

Student Pre-unit of work Questionnaire Part 2

1. I think programming (or coding) is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

2. I think making music is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

3. I think programming (or coding) is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

4. I think making music is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

5. I am excited to learn the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

6. I am excited to learn how to make music. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

7. I think learning programming (or coding) skills are relevant for my future. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

8. I think learning music composition skills is relevant for my future. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

9. I can't see the point in learning the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

10. I can't see the point in learning the skill of making music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

11. I might want to study programming (or coding) in senior high school or university. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

12. I might want to study music composition in senior high school or university. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

13. I find (or might find) programming (or coding) easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

14. I find (or might find) making my own music easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

15. Programming (or coding) can only be learnt by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

16. Making or composing music can only be learnt by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

17. I feel I know what kinds of activities are involved with programming (or coding). (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

18. I feel I know what kinds of activities are involved with making (or composing) music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

Please read this before submitting!

Please check over your answers to ensure you have answered ALL questions correctly.

I understand that by clicking on the submit button that I agree to participate in the study and that my information will be used for research purposes and will be kept anonymous.

Thank you!

Post questionnaire for students:

Information and instructions:

All names will be kept confidential through using pseudonyms in all publications relating to the study. The school's name will also not be revealed in any publications. Chris Petrie (the researcher) and the music teacher will be the only people with access to the matched list of names to pseudonyms.

This study is voluntary. If you do not wish to continue at any time, please quietly inform a teacher.

This questionnaire asks background information on your experiences in music and programming (also known as coding), and questions relating to your attitude to music and programming.

Once you click 'submit' at the end of this questionnaire, your responses are recorded. This will indicate that you agree to your responses to be used for the purposes of this study.

Please make sure you answer ALL questions as truthfully as possible.

Student Post-unit of work Questionnaire Part 1

1. Please write your full name: ______________________________

2. Briefly describe your feelings towards learning to program (or code).

   ____

   ____

3. Briefly describe your feelings towards learning to make your own music.

   ____

   ____

4. Do you think programming (or coding) can be creative? (Please circle one of the following)

   YES | NO

Student Post-unit of work Questionnaire Part 2

1. I think programming (or coding) is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

2. I think making music is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

3. I think programming (or coding) is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

4. I think making music is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

5. I am excited to learn the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

6. I am excited to learn how to make music. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

7. I think learning programming (or coding) skills are relevant for my future. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

8. I think learning music composition skills is relevant for my future. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

9. I can’t see the point in learning the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

10. I can’t see the point in learning the skill of making music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

11. I might want to study programming (or coding) in senior high school or university. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

12. I might want to study music composition in senior high school or university. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

13. I find (or might find) programming (or coding) easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

14. I find (or might find) making my own music easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

15. Programming (or coding) can only be learnt by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

16. Making or composing music can only be learnt by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

17. I feel I know what kinds of activities are involved with programming (or coding). (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

18. I feel I know what kinds of activities are involved with making (or composing) music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

Please read this before submitting!

Please check over your answers to ensure you have answered ALL questions correctly.

I understand that by clicking on the submit button that I agree to participate in the study and that my information will be used for research purposes and will be kept anonymous.

Thank you!

Appendix E: Pre and post questionnaires for the participant music teacher

Pre-questionnaire for the participant music teacher:

Information and instructions:

All names will be kept confidential through using pseudonyms in all publications relating to the study. The school's name will also not be revealed in any publications. Chris Petrie (the researcher) and the music teacher will be the only people with access to the matched list of names to pseudonyms.

This study is voluntary. If you do not wish to continue at any time, please quietly inform a teacher.

This questionnaire asks background information on your experiences in music and programming (also known as coding), and questions relating to your attitude to music and programming.

Once you click 'submit' at the end of this questionnaire, your responses are recorded. This will indicate that you agree to your responses to be used for the purposes of this study.

Please make sure you answer ALL questions as truthfully as possible.

Music Teacher Pre-unit of work Questionnaire Part 1

1. Please write your full name: ___________________________

2. Have you previously learnt any programming (or coding) skills? If so, for how long? (Please circle one of the following)

   None | A few weeks | A month | A few months | More than 6 months | Over a year

3. If you have had some programming experience, what programming languages did you use?

   ____

   ____

4. If you have had some experiences programming, did you try to complete any projects? If so, briefly describe these below.

5. Do you play a musical instrument? If you do: what do you play? For how long?

6. Have you taught music composition before? (Please circle one of the following)

   YES | NO

7. Have you taught music production on computers before? (Please circle one of the following)

   YES | NO

Music Teacher Pre-unit of work Questionnaire Part 2

1. Briefly describe your feelings towards teaching programming (or coding)?

2. Briefly describe your feelings towards teaching music composition?

3. Do you think programming (or coding) can be creative? (Please circle one of the following)

YES | NO

Music Teacher Pre-unit of work Questionnaire Part 2

1. I think teaching programming (or coding) is a lot of fun. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

2. I think teaching music composition is a lot of fun. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

3. I think teaching programming (or coding) is not enjoyable. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

4. I think teaching music composition is not enjoyable. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

5. I am excited to learn about the skill of programming (or coding) so I can teach it. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

6. I am excited to learn about music composition so I can teach it. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

7. I think learning programming (or coding) skills are relevant for my future in teaching. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

8. I think learning music composition skills is relevant for my future in teaching. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

9. I can't see the point in teaching the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

10. I can't see the point in teaching the skill of making music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

11. I might want to learn about programming (or coding) to teach this skill in the future. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

12. I might want to learn about music composition to teach it in the future. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

13. I find (or might find) teaching programming (or coding) easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

14. I find (or might find) teaching making music easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

15. Programming (or coding) can only be learnt and taught by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

16. Making or composing music can only be learnt and taught by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

17. I feel I know what kinds of activities are involved with programming (or coding). (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

18. I feel I know what kinds of activities are involved with making (or composing) music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

Please read this before submitting!

Please check over your answers to ensure you have answered ALL questions correctly.

I understand that by clicking on the submit button that I agree to participate in the study and that my information will be used for research purposes and will be kept anonymous.

Thank you!

Post-unit of work questionnaire for the participant music teacher:

Information and instructions:

All names will be kept confidential through using pseudonyms in all publications relating to the study. The school's name will also not be revealed in any publications. Chris Petrie (the researcher) and the music teacher will be the only people with access to the matched list of names to pseudonyms.

This study is voluntary. If you do not wish to continue at any time, please quietly inform a teacher.

This questionnaire asks background information on your experiences in music and programming (also known as coding), and questions relating to your attitude to music and programming.

Once you click 'submit' at the end of this questionnaire, your responses are recorded. This will indicate that you agree to your responses to be used for the purposes of this study.

Please make sure you answer ALL questions as truthfully as possible.

Music Teacher Post-unit of work Questionnaire Part 1

1. Briefly describe your feelings towards teaching programming (or coding).

   ____

   ____

2. Briefly describe your feelings towards teaching music composition.

   ____

   ____

3. Do you think programming (or coding) can be creative? (Please circle one of the following)

   YES | NO

Music Teacher Post-unit of work Questionnaire Part 2

1. I think teaching programming (or coding) is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

2. I think teaching music composition is a lot of fun. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

3. I think teaching programming (or coding) is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

4. I think teaching music composition is not enjoyable. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

5. I am excited to learn about the skill of programming (or coding) so I can teach it. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

6. I am excited to learn about music composition so I can teach it. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

7. I think learning programming (or coding) skills are relevant for my future in teaching. (Please circle one of the following)

Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

8. I think learning music composition skills is relevant for my future in teaching. (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

9. I can’t see the point in teaching the skill of programming (or coding). (Please circle one of the following)

   Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

10. I can’t see the point in teaching the skill of making music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

11. I might want to learn about programming (or coding) to teach this skill in the future. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

12. I might want to learn about music composition to teach it in the future. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

13. I find (or might find) teaching programming (or coding) easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

14. I find (or might find) teaching making music easy. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

15. Programming (or coding) can only be learnt and taught by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

16. Making or composing music can only be learnt and taught by really intelligent people. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

17. I feel I know what kinds of activities are involved with programming (or coding). (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

18. I feel I know what kinds of activities are involved with making (or composing) music. (Please circle one of the following)

    Strongly disagree | Moderately disagree | Slightly disagree | Slightly agree | Moderately agree | Strongly agree

Please read this before submitting!

Please check over your answers to ensure you have answered ALL questions correctly.

I understand that by clicking on the submit button that I agree to participate in the study and that my information will be used for research purposes and will be kept anonymous.

Thank you!

Appendix F: Student individual project exemplars (Ben, Emma, Grace and Lucas)

Ben's individual project (Extended abstract grade for programming)

```
## A Homeless Bear in London

live_loop :phase1 do
  use_synth :hollow
  with_fx :reverb, mix: 0.9 do
    play choose([:Fs4,:G4]), attack: 4, release: 5
    sleep 10
  end
end

live_loop :phase2 do
  use_synth :hollow
  with_fx :reverb, mix: 0.76 do
    play choose([:D4,:E4]), attack: 6, release: 6
    sleep 8
  end
end

live_loop :phase3 do
  use_synth :hollow
  with_fx :reverb, mix: 1 do
    play choose([:A4, :Cs5]), attack: 5, release: 5
    sleep 11
  end
end

live_loop :phase4 do
  use_synth :hollow
  with_fx :reverb, mix: 0.6 do
    play choose([:F4, :C4]), attack: 3, release: 4
    sleep 15
  end
end

notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4,
             :E4, :Cs5, :B4, :Fs4, :D5, :Cs5)

live_loop :first do
  use_synth :hollow
  play notes.tick, release: 0.5, amp: 0.2, pan: 0.8
  sleep 0.33
end
live_loop :second do
  use_synth :hollow
  play notes.tick, release: 0.7, amp: 0.1, pan: -0.75
  sleep 0.294
end

loop do
  rrand(60, 110)
  if one_in(6)
    use_synth :dark_ambience
    play [:A, :E, :D, :G].choose, pan: 0.4, amp: 0.3
    sleep 3
  else
    use_synth :dark_ambience
    play [:C, :A3, :A5].choose, pan: -0.3, sustain: 0.5, amp: 0.4, decay: 2
    sleep 0.7
  end
end
```

Emma's individual project (Relational grade for programming)

```
live_loop :rad do
  use_synth :prophet
  with_fx :reverb, mix: 0.3 do
    play choose([:C3,:D3, :A3, :G5]), attack: 5, release: 5, sustain: rrand(0.1, 3), decay: rrand(0.1, 3)
    sleep 2
  end
end

sleep 5

with_fx :reverb, phase: 0.5, decay: 8 do
  live_loop :th do
    yep = (scale :e4, :aeolian, num_octaves: 3).choose
    play yep, attack: rrand(1, 4), amp: rrand(0.4, 0.8)
    sleep rrand(3, 12)
  end
end

sleep 5

live_loop :low do
  sample :mehackit_phone3, rate: 0.05, beat_stretch: 1, attack: 2, decay: 4, amp: 2
  sleep 1
end

live_loop :chords do
  cho = (chord_degree, [:i, :v].ring.choose, :C4, :major, 3)
  play chord_invert(cho, [-1, 0, 1].choose)
  sleep rrand(0.5, 2)
end
```

Grace's individual project (Multi-structural abstract grade for programming)

```
live_loop :greg do
  synth :beep, note: [:c3, :c4, :g3].choose
  sleep 0.2
end

with_fx :slicer do
  live_loop :sample do
    sample :ambi_glass_rub, rate: 0.2, amp: 0.5
    sleep 1
  end
end

live_loop :daniel do
  synth :dsaw, note: [:c6, :c5, :g6].choose, amp: 0.1, attack: rrand(1, 3)
  sleep 0.2
end

live_loop :ggg do
  play 60
  sleep 3
  play 53
  play 55
  play 66
  sleep 5
  sleep 3
  play 53
  play 55
end

live_loop :dm do
  sample :bd_808
  sleep 0.4
  sample :drum_cowbell
  sleep 0.4
end

loop do
  synth :noise
  sleep 2
  synth :supersaw
  sleep 4
end
```

Lucas's individual project (Uni-structural grade for programming)

```
## my piece

loop do
  synth :mod_dsaw, sustain: 2, decay: 6, attack: 2
  sleep 4
end

use_synth :prophet
live_loop :not do
  play choose([:d3, :g4]), attack: 2, sustain: 3
  sleep 4
end

loop do
  synth :fm
  sleep 2
end

sample "/Users/user3/Desktop/samples/99Sounds Drones/Atmospheric/Abandoned Turbine.wav"
sleep 3

loop do
  play 22.444
  sleep 2
  play 53

  play 55
end

3.times do
  use_synth :pluck
  play rrand_i(44, 60)
  sleep 0.12
end
```

Appendix G: Information sheets and permission forms to participants

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Information Sheet for the Participant School and Music Teacher

Hello. My name is Chris Petrie and I am a Master of Education thesis student at Canterbury University. You are receiving this information sheet and attached consent form because [school name] has been identified as a potential participant school in a case study on an introduction to interdisciplinary learning with music and programming. I have designed a six-lesson unit of work to be completed aligned with the new Digital Technologies Curriculum the New Zealand Government is formally phasing in for all schools and years in 2018, which will be made compulsory in 2020. I'm an experienced music and computer science teacher with over 10 years teaching in schools in New Zealand, Germany, and Nepal. Recently, I have been the computer science teacher at Saint Kentigern College from Years 10 to 13. A small description of the research, what is involved, and what will be expected is given in this information sheet. Please do not hesitate to contact me using my email (cpe20@uclive.ac.nz) if you have any concerns, issues, or queries about this research at any time.

The newly developed unit of work would include six one hour and 40-minute lessons (total class time will be 10 hours spread over three weeks),and is designed to engage a Year 8 class of students using a software platform called Sonic Pi (http://sonic-pi.net). This is to be timetabled with the school's administration and the music teacher. This research project aims to examine the effectiveness of this unit on students' skill development and attitudes towards programming. Student participants will complete two music compositions (one in pairs, and one individually) using the Sonic Pi platform to one-minute films on current issues in the world today like pollution and climate change. The participant music teacher will be involved in professional development to help your school with the integration of the new Digital Technology Curriculum, and help facilitate these classes.

It is intended all students in the Year 8 class (selected by the music teacher, Board of Trusties and Principal) will be taught the unit of work. Only when all three consent forms have been signed (first school, second parent, and third the student themselves) to take part in this research, will a student within the class be able to participate.

The unit is designed for a minimum of 14 students with help from one of the school's music teacher. Participation in this research will require participant students to take part in questionnaires both before and after the unit of work, two digital projects, in class quizzes, and student reflective diaries. These will be completed during the taught lessons (so no extra time will be needed). Non-participants will do every activity participants will do, except in-class quizzes and student diaries. While the participants are completing these activities, non-participants will be given this time to work more on their individual final projects for this unit with the music teacher. This extra time is relatively short (as the in-class quizzes and student diaries are planned to take up a maximum of 10 minutes per lesson), so the advantage non-participants receive in getting more time to improve their compositions will be relatively small. While participants are completing the 10 minute questionnaires both before and after the unit of work, non-participants will be involved in a listening activity with the music teacher.

Through the consent form for parents and caregivers, I will be asking permission for interviews from five selected student participants in class [class name] both before (5-10 minutes additional time) and after (10-15 minutes additional time) the unit of work. The purposes of these student interviews are to help investigate the aforementioned research aims from a student voice. The selection of these five students will be based on those who will likely give useful and descriptive answers from the pool of students who have had the required consent forms signed and have indicated they would like to participate in these interviews. Parents or caregivers can opt their child out of these interviews and still participate in this research by ticking the appropriate box through the consent form attached to an information sheet I will provide. There will be an opportunity for the student to check the accuracy of the transcript of each interview for all selected students, which will be arranged with the school after the unit of work has been completed at an interval or lunchtime within school hours.

Please note, a school's music teacher will also need to be engaged in order to help deliver and facilitate the unit of work. They will need to sign the consent form attached to this information sheet. The music teacher will also be asked to complete before and after questionnaires as well as semi-structured interviews. Also note that post-lesson reflections from myself will also be made after each lesson as part of the data gathering process. All observations, interviews, and questionnaires will carefully reflect only on the participants in the research (observations on non-participants in the class will not be recorded).

In this unit of work, you should be aware that there is a potential risk that some of the intended interdisciplinary learning outcomes in either music or programming will not be achieved because the design of the unit of work has not been trialled. As a consequence, there may be further time needed beyond this research project in order to fill in gaps. This is unlikely to be the case because the unit is designed for beginners and adaptable to the learning needs of the students, while still linking with the New Zealand Curriculum in both domains. The music teacher will need to be aware of this risk, and plan to ensure time is reserved for learning outcomes that may have been missed. Furthermore, there is a social risk where non-participants may be isolated or left out when their activities are different from the participants (of which take up a relatively small amount of time: in-class quizzes, student diaries, and questionnaires). What then will happen (where activities are different for participants and non-participants in this research) has been outlined above.

Participation in data gathering for this research is voluntary, and the school's Board of Trusties, Principal, participant music teacher, parents or caregivers, and students have the right to have the data recorded about them returned or destroyed before the 1st of June 2018 (when analysis of raw data starts) without penalty. Please also note the final date for withdrawal from this study is the 1st of June 2018.

The results of the project will be published, but you may be assured confidentiality of data gathered in this investigation, as any participant's identity will not be made public. All identities will be kept hidden through using pseudonyms in all publications relating to the study. The name of the school will also be substituted with a pseudonym in any published documents relating to this research. All material collected is securely stored for five years and destroyed after this time in accordance with the University of Canterbury's Educational Research Human Ethics Committee policy. All data of participants will be kept digitally and secure by myself with a password coded cloud storage system and backed up on my personal laptop computer (also password protected). All participant data will be stored using pseudonyms, with no real names associated with this data. Even during the analysis and reporting of results, pseudonyms will also be used for all participants and the participating school so that privacy and confidentiality can be further ensured. Myself and the participant music teacher will be the only people with access to the matched list of names and codes. Please be aware that a thesis is a public document and will be available through the UC Library.

Please indicate to the researcher on the consent form attached if you would like to receive a copy of the summary results for this project.

The project is being carried out as a requirement to complete a thesis for a Master's in Education by Chris Petrie under the supervision of Niki Davis (main supervisor, until approximately April 2018 where Cheryl Brown will take over as the main supervisor) and Valerie Sotardi (co-supervisor), who can be contacted at Niki Davis (niki.davis@canterbury.ac.nz); Valerie Sotardi (valerie.sotardi@canterbury.ac.nz); Cheryl Brown (cheryl.brown@canterbury.ac.nz). They will be pleased to discuss any concerns you may have about participation in this research.

This project has been reviewed and approved by the University of Canterbury Educational Research Human Ethics Committee, and participants should address any complaints to The Chair, Educational Research Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz).

If you agree for [school name] to participate in the study, you are asked to complete the consent form (attached) and have it returned to Chris Petrie.

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Consent Form for the Participant School and Music Teacher

- ☐ I have been given a full explanation of this project and have had the opportunity to ask questions.
- ☐ I understand what is required for this unit of work.
- ☐ I understand that participation is voluntary and may stop the research before the 1st of June 2018. This will also include destroying of any information collected.
- ☐ I understand that any information or opinions given in the raw data will be kept confidential to the researcher and the music teacher [name here]. I understand that any published or reported results will not identify the participants or school. I understand that a thesis is a public document and will be available through the UC Library.
- ☐ I understand that all data collected for the study will be kept in locked and secure facilities and/or in password protected electronic form. I understand that all data will be destroyed after five years.
- ☐ I understand the risks associated with taking part and how they will be managed.
- ☐ I understand that I can contact the researcher (Chris Petrie, email: cpe20@uclive.ac.nz) or supervisors (Niki Davis niki.davis@canterbury.ac.nz; Valerie Sotardi valerie.sotardi@canterbury.ac.nz; Cheryl Brown cheryl.brown@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Educational Research Human Ethics Committee, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz)
- ☐ I would like a summary of the results for this research.
- ☐ I give permission to allow two audio-recorded semi-structured interviews (of the participant music teacher and participant students in the selected Year 8 class) on questions relating to the aims of this research before (for 15 minutes) and/or after (for 20 minutes) the taught lessons. I understand that the participant music teacher [name here] and students will have an opportunity to check the transcript of all interviews.
- ☐ By signing below, I agree for [school name, class name, and participant music teacher's name] to participate in this research project.

Name:____________________Position:______________________

Signed:____________________Date:___________________________

Email address *(for report of findings, if applicable)*:

Please return this form to Chris Petrie.

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Information Sheet for Parents and Caregivers

Hello. My name is Chris Petrie and I am a Master of Education thesis student at Canterbury University. You are receiving this information sheet and attached consent form because your child's class [class name] has been identified, with approval from [school name], as a potential participant in a research project on music composition and programming. I'm an experienced music and computer science teacher with over 10 years teaching in schools in New Zealand, Germany, and Nepal. Recently, I have been the computer science teacher at Saint Kentigern College of Years 10 to 13. A small description of the research, what is involved, and what will be expected of your child is given in this information sheet. Please do not hesitate to contact me using my email (cpe20@uclive.ac.nz) if you have any concerns, issues, or queries about this research at any time.

The newly developed unit of work would include six one hour and 40-minute lessons (total class time will be 10 hours spread over three weeks), and is designed to engage students using a software platform called Sonic Pi (http://sonic-pi.net. This research project aims to examine the effectiveness of this unit on students' skill development and attitudes towards programming.

All students in [class name] will be taught the unit of work as [name of school and participant music teacher] have agreed to and planned this into the school's timetable. Only those students where both themselves and their parents or caregiver have signed consent forms will data be collected for this proposed research. If you choose to give permission for your child to take part in this research, their involvement in this project requires participation in data gathering within their regular class-time. This is timetabled with the school's administration and the music teacher from 09/05/2018 to 25/05/2018 on each Wednesday and Friday from 8:50am to 10:30am. Note that your child's consent will also be formally asked if consent has been approved from you as parent or caregiver.

Participation in this research will require your child and other student participants to take part in questionnaires both before and after the unit of work, two digital projects, in class quizzes, and student reflective diaries. These will be completed during the taught lessons (so no extra time will be needed). Non-participants will do every activity participants will do, except in-class quizzes and student diaries. While the participants are completing these activities, non-participants will be given this time to work more on their individual final projects for this unit with the music teacher. This extra time is relatively short (as the in-class quizzes and student diaries are planned to take up a maximum of 10 minutes per lesson), so the advantage non-participants get in getting more time to improve their compositions will be relatively small. While participants are completing the 10 minute questionnaires both before and after the unit of work, non-participants will be involved in an activity with the music teacher [name here].

Through the consent form (attached to this information sheet), I will be asking permission for interviews from five selected student participants in class [class name] both before (5-10 minutes additional time) and after (10-15 minutes additional time) the unit of work. The purposes of these student interviews are to help investigate the aforementioned research aims from a student voice. The selection of these five students will be based on those who will likely give useful and descriptive answers from the pool of students who have had the required consent forms signed, and have indicated they would like to participate in these interviews. You can opt your child out of these interviews and still participate in this research by ticking the appropriate box through the consent form attached to this letter. There will be an opportunity for your child to check the accuracy of the transcript of the interview for all selected students, which will be arranged with the school after the unit of work has been completed at an interval or lunchtime within school hours.

Please note, the school's music teacher [name here] will also be engaged with this research and help to deliver and facilitate the unit of work. The music teacher [name here] will also be asked to complete before and after questionnaires and semi-structured interviews. Post-lesson reflections from myself will also be made after each lesson as part of the data gathering process. Please note that all observations, interviews, and questionnaires will carefully reflect only on participants (observations on non-participants in the class will not be recorded).

In this unit of work, you should be aware that there is a potential risk that some of the intended interdisciplinary learning outcomes in either music or programming are not achieved because the design of the unit of work has not been trialled. As a consequence, there may be further time needed beyond this research project in order to fill in gaps. This is unlikely to be the case because the unit is designed for beginners and adaptable to the learning needs of the students, while still linking with the New Zealand Curriculum in both domains. The music teacher [name here] is aware of this risk and has planned to ensure time is reserved for learning outcomes that may have been missed. Furthermore, there is a social risk where non-participants may be isolated or left out when their activities are different from the participants (of which take up a relatively small amount of time: in-class quizzes, student diaries, and questionnaires). What then will happen (where activities are different for participants and non-participants in this research) has been outlined above.

Participation in data gathering for this research is voluntary, and your child has the right to have their data returned or destroyed before the 1st of June 2018 without penalty. Please be aware that once analysis of raw data starts on June 1st 2018, it will not be possible to remove data from the analysis.

The results of the project will be published, but you may be assured confidentiality of data gathered in this investigation, as your child's identity will not be made public. All identities will be kept hidden through using pseudonyms in all publications relating to the study. The name of the school will also be substituted with a pseudonym in any published documents relating to this research. All material collected is securely stored for five years and destroyed after this time in accordance with the University of Canterbury's Educational Research Human Ethics Committee policy. All data of participants will be kept digitally and secure by myself with a password coded cloud storage system and a back-up on my personal laptop computer (also password protected). All participant data will be stored using pseudonyms, with no real names associated with this data. Even during the analysis and reporting of results, pseudonyms will also be used for all participants and the participating school. Myself and the participant music teacher will be the only people with access to the matched list of names and codes. Please be aware that a thesis is a public document and will be available through the UC Library.

Please indicate to the researcher on the consent form attached if you would like to receive a copy of the summary results for this project, which will also be given to [school name] management.

The project is being carried out as a requirement to complete a thesis for a Master's in Education by Chris Petrie under the supervision of Niki Davis (main supervisor, until approximately April 2018 where Cheryl Brown will take over as the main supervisor) and Valerie Sotardi (co-supervisor), who can be contacted at Niki Davis (niki.davis@canterbury.ac.nz); Valerie Sotardi (valerie.sotardi@canterbury.ac.nz); Cheryl Brown (cheryl.brown@canterbury.ac.nz). They will be pleased to discuss any concerns you may have about participation in this research.

This project has been reviewed and approved by the University of Canterbury Educational Research Human Ethics Committee, and participants should address any complaints to The Chair, Educational Research Human Ethics Committee, University of Canterbury, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz).

If you agree for your child to participate in the study, you are asked to complete the consent form (attached) and have it returned to the school's administration.

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Consent Form for Parents and Caregivers

- ☐ I have been given a full explanation of this project and have had the opportunity to ask questions.
- ☐ I understand what is required of my child if I agree to for them to take part in the research as legal guardian.
- ☐ I understand that participation is voluntary, and I may request to remove my child's raw data from being included in this study before June 1st 2018.
- ☐ I understand that any information or opinions my child provides will be kept confidential to the researcher and the music teacher [name here], and that any published or reported results will not identify any student or their school. I understand that a thesis is a public document and will be available through the UC Library.
- ☐ I understand that all data collected for this study will be kept in locked and secure facilities and/or in password protected electronic form. I understand that all data will be destroyed after five years.
- ☐ I understand the risks associated with participation and how they will be managed.
- ☐ I understand that I can contact the researcher (Chris Petrie, email: cpe20@uclive.ac.nz) or supervisors (Niki Davis, niki.davis@canterbury.ac.nz; Valerie Sotardi valerie.sotardi@canterbury.ac.nz; Cheryl Brown cheryl.brown@canterbury.ac.nz) for further information. If I have any complaints, I can contact the Chair of the University of Canterbury Educational Research Human Ethics Committee, Private Bag 4800, Christchurch (human-ethics@canterbury.ac.nz)
- ☐ I would like a summary of the results for this research.
- ☐ I give permission for my child to be selected for an audio-recorded semi-structured interview on topics related to the aims of this research before (5-10 minutes) and/or after the unit of work (10-15 minutes). I understand that there will be an opportunity for my child to check the transcript of the interview.
- ☐ By signing below, I agree for my child to participate in this research project.

Name:____________________Signed:________________________Date:

Email address (for report of findings, if applicable):

Please return this form to the administration of [school name here].

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Information Sheet for Participant Students

Hello. My name is Chris Petrie and I am a Master of Education thesis student at Canterbury University. I am proposing to do some research on making music and learning to program (also known as coding).

What will you do?

During your music lessons [music teacher's name] and I are going to be teaching you composition using a cool new computer program (called Sonic Pi) for about three weeks. This is part of your regular music lessons, so most of you will not need to put in extra time out of these classes. However, everyone who has agreed to participate in this research (as well as had their parents or caregivers sign consent forms), will be asked if it is ok to interview you both before these lessons for 5-10 minutes and after for 10-15 minutes. These interviews will ask what you think of programming and music, and to ask questions about your final projects at the end of this unit.

I will also need to know what you think about the lessons and what you have learnt, so I will ask you to help me in my research by also doing some in-class quizzes, keeping a diary, and having a chat to me (if you are happy to). No extra time out of class will be needed to do this (except if you agree to be interviewed).

How much time will it take?

What will be required will take place during your regular music lessons, but you will need to be quite attentive the whole time. There will be six lessons of 1 hour and 40-minutes each. Those who are have agreed to be interviewed and are selected (I will need five students) will need to see me for an extra 15-25 minutes total.

Why is this research important?

The government is requiring that all school children learn to program or code. I think learning to do this while being creative through making music with Sonic Pi could provide another cool way to learn about programming.

What if I want to participate? What if I don't?

If you are reading this your school, teachers and parents have agreed for you to take part in this unit of work. It is simple to be included: (1) read through this information sheet and make sure you understand everything (please ask if you don't!), and (2) read and understand the consent form attached, then sign it if you would like to take part in this research.

If you don't want to participate in this research, do not sign the consent form attached to this information sheet.

What should I do if I have a concern or issue about this research?

Please let the researcher (Chris Petrie) know about your concern (if you feel comfortable), but you can also tell any of your teachers. You can also tell your parent or caregiver, and they can then get in touch with someone responsible at school.

What information will you be recording about us?

You will be involved with: two questionnaires, class quizzes, and writing a diary about your work. I am happy to answer any questions you have about these. As mentioned before, five students who agree to be involved with interviews will need an extra 15-25 minutes from these lessons outside of class time. Those that are interviewed will have the opportunity to check if I heard what they said right by reviewing a transcript (what you have said written down) of the interview after the six lessons (I will arrange a time for this with you and your music teacher at an interval or lunchtime).

Participation:

Participation is voluntary and you have the right to withdraw from having information recorded about you at any stage (and have your information destroyed) before the 1st June 2018 without penalty.

The results of the project will be published, but your name and the name of your school will not. Your name (and your school name) will be changed to a fake name (pseudonym) in any published material. Also, anything you write or say during this study will be kept in a password protected computer. All information collected is securely stored for five years, and destroyed after this time.

You should know that this research is a little experimental (combining music and programming is experimental), and may not work exactly as planned. Your music teacher will ensure you will cover all that you need if there is anything missed (in future lessons) after this unit of work. Also, please be aware that some students in class may not participate in this research and may feel isolated or left out. The activities all students participate in will be exactly the same for the majority of class-time except for the in-class quizzes, student diaries, questionnaires and interviews. Non-participants will get extra time to complete their final projects and do a few listening exercises with the music teacher during these times. Non-participants will not be taking part in the interviews mentioned before outside of class time.

I will be sharing the results with the school and you can have the opportunity to view these results as well.

If you agree to participate in the study, you are asked to: (1) to make sure you understand everything (please ask questions until you do!), (2) complete and sign the consent form (attached on the next page but make sure you understand this as well), and (3) return the consent form to Chris Petrie or [name of participant music teacher].

College of Education, Health and Human Development
Telephone: +64 21 236 3100
Email: cpe20@uclive.ac.nz
16/01/2018

Interdisciplinary Computational Thinking with Music and Programming: A Case Study on Algorithmic Music composition with Sonic Pi

Consent Form for Participant Students

- ☐ I have been given a full explanation of this project and have had the opportunity to ask questions.
- ☐ I understand what is required for this unit of work.
- ☐ I understand that participation is voluntary, and I may request to remove my raw data from being included in this study before June 1st 2018.
- ☐ I understand that any information or opinions given in the raw data will be kept confidential to the researcher and the music teacher [name here] that any published or reported results will not identify the participants or school. I understand that a thesis is a public document and will be available through the UC Library.
- ☐ I understand that all data collected for this study will be kept in locked and secure facilities and/or in password protected electronic form and will be destroyed after five years.
- ☐ I understand the risks associated with participation in this research and how they will be managed.
- ☐ I understand that I can contact the researcher (Chris Petrie, email: cpe20@uclive.ac.nz) or any teacher at the school if I have any concerns or issues with this research.
- ☐ I would like a summary of the results of the project.
- ☐ I give permission to be selected for an audio-recorded semi-structured interview on the aims of this research before (for 5-10 minutes) and after (10-15 minutes) the unit of work. I understand that I will have an opportunity to check the transcript of all interviews for accuracy.

☐ By signing below, I agree to participate in this research project.

Name:____________________

Signed:____________________Date:__________________________

Email address *(for report of findings, if applicable)*:

Please return this form to Chris Petrie.

Appendix H: Semi-structured interview guide

Student pre-unit of work interviews:
Part 1 On attitude subscales (enjoyment, importance and self-confidence):

1. Can you describe your feelings towards programming? What aspects do you think you will enjoy or not enjoy? Why?
2. Do you think learning about programming will be important for you in the future? Why?
3. Do you think you'll be able to learn (or learn more) about skills in programming in the next few weeks? Why?

Part 2 On creativity:

1. Do you think programming or coding can be creative? What aspects might be creative? Why?
2. Do you think of yourself as creative? Why?

Part 3 About prior experiences in music composition and programming (if the participant has any):

1. Can you describe your prior experiences in music composition? Can you describe your feelings towards these experiences?
2. Can you describe your prior experiences in programming? Can you describe your feelings towards these experiences?

Student post unit of work interviews:
Part 1 On attitude subscales (questions as per pre-unit of work for comparison):

1. Can you describe your feelings towards programming? What aspects do you think you enjoyed or did not enjoy? Why?
2. Do you think learning about programming will be important for you in the future? Why?
3. How confident do you feel about learning programming in the future? Why?

On creativity:

1. Do you think programming or coding can be creative? What aspects are creative? Why?
2. Do you think of yourself as creative now? Why?

Part 2 About the unit of work (compare with prior experiences if they have any):

1. Can you describe your feelings towards the unit of work we did on Sonic Pi over the last few weeks? What aspects were good or not so good? Why?
2. Can you describe your experiences in music composition with Sonic Pi? How do these experiences compare with your previous experiences in making music?

3. Can you describe your experiences in programming with Sonic Pi? Can you describe your feelings towards these experiences? How do these experiences compare with your previous experiences in programming?

Part 3 Digital Artefact based interview (various code blocks were selected prior to the interview starting):

Interviewer will point to a selected code block. Suggested questioning (in no particular order):

- Tell me what's going on here?
- Step me through how each line of this code block works
- What does this code block do exactly?
- Why did you decide to have this sound/pattern?
- What did you struggle with the most here?
- What did you tinker with the most, why?
- How long did you work on this? How did it change over this time?

Pre-unit of work interview with the participant music teacher.

Part 1 Background:

1. Can you tell me about your prior training and teaching experiences in music?
2. Can you tell me about your prior teaching experiences in music composition? In music production?
3. Do you have any prior experiences in programming?

Part 2 On attitude subscales (enjoyment, importance and self-confidence):

1. Can you describe your feelings towards teaching programming? What aspects do you think you will enjoy or not enjoy? Why?
2. Do you think teaching programming will be important for you in the future? Why?
3. Do you think you'll be able to learn (or learn more) about skills in programming in the next few weeks? Why?

Part 3 On creativity:

1. Do you think programming or coding can be creative? What aspects might be creative? Why?
2. Do you think of yourself as creative? Why?

Part 4 On potential successes and challenges of the unit of work:

1. What predictions would you make about the potential successes and challenges of this unit of work?

2. Do you see any benefits for music in schools?

Post unit of work interview with the participant music teacher.

Part 1 On attitude subscales (enjoyment, importance and self-confidence):

1. Can you describe your feelings towards teaching programming? What aspects do you think you will enjoy or not enjoy? Why?
2. Do you think teaching programming will be important for you in the future? Why?
3. How confident do you feel about teaching programming in the future? Why?

Part 2 On creativity:

1. Do you think programming or coding can be creative? What aspects might be creative? Why?
2. Do you think of yourself as creative? Why?

Part 3 Predictions from the pre-unit of work interview:

1. You made several predictions about the challenges and successes of this unit of work. [*Read transcript of answers to these questions*]. To what extent do you think these turned out to be true?
2. What were you surprised about?

Part 4

1. Can you comment on your confidence levels for teaching Sonic Pi in the future by yourself?
2. Do you think you will plan for a unit on Sonic Pi with your students in the future?

Appendix I Individual student results for questionnaires, quizzes and project grades

Programming										Music								
Students grouped into grades for individual project programming	**Pre programming enjoyment**	**Pre programming importance**	**Pre programming self-confidence**	**Post programming enjoyment**	**Post programming importance**	**Post self-confidence programming**	**Total programming quiz results /30**	**Grade for group project for programming**	**Grade for individual project for programming**	**Pre music composition enjoyment**	**Pre music composition importance**	**Pre music composition self-confidence**	**Post music composition enjoyment**	**Post music composition importance**	**Post music composition self-confidence**	**Total music quiz results /20**	**Grade for group project music**	**Grade for individual project music**
<u>Students who received 5/5 (Extended abstract) for programming in their individual project</u>																		
Laura	1.33	0.67	1.00	2.67	2.00	2.00	27	Extended Abstract	Extended Abstract	-1.00	-1.00	-1.33	2.33	2.33	1.00	19	Relational	Extended Abstract
Sarah	0.67	0.00	1.00	3.00	1.00	2.00	20	Relational	Extended Abstract	-1.00	-1.00	-1.00	1.33	2.33	1.33	18	Extended Abstract	Extended Abstract
Ben	0.67	1.33	0.00	3.00	2.67	2.00	28	Relational	Extended Abstract	2.33	1.33	1.33	3.00	2.33	2.00	19	Extended Abstract	Extended Abstract
Sam	1.33	0.67	0.33	2.33	2.33	1.67	25	Extended Abstract	Extended Abstract	1.67	0.67	1.67	3.00	2.00	1.67	17	Relational	Relational

Norman	2.33	1.33	0.67	2.67	2.33	2.00	25	Extended Abstract	Extended Abstract		2.33	0.33	2.00	3.00	2.33	2.67	19	Extended Abstract	Extended Abstract
Students who received 4/5 (Relational) for programming in their individual project																				
Charlotte	-0.67	-1.33	-0.33	1.67	1.67	1.33	19	Multi-structural	Relational		-1.67	-1.33	-2.33	1.33	1.67	1.00	16	Relational	Relational
Emma	1.00	-1.33	1.00	1.33	1.00	1.67	19	Relational	Relational		-0.33	-0.67	-1.33	1.33	1.67	1.00	18	Extended Abstract	Extended Abstract
Kate	-0.33	-1.33	-1.00	0.67	1.00	0.33	22	Extended abstract	Relational		-1.67	-1.67	-2.00	-0.67	0.67	-0.33	17	Extended abstract	Extended Abstract
Olivia	1.00	1.00	1.00	2.33	1.67	1.67	20	Multi-structural	Relational		-0.33	-0.33	-1.33	1.67	1.67	1.67	20	Extended Abstract	Relational
Liam	-1.00	-1.67	-1.67	1.00	1.67	1.00	21	Relational	Relational		-0.33	-0.67	-0.33	1.33	1.33	1.33	15	Multi-structural	Relational
William	-0.33	-0.33	-1.33	2.00	2.00	1.33	20	Multi-structural	Relational		0.67	0.67	1.00	2.67	2.33	2.00	17	Relational	Multi-structural
Students who received 3/5 Multi-structural for programming in their individual project																				
Jacob	1.00	1.00	0.33	1.67	1.67	1.00	20	Multi-structural	Multi-structural		0.67	0.67	1.00	2.67	2.00	2.00	11	Multi-structural	Multi-structural
James	-0.33	-0.67	-1.33	0.33	1.33	-0.33	16	Relational	Multi-structural		0.33	-0.33	1.00	1.33	1.00	1.33	14	Uni-structural	Multi-structural
Michael	-2.33	-1.33	-2.00	1.33	2.33	1.33	17	Multi-structural	Multi-structural		-0.33	-0.33	-1.00	1.67	1.67	1.33	15	Multi-structural	Multi-structural

Oliver	-0.67	-0.67	-1.33	1.33	2.00	1.67	17	Uni-structural	Multi-structural		1.33	0.33	1.00	1.67	0.67	1.67	15	Multi-structural	Relational
Elizabeth	-1.67	-1.67	-2.00	0.33	1.33	0.33	18	Relational	Multi-structural		-2.67	-3.00	-2.67	-0.67	0.00	0.33	18	Multi-structural	Multi-structural
Grace	-1.00	-0.33	-0.33	-0.67	-1.33	0.00	15	Multi-structural	Multi-structural		-1.67	-1.67	-1.33	0.33	1.33	0.00	17	Multi-structural	Multi-structural
Sophia	-0.33	0.00	-0.33	-0.33	-0.67	0.67	15	Multi-structural	Multi-structural		-2.00	-2.00	-1.67	-0.67	-0.33	0.00	14	Uni-structural	Multi-structural
Students who received 2/5 Uni-structural for programming in their individual project																			
Ava	-2.00	-1.67	-1.67	-2.67	-0.67	-0.67	12	Multi-structural	Uni-structural		-2.00	-2.67	-2.00	-1.67	-1.00	-1.33	11	Uni-structural	Uni-structural
Aiden	1.00	0.33	1.00	2.33	1.67	2.00	11	Multi-structural	Uni-structural		1.67	0.67	1.33	2.33	2.00	1.67	9	Uni-structural	Uni-structural
Daniel	0.67	0.67	1.00	1.67	2.00	0.33	8	Uni-structural	Uni-structural		-0.33	-0.33	-0.33	2.67	1.33	1.67	12	Uni-structural	Uni-structural
Lucas	1.00	0.33	-0.33	1.67	2.33	1.33	7	Multi-structural	Uni-structural		2.00	1.33	1.67	2.33	2.33	2.00	10	Uni-structural	Uni-structural

Appendix K: List of YouTube video links

Video #1 (1:35)
Theme and goal: Climate change – United Nations Sustainable Development Goal 13
Title: A Homeless Polar Bear in London
Published by: Greenpeace International
Internet Link: https://www.youtube.com/watch?v=4XpF04nximI

Video #2 (1:15)
Theme and goal: Refugee crisis – United Nations Sustainable Development Goal 16
Title: What if Manhattan...
Published by: Hamdi Foundation
Internet Link: https://www.youtube.com/watch?v=Vc_VNvD9B3c

Video #3 (1:30)
Theme and goal: Refugee crisis – United Nations Sustainable Development Goal 16
Title: UNICEF | for every child
Published by: UNICEF
Internet Link: https://www.youtube.com/watch?v=E1xkXZs0cAQ

Video #4 (1:33)
Theme and goal: Refugee crisis – United Nations Sustainable Development Goal 16
Title: Most Shocking Second a Day Video
Published by: Save The Children
Internet Link: https://www.youtube.com/watch?v=RBQ-IoHfimQ

Video #5 (1:20)
Theme and goal: Refugee crisis – United Nations Sustainable Development Goal 14
Title: How does plastic end up in our oceans?
Published by: Greenpeace UK
Internet Link: https://www.youtube.com/watch?v=Our5CZz5qoU