

FACULTY OF SCIENCE

School of Computer Science

TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

ERNI MARLINA SAARI

Thesis submitted to the University of Nottingham UK

For the degree of Doctor of Philosophy

November 2017

Abstract

The need for teachers of Elementary School children to learn to program or rather to understand the Computational Thinking behind programming has been accelerated in many countries by the mandated teaching of programming in the Elementary School context. Many steps have been taken in order to create awareness of this issue, such as the Computing At Schools initiative (CAS) which is established in the UK. CAS aims to support teaching in computing and connected fields in UK schools. Moreover, in the USA the Computer Science Teachers Association (CSTA) was established to meet the purpose of informing and advising about the current development of computational thinking and to investigate and disseminate teaching and learning resources related to computational thinking. In Singapore research has been conducted by the government agency Infocomm Development Authority of Singapore (IDA) whereby the major goal is to meet the needs in the ICT sector and ultimately to focus and inspire learners about programming.

The research for this thesis involves the development of a training scheme for pre-service teachers that will introduce them to computational thinking through the use of the Flash Action Script Development environment. Flash Action Scripts – amongst several other tools – are used as a tool for creating interactive content and because Flash is one of the premiere tools used to create content for the internet; a tool programmed with Flash looks practically the same in every browser and on every operating system. Flash Action scripts use traditional coding skills but permit the user to see how each piece of code affects the running or execution of the program, allowing the user to have an instant visual understanding of what the code is doing. It is also widely available within university campuses.

A major problem in promoting the teaching of programming and computational thinking to Elementary School teachers is that the majority of such teachers have no concept of how to program and naturally are not motivated to learn programming. Experienced teachers involved in the current study felt that programming was too complicated and thus it was hard to gain fluency in programming. Student teachers who had no previous experience in programming were, however, easier to get engaged in

learning programming principles. Eighty percent of this group found Action Scripting a useful tool to understand basic programming and scripting. The need to teach programming will motivate most but to learn through a tool that can be seen to have intrinsic value in their role as teachers has a greater potential of success. This thesis defines the design and implementation of a tool to use the learning of Flash Action Scripting as a motivational mechanism for pre-service teachers. The intrinsic value to them is intended to be utilisation of the learned Action Scripting skills to produce their own teaching material. Initial results indicate an enhanced engagement and motivation to learn to program and improved confidence in doing so. As projected the pre-service teachers had a more positive attitude towards the potential of the learning tool but both they and the in-service teachers had improved attitudes and enthusiasm after the experiment. The results show that both pre-service and in-service teachers can be trained to be designers and producers of digital courseware in the previous absence of computational thinking skills and definitely they can acquire skills in computer programming such as Flash Action Scripts.

Acknowledgements

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, an abundance of praise is due to The Rabb, The Merciful for His nurture, excessive mercy and blessing in this thesis completion.

In the completion of this thesis, I owed many people in different ways. First and foremost I would like to render my gratitude to the management of the Sultan Idris Education University (UPSI) for awarding me the scholarship and study leave and The Ministry of Higher Education Malaysia for the full sponsorship. Throughout my studies here, I am truly indebted and thankful for all the support that I have received. In general like to thank the following people:

Special thanks to both of my supervisors, Associate Professor Dr. Peter Blanchfield and Associate Professor Dr. Gail Hopkins, for guiding me and relentlessly advising me without fail, assisting and pointing me in the right direction. I have learned so much from both of them and I will always cherish of all the lessons, meetings and advice given.

My appreciation also goes to all my friends for helping me and make my PhD life enjoyable.

To my ever loving family back home, thank you for the prayers for my success in this task entrusted upon me. My beloved mother, Zaiton and all my siblings, I hope I have made you proud and essentially thank you for all the support. And I know, I would have made bapa proud too, Al-Fatihah.

Words and numbers could not represent the amount I owed to my own family, my children Nurdayana Batrisya, Danish Irfan, Danish Firhan and Nurdamia Aleesya. Finally, if anyone deserves a multitude of recognition and gratitude, it will be my dearest husband, Mohamad Zamri Abdul Aziz who has not only been the heart to the family, but also my inspiration. Thank you for the love, the humor and most importantly the constant assurance that 'I can do this' for without your love and sacrifices, PhD is only a dream afar. I dedicate this thesis to you.

Declaration of Authorship

I, Erni Marlina Saari, declare that the thesis entitled "Toolbox for Adopting Computational Thinking through Learning Flash Action Scripts" and work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that this work was done wholly or mainly while in candidature for a research degree at this University;

Where I have consulted the published work of others, this is always clearly attributed;

Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

I have acknowledged all main sources of help;

Where the thesis is based on work done by myself, I have made clear exactly what was done by others and what I have contributed myself;

Part of this work have been published as:

Saari, E.M., Blanchfield, P. and Hopkins, G., 2015. Learning Computational Thinking through the Use of Flash Action Scripts: Preparing Trainee Elementary School Teachers for Teaching Computer Programming *In: CSEDU 2015, The 7th International Conference on Computer Supported Education, Proceedings* 2: 75–84.

Saari, E.M., Blanchfield, P. and Hopkins, G., 2016. Computational Thinking - A Tool to Motivate Understanding in Elementary School Teachers. Pp. 348–364 in *Computer Supported Education*, *Chapter of Book*, edited by Markus Helfert Susan Zvacek, Maria Teresa Restivo, James Uhomoibhi. Switzerland: Springer International Publishing Switzerland.

Signed:  March 2018

Table of Contents

Chapter 1 1

1.1 BACKGROUND OF THE STUDY 1

1.2 RESEARCH QUESTIONS 7

1.3 STRUCTURE OF THE THESIS 11

1.4 EXPECTED CONTRIBUTIONS 12

Chapter 2 13

2.1 INTRODUCTION 13

2.2 COMPUTATIONAL THINKING BACKGROUND 14

2.2.1 Introduction 14

2.2.2 Defining Computational Thinking 15

2.2.3 Adoption of Computational Thinking 15

2.2.4 Pedagogical Approaches to Teaching Programming and CT 16

2.2.5 CT in the primary curriculum 19

2.3 TOOLS AND TECHNOLOGIES FOR TEACHING PROGRAMMING AND CT 23

2.3.1 Graphical User Interfaces for Programming 24

2.3.2 Serious Games 28

2.3.3 Robotics and Agents 30

2.3.4 Tangible Programming 33

2.3.5 Collaborative Learning Approaches 35

2.4 ASSESSING COMPUTATIONAL THINKING 39

2.5 MOTIVATION 48

2.5.1 Motivation to learn 49

2.5.2 Types of Motivation 50

2.5.3 Motivating In-Service Teachers and Pre-Service Teachers 51

2.6 GLOBAL RECOGNITION OF COMPUTATIONAL THINKING 52
2.6.1 Digital Economy Initiatives 52
2.6.2 Continuing Professional Development (CPD) in the Malaysian Education System 56
2.7 CONCLUSION 58
Chapter 3 60
3.1 INTRODUCTION 60
3.2 OUTLINE OF WORK 61
3.3 METHODS 61
3.3.1 Questionnaires 62
3.3.2 Observations 64
3.3.3 Interviews 66
3.3.4 Prototyping 67
3.3.5 Pilot Study 69
3.4 QUALITATIVE ANALYSIS 70
3.4.1 Content Analysis 70
3.4.2 Thematic Analysis 71
3.5 RELIABILITY AND VALIDITY OF DATA 71
3.6 ETHICAL CONSIDERATIONS 73
3.7 CONCLUSION 74
Chapter 4 75
4.1 INTRODUCTION 75
4.2 PROGRAMMING TEACHING PROVISION IN MALAYSIA TEACHER TRAINING COURSES 75
4.3 ATTITUDES TOWARDS PROGRAMMING IN UPSI TEACHER TRAINING STUDENTS 77

4.3.1 Participants 77
4.3.2 Method 78
4.3.3 Thematic Coding 79
4.3.4 Results 84
4.3 CONCLUSION 89
Chapter 5 92
5.1 INTRODUCTION 92
5.2 METHOD 93
5.2.1 Participants 93
5.2.2 Experimental Design 94
5.2.3 Tasks Design 95
5.3 RESULTS 100
5.3.1 Questionnaires 100
5.3.2 Interviews 113
5.4 DISCUSSION 114
5.5 CONCLUSION 115
Chapter 6 117
6.1 INTRODUCTION 117
6.2 TOOL DESIGN 118
6.2.1 Architecture of the teaching tool 118
6.2.2 Prototyping of the Tool 119
6.2.3 Design Tool 120
6.3 PILOT TEST 128
6.4 EXPERIMENTAL METHOD 133
6.5 RESULTS – PROCESS OF LEARNING 134
6.6 RESULTS – FINAL OUTCOMES 144

6.6.1 Completing a complex artefact. 144
6.6.2 Changes in Attitudes. 145
6.6.3 Participants' ability to use another animation tool - Generalisation in CT. 148
6.7 CONCLUSION 150
CHAPTER 7 152
7.1 INTRODUCTION 152
7.2 RESEARCH QUESTIONS 153
7.3 LIMITATIONS 163
7.4 FUTURE WORK 164
7.5 CONTRIBUTIONS 165
REFERENCES 167
Appendix A 184
Appendix b 185
Appendix c 187
Appendix d 190
Appendix D 192
Tutorial Lesson for Study 2 Experiment 192
Appendix E 194
Lecture on Flash Action Scripts 194
Appendix F 202
Ministry of Education Malaysia 202
Appendix G 204
Appendix H 214
Appendix j 231
Appendix I 232

Appendix II 234

Appendix III 236

Appendix IV 237

Appendix V 239

Appendix VI 240

Appendix VII 242

Appendix IX 246

APPENDIX X 252

List of Figures

Figure 2.1: Storytelling Alice Interface. 25
Figure 2.2: Scratch Interface 26
Figure 2.3: Lego© NXT Programming. 33
Figure 2.4: Tur Tan Interface Design 35
Figure 2.5: Cleogo Screen Layout 37
Figure 3.1: Outline of Research Study. 61
Figure 4.1: *Initial thematic map, showing five codes* (analysis as presented in Braun & Wilkinson, 2003). 80
Figure 4.2: *Thematic map, showing one theme and a few codes* (analysis as presented in Braun & Wilkinson, 2003) *towards respondents' attitude to using Flash.* 81
Figure 5.1: Fundamental Step for Creating Text, Graphic and Navigation Link. 96
Figure 5.3: The Compiler Error Log Screen 97
Figure 5.2: The Action Script Screen Showing a Movie Clip Action for Animation 97
Figure 5.4: Complex Syntax for the Lesson Module. 99
Figure 5.5: Adding On Release Syntax for a New Function 100
Figure 5.6: Learning About Animation through Flash Action Scripting. 108
Figure 5.7: Screen Captures on Wrong Thinking in Producing Scripts. 110
Figure 5.8: Error Screen captured due to Copy and Paste Attitude. 111
Figure 5.9: Increasing Confidence Level and Improved CT Skills (Respondent ET2). 112
Figure 5.10: Good Understanding of the Concept of Abstraction (Respondent TT2). .. 112
Figure 6.1: Design and Basic Flow Using the Tool. 119
Figure 6.2: The Home Screen. 120
Figure 6.3: Enquiry Screen from Training Tool. The Place Where the Participant Clicked Revealed Any Problems They Had. 122
Figure 6.4: Layer screen - Adapting Abstraction Domain from CT Skill. 123
Figure 6.5: Interaction Screen - Adapting Decomposition from CT skill. 124
Figure 6.6: Navigation Screens. 124
Figure 6.7: *Lesson Tool Switch-Case Screen* 125
Figure 6.8: A Respondent's Skill in Developing a Complex Coding Page 126

Figure 6.9: Analysis – debugging errors 127
Figure 6.10: Different Screens Designed by Two Respondents. 135
Figure 6.11: Participants Successfully Attempt to Use Switch Case Syntax. 135
Figure 6.12: Additional Navigation Links Created by Respondent. 136
Figure 6.13: Additional Commands Had Been Practiced. 137
Figure 6.14: Additional Feature and Technical Skill Developed from Respondent (TT2). 138
Figure 6.15: The Same Task performed in Anime Studio (TT2)..... 149
Figure 6.16: Short Movie Clip using Anime Software (TT3). 150

List of Tables

Table 4-1*: Example of Positive Themes in Coding Enthusiasm and Attitude*. The example for each quote was selected from respondents' feedback. *(data extracted with codes applied from Clarke, Burns & Burgoyne, 2005)* 82

Table 4-2: Examples of Negative Themes in Coding Enthusiasm and Attitude. The example for each quote was selected from respondents' feedback. (Data extracted with codes applied from Clarke, Burns & Burgoyne, 2005) 83

Table 4-3: Examples of Codes Related to Professional Enhancement. Each example quote was selected from respondents' feedback. (Data extracted with codes applied from Clarke, Burns & Burgoyne, 2005). 84

Table 4-4: Attitudes and Emotions Before and After the Module. 88

Table 4-5: Responses Relating to Professional Skills. 89

Table 5-1: Example Positive and Negative Responses from Participants before the Activity. The example for each quote was selected from respondents' feedback. 101

Table 5-2*: Example Positive and Negative Responses from Participants after the Activity.* 102

Table 5-3: Frequency of Appearance of Code Phrases in Use by the Pre-Service Teachers and the In-Service Teachers **before and after** Taking Part in the Activity in Study 1. .. 103

Table 5-4: Frequency of Appearance of Code Phrases in Use by the Pre-Service Teachers (TT) and the In-Service Teachers (ET) **before** Taking Part in the Activity in Study 2. 104

Table 5-5: Frequency of Appearance of Code Phrases in use by the Pre-Service Teachers (TT) and the In-service Teachers (ET) **after** taking part in the Activity Study 2. 105

Table 6-1: Positive Responses from Participants after the Pilot test (The example for each quote was selected from respondents' feedback. 128

Table 6-2: Code Frequencies After Completion of the Pilot Study. 130

Table 6-3: Enthusiasm and Attitude Codes from Study 3 Responses..... 139

Table 6-4: Computational Thinking Domain Thematic Codes and Example Quotations. 141

Table 6-5: Professional Practice Domain Thematic Codes and Example Quotations. .. 143

Table 6-6: Code Frequencies **Before** Completion of the Tool Based Training (Study 3). 146

Table 6-7: Code Frequencies **After** Completion of the Tool Based Training (Study 3).. 147

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The need for more programmers is now serious and software companies and governments around the world have become convinced that a change in the way technology is taught in schools is needed. It is felt vital that children begin to learn to program and more essentially think like programmers from an early age. An important issue arises in how to introduce programming at an early age: how can teachers without a background in computing learn how to teach programming? A common sense answer to this might be that they should first learn to program themselves but how should this be tackled? Importantly, what will motivate them to learn?

The work in this thesis focuses specifically on tackling this problem in a Malaysian context though much of what needs to be done will be relevant in a broader context. Educators the world over are trying to respond to these challenges. The situation in Malaysia needs specific responses relating to the culture of education in that context but many of these problems will also be true elsewhere. Of specific interest is tackling the expectations of continual professional development in Malaysia. At the outset of this research it was expected that there would be much to learn from the experience in other places and particularly in the UK where work was already underway in tackling this.

The term computational thinking (CT) is a way of expressing the sort of thinking needed by programmers. In introducing this concept to a broad audience of those with an interest in computing Jeannette Wing (2006) tried to classify CT as ways of thinking that went beyond programming specifically and looked at it as a way of approaching problem solving more generally. The idea was that every student should be given the chance to engage with computational thinking independent of their area of study (J. M. Wing 2006a). CT involves more than learning to program, but many of the concepts in

CT are exercised during programming (Duncan, Bell, and Tanimoto 2014). CT can be exploited in many situations and can lead to better approaches to problem solving throughout life. By Wing's definition CT is human thinking (problem solving by humans rather than just machine thinking) and involves taking an approach to solving problems, designing systems, applying tools, techniques and understanding of human behaviour that draws upon concepts essential to computing (J. M. Wing 2006a; J. M. Wing 2008; Grover and Pea 2013). CT does not mean that all students must master the details of a complex programming language but it is essential for students to understand the ideas of programming, to appreciate the difficulty of expressing a solution in a form that can be interpreted by a machine, and to grasp the concepts of debugging (Curzon et al., 2009). As such it creates an important focus on the concept of thinking about problems in a way that leads to solutions that may be implemented in a computing device (or perhaps a teaching tool as will be talked about in this thesis). Making CT manageable for teachers or students may mean it can be possible to teach computing more effectively. Whilst computing education researchers explore how humans come to understand computing, and how to improve that understanding, the research in this thesis studies the idea that learning to think computationally will enhance pre-service teachers' abilities to learn how to teach programming.

The concept of Computational Thinking has thus become a topic of discussion (Dorling M., 2014; Denning, 2009; J. M. Wing, 2006a) and the role that this plays in developing the correct thinking for learners to become programmers is clear (Dorling M. 2014; Csizmadia et al. 2015). It is not expected that teachers will themselves become programmers but they need to acquire the skills to teach programming and this is the case in countries throughout the world.

Malaysia's vision to become a developed nation by the year 2020 (Bernama 2008) has placed science and technology at the top of the list of subjects in which to excel. There is no specific subject on computer programming that has been developed for primary school students in Malaysia. Learning has instead been more focused on use of application software (L. Lezam et al., 2015). Computer programming is to be incorporated into the national curriculum of primary schools in Malaysia from 2017 according to the MDEC (Multimedia Digital Economy Corporation) CEO Datuk Yasmin

Mahmood, indicating the importance that is now being placed on coding skills (Azlee 2016). "Educators should be made aware that teachings of programming are not only confined to acquiring the skill of programming but acquiring this skill could also increase the understanding of more fundamental subjects like language, mathematics and science" (Husain, Kamal, and Ibrahim 2017). This was also announced by Malaysia's Ministry of Education Director-General Khair Mohamad Yusof, who confirmed that Computational Thinking (CT) would additionally form an important part of the curriculum from January 2017 (Azlee 2016; A. Asohan 2016). However, this is not a completely new directive; in 1988 the former Education Minister Anwar Ibrahim announced that computing skills and programming would be part of the curriculum by 1990 (A. Asohan 2016). However, political issues prevented this objective from being realised at the time (A. Asohan 2016).

Whilst there are defined plans to raise the standard and quality of ICT education in Malaysia (Dewitt, Alias, and Siraj 2016; Suliman, Hawari, and Othman 2011; Khambari et al. 2010; A. Asohan 2016) and it is accepted that teaching coding in schools is important (Bernama 2017) awareness of CT is still low when compared with countries such as the USA and the UK. Challenges exist around the fact that learning to programme is not just about having Computer Science as a subject on the curriculum (Grover and Pea 2013; Brown and Kölling 2013; Gretter and Yadav 2016). It is important that CT skills, such as learning how to dissect problems and formulate solutions, form a basis for good programming (Ebrahimi et al., 2013; Kelleher et al., 2007; Sentance & Csizmadia, 2016). Additionally, any programme of education would need to involve not only educating students, but also educating teachers to teach coding skills. Presently, in Malaysia, there has been little in the curriculum which has addressed this and little provision for pre-service and in-service teachers to learn to teach programming. Teachers have recently shown concern regarding the programme to integrate coding into the National School Curriculum stating that they had not been informed of the programme early enough (Azlee 2016). Malaysia's National Union of the Teaching Profession President Hashim Adnan has reinforced this view and emphasised the importance of training the next generation of teachers in a systematic way before the introduction of teaching coding in schools. He stated that without a proper foundation

to the programme, it would fail, but that no preparation appeared to be taking place (Azlee 2016). It is apparent that a lot of work needs to be done if the country is to achieve its goal of a full roll-out of ICT provision in all schools by 2020.

A study that was conducted at one of the top education universities in Malaysia highlighted Malaysian students' lack of problem solving skills (Dewitt, Alias, and Siraj 2016). The Malaysian Education Blueprint from preschool to post-secondary, and for higher education has noted that Malaysian students need to develop thinking skills in order to be prepared for their future jobs (Ministry of Education (MOE), 2013; 2015). In January 2017, computational thinking was made part of the reviewed Standard Curriculum for Primary Schools (KSSR) for Year One pupils – making Malaysia the first country in Asean to introduce it into the national syllabus. Deputy Education Minister Datuk Chong Sin Woon added that the programme had been developed with the help of MDEC and that the syllabus was benchmarked against courses offered by Britain's Computing at School as well as the United States' Computer Science Teachers Associations. "Computational thinking is the ability to dissect problems and formulate solutions by drawing from concepts in computer science." (https://www.digitalnewsasia.com) said MDEC vice-president for talent and digital entrepreneurship Sumitra Nair (http://www.digital-economy/computational-thinking-come-fore-malaysian-schools). Malaysia's education ministry director general Tan Sri Dr Khair Mohamad Yusof said this is not just about having computer science as a subject. Malaysia currently must begin educating students to have computational thinking skills, linking the move to also helping Malaysia achieve its goal of being in the top 30% of the 2025 Programme for International Student Assessment (PISA) (http://www.oecd.org/pisa/aboutpisa/). MDEC and the Ministry of Education are strongly concerned that the responsibility to educate, excite and enrich students cannot just start and end in school, which is why the private sector and academia have been tied in through the #mydigitalmaker movement. #Mydigitalmaker is a public-private-academia partnership scheme to move away from being digital users and towards a nation of digital developers. According to MDEC, the private sector and academia will continue to nurture and groom students outside of the classroom through the introduction of Digital Maker Clubs in schools and Digital Maker Hubs in communities.

There are six such Digital Maker Hubs in Malaysia at present and this is expected to rise to 14 by the end of 2018.

Students should be inspired in order to help them succeed in any academic assignment. Jenkins and Davy (2002) stated that the goal of a teacher should not only be to transfer knowledge to their learners but also to motivate them to learn and to help them realize the benefits of learning. Learners should be encouraged to explore and apply the concepts to other aspects of learning. However nowadays, it is hard to nurture and maintain students' motivation to study programming merely with the classical programming exercises designed to perform a sequence of calculations.

Over the past 25 years the understanding of the existence of numerous learning styles has brought increasing attention to the idea that students learn in diverse ways and that the purely verbal approach to teaching does not work for every student or even most students (Hawk 2007). The importance of programming skills is becoming more evident in the UK as they have been made core curriculum objectives for school students. In order to teach programming there needs to be a route for teachers to develop an understanding of the underlying concepts. However, it is not necessary to have an understanding of computer programming to design and produce digital material for teaching (Werner et al. 2012). The University Pendidikan Sultan Idris (UPSI) in Malaysia is typical of teaching universities in Malaysia. Personal experience teaching there in early 2011, was the starting point for the current research. It was realized that what a group of students (that did not have any prior computing skills yet took a multimedia programming course) needed was to be provided with the right methods of learning - so they could practice success, which would inspire them to continue studying multimedia programming after leaving the class. In UPSI Adobe Flash Action scripting was being taught to students including those without a computing background. Action scripting allows users to add dynamicity to their multimedia presentations and this has prompted the research for this thesis to investigate whether learning through action scripting can be used to teach underlying programming concepts to teachers while they create their learning material (Lin 2012); in other words, whether it can stimulate the development of CT and therefore an understanding of programming. Some of the components of CT include the following (http://beaver.my): formulating problems in a

way that enable the use of computers and other tools to help solve them; logically organizing and analysing data; representing data through abstraction such as models and simulations; automation of solutions through algorithmic thinking (a series of ordered steps); identifying, analysing and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources and finally, generalizing and transferring this problem solving process to a wide variety of problems. Beaver is reasonable for all students within the age group between six to nineteen years old. Participants do not need prior computer science or programming knowledge. There are 5 categories for students aged 6 to 19 (Year 1 to Form 6). According to Malaysia Deputy Education Minister Datuk Chong Sin Woon, 8 000 teachers in Malaysia underwent training in July 2017 and will be fully equipped to teach computational thinking to students one to two years later. Datuk Chong added that this special training was being conducted with the help of 100 lecturers from teacher training institutes, ministry officers who are British Computer Society-certified master trainers, as well as 90 lecturers from Institut Aminuddin Baki (IAB) (http://www.thestar.com.my/news/nation/2017). Because the term 'computational thinking' is a relatively recent introduction (J. M. Wing 2006a), there is less literature published in Malaysia for instance specifically with this key word. Therefore, the results of the proposed study should help fill the gap between the teaching of programming and computational thinking, specifically in the context of the pre-service and in-service teachers in the Malaysia educational system.

Although programming is a separate skill from CT, the latter draws heavily upon concepts fundamental to computing (Wing, 2008). CT encompasses several skills important for problem solving and can result from studying the nature of computation. It also incorporates creativity, the ability to explain and team work. Effective programming relies on the ability to think computationally, particularly the ability to think logically, algorithmically and recursively (Hu 2011; J. M. Wing 2008; Yinnan and Chaosheng 2012). Specifically, this refers to how a user thinks when producing programs, rather than producing the code itself. Such skills are both important to learn when teaching others to program and to transfer to other problem solving domains in life. For many the whole concept of programming seems foreign and difficult and much

work has been done to look at how to get people to learn programming (Crawford and Boese 2006; Martins, Mendes, and Figueiredo 2010). Much that is successful involves people learning through concrete problems where they understand what the result of the process should be. The internal workings of computers are, however, mysterious to many and so conceiving what is needed to solve a problem through a computer program can be too difficult to imagine and students can be fearful of trying.

In order to motivate students to learn programming many instructors have tried to use games, robots and narrative media hoping that they would impress their students (this is covered in Chapter 2). Jenkins (2001b) highlights the difficulty in obtaining solid conclusions from research on the motivation of students for programming. Students tend to learn in different ways and so wish to use varied teaching resources. There is a variety of tools that have previously been used in teaching programming but these tools generally require some computing background from the students.

The current work has focused on working with teachers who need to learn to program and who desire to learn programming and to develop teaching tools to demonstrate concepts to their students. Electronic artefacts to demonstrate a specific learning area are not always available commercially and those that are available are often expensive and may not be an exact match for the specific needs of the teacher.

The primary aim of this research was therefore to try to establish a way of changing accessibility to programming for many and a number of research questions were posed, as detailed in the following section.

1.2 RESEARCH QUESTIONS

The discussion so far has set the scene for the main hypothesis of this research. That is in order to meet the desire that children in Malaysia (and elsewhere) learn to program it is first essential that their teachers understand the thinking behind programming and are thus more able to understand how to teach it. At the time this was being conceived the position in Malaysia was particularly difficult. There was little or no incentive for teachers to learn to teach programming. However, the basic belief behind this research was that the need was there and would soon become critical. This belief has been

borne out by more recent developments that are making the teaching of programming compulsory at all levels in school education in Malaysia. Experience of teaching trainee teachers had led to the conclusion that many teachers with non-computing backgrounds would be motivated by learning how to create artifacts using a tool like Flash Scripting. Further this would allow them to have a reason to learn to think computationally for themselves. To investigate the feasibility of teaching programming using a computer-based tool the proposed work needed to be divided into a number of steps. The first step was to investigate more formally the impression given by experience that teachers and trainee teachers were generally not already thinking computationally. Thus the first research question was formulated as:

RQ 1 Are primary school teachers ready to teach programming?

In order to answer each of the questions a number of objectives needed to be attained. In this case:

- To more formally identify attitudes to programming among teachers and trainees in particular.
- To identify the challenges for them in learning to teach programming.
- To determine to what extent they already think computationally and identify ways in which this might be helping or preventing them meeting the challenges of learning to program.

Achieving these objectives would imply a number of things:

First a literature search was needed to look at other research evidence of attitudes of trainee and other teachers to learning to program One aspect of the literature review would be to find ways in which computational thinking was already being measured and thus what would be needed to measure its presence or absence in the teachers. It would also be necessary to survey trainee teachers more formally about their attitudes. This was undertaken at several stages in the research. Previous experience in teaching teachers to use IT tools had suggested that they were generally fearful of learning to program and of using other technology beyond office tools. Thus a second research question was expected to need answering:

RQ 2 Can those without a CT background learn to think computationally?

One way in which this would be expected to be promoted would be through motivation. Thus this question implies that many teachers and trainee teachers will be found not to think computationally and probably not be naturally intrinsically motivated to learn. This then implies another objective:

- To understand how teachers might be motivated to learn to think computationally.

Thus part of the literature review would focus on motivation. Previous experience of trainee teachers at UPSI was that learning to use Flash Action scripting was attractive to non-computing students. Thus it was expected that learning this would be a possible motivator to change their attitudes at least. Thus the first intervention was designed to carry on from the initial survey and concentrate on those who had already expressed an interest in taking an elective module in Flash Action Scripting (Flash). The initial survey would then be repeated with the same group of students after exposure to Flash. This would allow a third research question to be posed:

RQ 3 Can we detect where CT is not happening and lead the learner to re-analyse their thinking strategies and consequently learn to think computationally?

Objective:

- To work out how to detect use or lack thereof of the five CT domains: abstraction, algorithmic thinking, decomposition, generalization and analysis.

Should it be possible to teach CT, part of the value in this, and an aim of the research, was to see whether the newly acquired skills could be transferred to computer programming. Thus both in the second and third phases of this research it was the aim to see whether first Flash and subsequently a similar but different software could be programmed effectively by the learners who had been originally unfamiliar with the ideas in programming. Consequently the next research question was:

RQ 4 Can pre-service teachers who have adopted CT demonstrate this by applying it to undertaking "programming" tasks?

In order to test this two more objectives would be looked at:

- To provide tasks the successful completion of which would demonstrate the use of CT.
- To provide tasks that would demonstrate they can generalise what they learned to using other software.

The initial exposure of the large survey group would be done through a traditional class in UPSI. However, to answer RQ3 and 4 it would be necessary to conduct a more detailed instruction in order to spend time looking at the learner's behavior. Thus a second phase of the study was proposed to tackle these questions. This would involve a small group of students from a similar cohort to the previous phase being given one on one teaching in the use of Flash while being observed for their behaviour as well. The evaluation of this leads to a further research question and allows an automated teaching/learning tool to be designed focused on a user centred design approach.

RQ 5 Does the use of a suitable tool in learning create users who will be motivated to go on programming?

Objectives:

- To identify changes in attitude after use of the tool compared to before.
- To identify what barriers still remained.

Again positive outcomes from the previous research questions will lead to a final research question which asks about the possible implications of the study as a whole.

RQ 6 What are the implications of this study for the teaching of CT and programming in teacher training courses in Malaysia?

The final question can only be completely answered when the research is over and it is possible to engage with the authorities in Malaysia. At the time of writing the Malaysian government have already started to ask questions which this research will have begun to answer.

1.3 STRUCTURE OF THE THESIS

The remainder of this thesis is structured as follows:

Chapter 2 commences with a review of the relevant literature and outlines the broad perspectives and related concepts – programming and CT – which are then linked to the discussion of findings in chapter seven. The subjects of the literature review are designed to aid the answering of the research questions. So for example RQ 1 begins to be answered by knowing what the literature has to say about attitudes of teachers to learning programming.

Chapter 3 frames the methodological considerations in carrying out this investigation. It presents the methods that were used in this study, including recruitment and participant selection, and discusses the potential ethical concerns and pitfalls of the approach. The methods look particularly at ways of gathering data on the attitudes of the teachers and trainee teachers towards learning to teach programming. Thus they involve questionnaires and the use of open ended questions. Chapter 3 seeks to give background to why these research methods were appropriate.

Chapter 4 presents Study 1 which involves observations of Malaysian pre-service teachers at UPSI in order to identify difficulties experienced with learning to program. This will give further answer to RQ1 in particular and RQ2 in part.

Chapter 5 describes Study 2 which looks at how an individual one-on-one approach to teaching programming works among pre-service and in-service teachers. In doing so, it seeks to identify where non-CT exists and how it affects the learning of programming. It also seeks to identify ways in which to change that thinking. In doing so it lays the basis for the design of the teaching/learning tool that is presented in Chapter 6 and allows RQ2 and RQ 3 to begin to be answered.

Chapter 6 presents Study 3, the implementation of a tool to automate the teaching of CT skills and its evaluation with Malaysian pre-service teachers and in-service teachers who were resident as students in the UK. In doing so it looks at providing further answers to RQ 4 and RQ 5.

Chapter 7 discusses the research findings in light of the proposed research questions. It looks at how these findings contribute to the existing knowledge of how to teach computer programming, existing beliefs regarding the role of CT in this and the implications of this research. Chapter 7 also concludes in revisiting the research questions and summarizing the findings of this research. It discusses potential future work related to RQ 6 – addressing its uptake in practice in Malaysia for training teachers to be ready to teach programming and presents the specific contributions of this PhD. The expected overall contribution is detailed in Section 1.4.

1.4 EXPECTED CONTRIBUTIONS

Teaching non-programmers how to think computationally will be of immense value in education. It is expected that this work will demonstrate how to aid and even motivate the acquisition of CT skills and that this will offer the opportunity to challenge and change teachers' thinking. Ultimately, it is expected that if thinking regarding programming is changed then education will be more powerful in the field of technology and the consequential development of CT in users will additionally benefit other areas of life especially for Malaysian primary school teachers.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Chapter 1 established that the motivator for this research arose from work undertaken in Malaysia to teach pre-service teachers how to use IT tools. It also introduced Computational thinking as an important part of the process of learning programming. However, it did acknowledge that as proposed by Wing (2006) there is potential benefit across other forms of learning and problem solving for applying some of the CT central skills. This chapter will begin by looking at the background to understanding computational thinking. It will continue to investigate current practices in trying to promote the teaching of computing in schools. As was indicated in the introduction the main interest of this work focuses on specific issues in promoting this in the Malaysian context but as it already states a lot can be learned from what is already being undertaken in other places.

Teaching computer programming rather than "IT skills" has become a focus in recent changes made to the UK curriculum (Department for Education 2013). However, students being taught programming, and teachers being taught to teach programming, do not always appreciate the importance of Computational Thinking (CT) in this process (Department for Education 2013). As was introduced in the first chapter, while the concept of CT has wider applications in learning than just in programming it is still a cornerstone of learning to program.

This chapter presents a review, analysis and classification of related literature in order to begin to answer the research questions posed in Chapter 1. The rest of Chapter 2 is split into three broad sections. Section 2.2 begins by introducing CT and reviewing the literature on CT to provide the background to this research. The concept of CT is investigated in terms of the individual skills that make up CT in order to understand how to detect CT skills and how to start changing thinking strategies to be more

computational (which relates to Research question 3). This section also introduces organizations across the world that have been promoting wider participation in computer programming and a consequent understanding of CT.

Having introduced CT, the following section (Section 2.3) examines existing tools and platforms used to teach programming and CT and therefore allows us to start to address Research Questions 1 and 2, which ask about the readiness of teachers to teach programming and their ability to develop CT skills. Section 2.4 examines methods for assessing CT which enables Research Question 3 to be addressed. Section 2.5 then moves on to review literature on motivation in order to further address Research Question 2 in terms of understanding how to motivate teachers to develop CT skills.

2.2 COMPUTATIONAL THINKING BACKGROUND

2.2.1 Introduction

Programming is a very practical topic that is ideally learnt by 'doing' rather than 'memorizing'. However, programming is more easily learnt by some people than others as being able to program is greatly aided by the ability to think computationally (Huang, Deng, and Rongsheng 2009). For example, in programming, it is not appropriate just to copy and paste code that solves one problem to solve another without understanding the logic of the syntax (E. Saari, P.Blanchfield, & G.Hopkins, 2015). Therefore the method of teaching programming should facilitate a better understanding of Computational Thinking first rather than 'plunging headlong' into coding as a first step. (L. Lezam et al., 2015).

Computational thinking is not about programming alone (Yeh et al., 2011), rather it is the development of models which represent problems to be solved (Lee et al., 2011). These can be computer programming-based problems or problems experienced in everyday life. CT can enhance creative ability by enabling technology to be shaped and not just used (Mishra and Yadav 2013) but it is widely acknowledged to be fundamental throughout life (J. M. Wing, 2006; National Research Council, 2010).

2.2.2 Defining Computational Thinking

There is little agreement about exactly what CT encompasses and even less agreement about strategies for assessing the development of CT in young people (Brennan and Resnick 2012). However, the essence of CT involves breaking down complex problems into more manageable sub-problems and using a sequence of steps to solve these sub-problems. Without agreement on a common definition of computational thinking, it will be difficult, if not impossible, to develop appropriate assessment tools that actually measure the ability to think computationally (National Research Council, 2010). Thus at this point CT can be seen to be useful for tackling problems in any situation. "We are also at the point where we can determine if a computer can help us more efficiently solve those problems" (Wing 2011; Curzon et al. 2009; Grover & Pea 2013a).

The efficient resolution of problems is thus core to CT (Huang, Deng, and Rongsheng 2009; Brennan and Resnick 2012), involving problem-solving and thinking skills (L'Heureux and Boisvert 2012) which then enable programmers to articulate that problem-solving using the programming language of their choice, but which can also be applied in many other domains like algorithmic thinking and abstraction.

The process of CT therefore involves abstracting problems and devising solutions that can be transferred and applied across subjects (V. Barr and Stephenson 2011). When students face problems CT allows them to ask questions based on what they have learned, propose algorithms and ultimately perform procedures to test the effect of the solution (Yinnan and Chaosheng 2012). Adopting CT has a number of benefits, not just for students but also for those teaching the students and this is discussed in the next section.

2.2.3 Adoption of Computational Thinking

Adapting students to a computational way of thinking involves a change of teaching focus from the way the traditional curriculum is taught (Dorling M. 2014). Currently teachers commonly engage students in what they are learning and *how* to learn it and then only subsequently teach them why it is important. Teaching CT, however, involves firstly encouraging students to examine why they need to perform a task or learn a topic so that they ask questions and more fully understand the problem before trying to

solve it. After this they engage in the process of how to solve the problem and then what exactly should be done or used to solve the problem. As stated by Dorling (2014), the CT concepts used to achieve this sit within the "How" stage and classroom techniques can be used to stimulate an understanding of these concepts before the execution that takes place within the "What" stage. This thesis proposes the use of the description of the components of CT adopted by Dorling and presented in a Computing At Schools (CAS) meeting in Birmingham in 2014 (Dorling 2014). (For more on CAS see section 2.x.y below.) His concept presented two parallel processes. The first of these is designed to produce skills and the second to produce knowledge and understanding. In his model these two processes have three common stages. They development of either should start by adopting the "Why" rather than the "How". In both cases this starts with a definition of the aim - being to understand or solve a problem. The next stage or "How" for developing knowledge is the 'thought process'. This will have two stages, understanding the Computational Thinking concepts and then putting them into practice in classroom techniques. The parallel to this in developing skills is to acquire and develop the functional skills (the coding process in programming – acquiring knowledge of syntax and function). The final step is the "What" which concentrates on outcomes. For practical skills this will be the development of the artifact and for knowledge acquisition the "learning outcome assessment framework". (Dorling 2014)

2.2.4 Pedagogical Approaches to Teaching Programming and CT

In considering the "How" stage it is important to examine pedagogical approaches that might be used to teach CT and programming. Carter and Boyle (2002) highlight the difficulties experienced when teaching programming whereby learners, whatever their level of intelligence, find the development of programming skills confusing and difficult. This is not a new phenomenon (T. Jenkins, 2002; Martins et al., 2010; Radošević et al., 2009).

The appropriate approach needs the instructor to keep early facts, models, and rules simple and only develop and purify them as the student obtains experience (Leon E. Winslow 1996). Carter and Boyle (2002) stated in their articles about the difficulty of teaching programming, "there is a perennial problem in teaching this essential skill to

people who find it genuinely confusing and difficult, whatever their intelligence" (p.84). In addition, motivation is a main problem among graduates and its deficiency rapidly undermines many methods in teaching (Carter and Boyle 2002).

There are a number of pedagogical approaches that could be used to teach primary school programming. These include self-directed learning, collaborative learning, enquiry-based learning, project-based learning and explorative learning. The first two approaches contrast each other (Pedersen 2003; Gomez, Wu, and Passerini 2010; Ng 2012), in that one involves the student themselves driving their own learning forward, whereas the second approach involves students working together and thus enables them to scaffold each other's learning. Jenkins (2001) stated that programming is a difficult skill to obtain thus "it is best learned by practice and, if students are to learn effectively, at least some of this practice have [sic] to be self-directed" (p.4). Therefore, the instructor plays an important role determining how to motivate students to engage in learning how to program properly. Discovering an interest in programming can lead to the development of intrinsic motivation to learn more, but the wrong teaching styles can result in lost motivation (Jenkins 2001). In today's "knowledge economy", students need to be self-directed and possess lifelong learning skills. They are expected to retain "21st century skills", such as creativity and critical thinking, problem solving, and analytical reasoning in their learning (Wagner 2008). Students can no longer survive by memorizing textbooks; they now need to explore and experience authentic tasks that can be connected to the real world, in which they can develop, master, and demonstrate authentic skills (Krishnan & Muhammad Yassin, 2009). In this study, the proposed tool will help pre-service and in-service teachers teach their students to become successful self-directed learners. The appropriate tool is designed to motivate users to learn Flash Action Script programming in more attractive way by adapting CT domains (the details about the tool will be explained in chapter 6).

This latter approach benefits from the discourse that takes place between students, thus strengthening their ideas and knowledge. These approaches can be combined with **project-based learning (PBL)** (Parberry, Kazemzadeh, and Roden 2006) which is designed to present the learner with longer-term tasks, often involving real problems or environments. Project-based learning (PBL) is a learning approach that organizes

students' learning activities around projects (Köse 2010). Students are required to complete the task and, in doing so, they gain valuable knowledge of how their learned skills relate to real-world problems. Liu and Hsiao (2002) stated that it is an inventive instructional approach that shifts learning focus from "teacher telling" to student-centered "learning by doing". PBL enables students to make interdisciplinary associations between various ideas (Curtis, 2002)

Liu, M., & Hsiao, Y. (2002). Middle school students as multimedia designers: A project-based learning approach. Journal of Interactive Learning Research, 13 (4), 311 - 337.

Curtis, D. (2002). The power of projects. Educational Leadership, 60 (1), 50 - 53

Enquiry-based learning (Gomez, Wu, and Passerini 2010) is a more structured approach which presents the learner with questions or problems to solve. In this way, it is similar to problem-based learning, in that both involve the student finding an answer or solution to a given situation, but with the latter being more student-centred and open-ended whilst appropriately mapping the curriculum (Whitton, 2007; Merah et al., 2017).

Explorative learning focuses more on an investigative approach and discovering and understanding relationships between a learner's existing knowledge and new material and concepts (de Freitas, 2006). This has been an approach commonly believed in since the early days of general programming (Knuth 1975).

Teaching CT and programming therefore has a number of approaches open to it, but ideally requires learners to be actively involved in trying out solutions to problems and experimenting in order to see cause and effect. Each of the pedagogical approaches described has some merit, but those which involve problem-solving and hands-on trial and error present the learner with the best opportunities to understand and develop programming and CT skills. Since learning to program has been found to present a number of challenges to learners then shorter-term goals and problems are more suitable in the early stages with these being built upon as the learner progresses. The scope for trying out solutions to problems, however, is limited in Malaysian education institutions due to the sparse provision of technological resources. Whilst this situation

is currently being addressed by the Malaysian Ministry of Education this means that progress in teaching programming and CT skills has been restricted in the country. The Malaysian Ministry of Education states that problem-based learning should be used to teach the programming curriculum. However, irrespective of pedagogy, the "limitations in expertise of those who teach programming at primary and secondary level compounds the problem" *(Department of Education MOE, 2016).* This reiterates the need for programming and CT skills to be taught to teachers before they embark on their teaching careers and the challenges involved in learning these skills call for different approaches to be explored, which has been restricted until now. It will also require levels of in-service training for those already in teaching if the ambitions of the Department of Education are to be met in the near future.

2.2.5 CT in the primary curriculum

The need for the Malaysian curriculum to include the teaching of Higher Order Thinking skills (HOT), which encompass CT skills, has been acknowledged. Resnick (2010) (as cited in Zohar, 2013), referred to "scaling up the thinking curriculum" and presented this as a major educational challenge if students are to become proficient in HOTS. Teachers agree that HOTS are crucial to guide the "idea generation" (Yee, et al. 2012). However, a report by the Malaysian Ministry of Education (MOE), the Malaysia Education Blueprint 2013-2025, has shown that there are insufficiencies in most lessons with respect to engaging students in such skills. It was reported that teachers have tended to rely more on didactic teaching rather than encouraging higher order thinking skills (Malaysia MOE, 2012). Until very recently teaching methods and learning outcomes in the Malaysian education system have been geared towards lower-order thinking skills due to the fact that educators and students view being given a direct answer as quicker and easier, rather than having to "think out of the box" (S. Y. Tan and Halili 2015). Additionally there is evidence that many teachers in Malaysia are confused about CT and how it relates to the use of computers/ICT in the classroom (Ling et al., 2017). This is due to the fact that few teachers have had the opportunity to engage in training or workshops related to CT in Malaysia (Singh 2016; U. L. Ling et al. 2017). Without the correct and appropriate scaffold in CT, teachers will not be able to

implement the newly revised syllabus and incorporate CT into their teaching and learning (Abas 2016).

In January 2017, computational thinking was made part of the reviewed Standard Curriculum for Primary Schools (KSSR) for Year One pupils – making Malaysia the first country in Asean to introduce it into the national syllabus (https://www.thestar.com.my/news/nation/2017/07/29). The National Union of the Teaching Profession president Hashim Adnan mentioned the importance of equipping the students with digital literacy, especially coding skills. He added that the implementation of the coding curriculum that should have been started in January 2017 would not succeed due to the poor preparation among teachers at primary or secondary schools (BERNAMA, 2016). He added that even if they started training teachers at that time, it would create a disruption in classrooms because of the need to train so many teachers in such a short period of time.

It is clear, however, that the development of CT can enable significant improvements in problem-solving as students begin to think in new ways (Yadav et al., 2014) and indeed the benefits of CT are multiple (Saari et al., 2015, 2016). CT promotes creative thinking (Resnick 2007, Lee et al. 2011) and reinforces and extends HOTS (Chang et al., 2012; Dorling M., 2014). It can thus be transferred to and applied within other disciplines (Bundy 2007) and the teaching of CT is now widely recognized as fundamental for all:

“A high quality computing education equips pupils to use computational thinking and creativity to understand and change the world” (UK Department of Education 2014, p.217).

Isabell and colleagues (2010) identified the secondary and post-secondary phases of education as particular targets for the teaching of CT skills. However, to ensure that students develop the ability to think computationally, it is important to provide teachers with an adequate knowledge of CT and how to incorporate it into their teaching. Teachers take on roles to facilitate and inspire creativity and learning (Lin 2012; E. M. Saari 2014; E. M. Saari, Blanchfield, and Hopk The behaviors of the students will definitely change once they discover an interest in programming in 2015, 2016) and to do this they should engage students in the exploration of real-world issues using

different resources to stimulate the understanding of problems (L. Wang, Ertmer, and Newby 2004; Lin 2012). There have been different approaches to this used in various countries and the next section examines some of these in order to gain insight into what might be effective in the Malaysian context. 2.2.6 Evidence from methods and motivations for learning CT from other countries

As stated in chapter 1 the main motivation for the work in this thesis is to focus on the situation and needs of Malaysia. However, it was also stated that experience in other countries (especially those who have been developing their computational thinking training for longer) will provide useful examples for how Malaysia should proceed.

Several western countries are in the middle of changes to school curricula as they see the value of introducing topics such as programming, CS and computational thinking. For example changes in both the UK and Australian curricula were introduced from 2014/2015 and include introductory programming experiences in primary schools (Duncan, Bell, and Tanimoto 2014). A key driver for updating the school program has been articulated by Rushkoff (2010) who was teaching children how to use software to write and not to write programs with the aim that they would become not only users of digital technologies but efficient problem solvers in general.

Computing At School (CAS) is a UK based initiative that involves development of a community of individuals who are enthusiastic about giving children a solid education in computing. It focuses on a professional procedure in which members join as colleagues in a shared endeavour (http://www.computingatschool.org.uk). Its membership is broad and includes teachers, parents, examiners, university faculty, professional societies and industry (Crick and Sentance 2011; Bradshaw and Woollard 2012). The aim of CAS is to share good practice between members in order to promote ways of teaching computing and support educators in providing the new UK curriculum with its focus on programming.

The drive to improve digital skills in children is gradually being adopted by other countries. In 2015, Microsoft conducted a study among 250 Singapore students to look at interest in coding. Results showed that the majority surveyed (76 percent) viewed coding as very important in their education and consequently future careers. The

positive comments towards coding was equally shared by males (79 percent) and females (74 percent) even though elsewhere coding is often dominated by males.

Singapore has now started the implementation of CT awareness among school children (https://www.techinasia.com/jules-school-of-fish-preschool-computational-thinking).
The Singapore-based company Jules has constructed a curriculum that is intended to promote and teach computational thinking to preschool children. They believe that these students would learn more easily about digital literacy given their young age. Jules founder Jonathan Chan stated that CT is a beneficial and important skill that allows children to become successful problem-solvers in the 21st-century digital economy. The lessons are taught to preschoolers through tablets, feature interactive teacher characters, games, activities, and videos. In March 2017, the Government of Singapore revealed a $3 million programme to distribute 100,000 pocket-size, codeable gadgets to school-going children and adults over the following two years to teach basic coding (http://www.straitstimes.com/singapore/education/pre-schoolers-get-early-start-in-coding-game).

Malaysia has begun working hard to implement a CT curriculum throughout schools at all levels from primary to secondary in order to be in line with its nearest neighbor Singapore. Normally schools and other institutions in Malaysia which teach programming do so using classroom-based teaching practices. However most recently the use of Virtual Learning Environments (VLE) has been used in Malaysia (Chean, Ho, and Chai 2018). The challenge is for Malaysian educators to design and prepare learning content that enables students to learn at their own pace and skill levels (Chean et al., 2018). It has been demonstrated that novice student learners who solely rely on learning content face a number of difficulties in learning a programming language for problem solving. One of the difficulties that often arises for them is that the problems and their solutions are complex. Use of software to visualize the process was found to be a clearer and more effective way of transferring knowledge. Motivation to use it, however, was not found to be good.

The concept of using programming as a tool to develop computational thinking skills, as explained in this study, owes its basis, in part to the holding effort of Denning (2009), Jenkins (2002), McCracken et al. (2001) and Pólya (1985). It is this augmentation to learning with tools and technologies which is discussed in the next section.

2.3 TOOLS AND TECHNOLOGIES FOR TEACHING PROGRAMMING AND CT

Numerous studies have argued that students view computer programming as a purely technical activity rather than a set of combined problem-solving skills (Kazimoglu et al. 2012). Learning CT is often thought of as a "barrier" for learning technology (B. D. Barr, Harrison, and Conery 2011) by those who do not perceive themselves as able to be programmers. Novice programmers may not actually think of the process of interpretation or compilation but to understand how the language is understood by the computer can be at a level of abstraction that is also beyond the novice (Guzdial 2003).

Despite pre-service teachers' views that computer technology can provide useful tools for personal and professional use (Levin et al., 2000) a number of factors have been identified to explain why teachers feel unprepared to use technology in their classrooms including insufficient access to technology, lack of time and lack of skills (Tondeur et al. 2012).

"It has been said that a person does not really understand something until after teaching it to someone else. Actually a person does not really understand something until after teaching it to a computer, i.e. expressing it as an algorithm" (Knuth, 1996).

However technological advancement is providing more pressure on pre-service teachers to improve their teaching expertise in this area (Boulton and Hramiak 2013) and the number of online resources are growing (see, for example, The Barefoot Computing Project (http://barefootcas.org.uk, 2015)). It has already been established that it is important to educate teachers with an adequate knowledge of CT and how to incorporate it into their teaching (Yadav et al. 2014) and research argues that the possession of CT skills is fundamental to this. Use of technology to teach programming

has met with limited success, however, (Takemura 2007) despite the different approaches that have been taken. The following section provides a review of tools and approaches that have been used to teach programming and CT and discusses the merits and disadvantages of each.

2.3.1 Graphical User Interfaces for Programming

One train of thought in teaching programming is to encourage the user to think about and plan the execution of their program without having to worry about syntax; in other words, to begin to think computationally (Sherrell 2010). Graphical User Interface-based programming tools like storytelling Alice (Daily, Leonard, and Jörg 2014) and Scratch (Sáez-López, Román-González, and Vázquez-Cano 2016) are tools designed to teach programming in this way. These tools have been popular in schools (Maloney, Resnick, and Rusk 2010) but their effectiveness still depends on the learner and their background.

2.3.1.1 Storytelling Alice

Storytelling Alice (SA) is aimed at teaching fundamental programming skills such as variables, looping and conditionals. It was originally designed to engage middle school girls aged 12 years and above and develop their digital fluency and CT skills (Daily, Leonard, and Jörg 2014) (see Figure 2.1). Learners construct programs by dragging and dropping code elements, thus removing the possibilities of making syntax errors, whilst the animations allow them to see semantic errors as they occur. Researchers also use Alice as a tool to support the development of algorithmic thinking, problem solving, and event processing (Utting, Cooper, and Kölling 2010; Daily, Leonard, and Jörg 2014). A study conducted by Kelleher et al (2007) used storytelling Alice programming to create a solid gateway to a programming experience that could attract girls into programming (Kelleher, Pausch, and Kiesler 2007). However, they also found that there is a risk that Storytelling Alice may not work as well for boys as girls. The effectiveness of Storytelling Alice has, however, been found to depend on the learner's background in computing. Users with some experience of programming have shown signs of engagement with the tool (Linda et al. 2009) but those with no computing background have been observed to

find the tool difficult to use (Utting, Cooper, and Kölling 2010). Werner, Campe, and Denner (2012) have subsequently emphasised that students can easily learn complex computer science concepts such as abstraction, modelling, control structures and event handlers while using Alice.

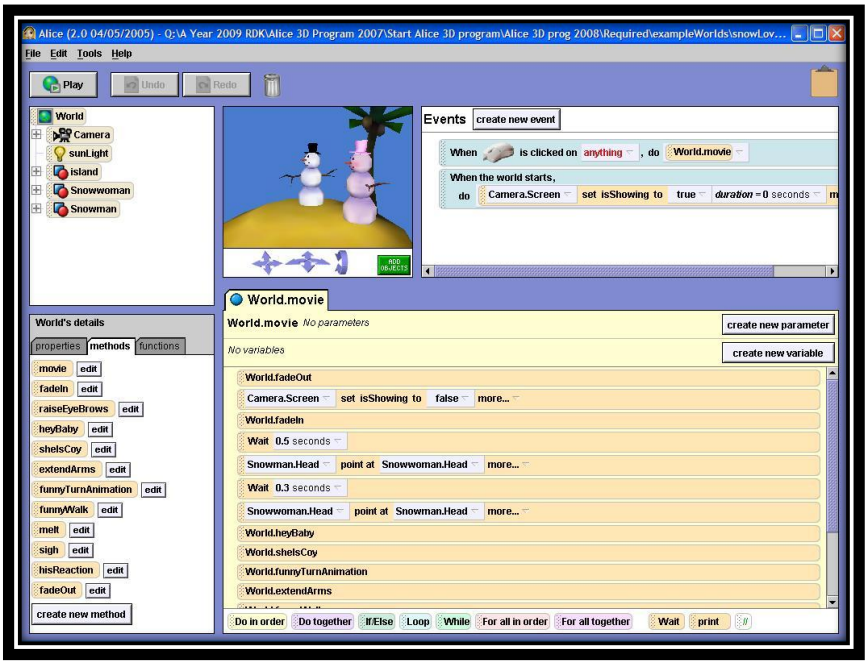


Figure 2.1: *Storytelling Alice Interface.*

*(source:www.**alice.**org/index.php)*

2.3.1.2 Scratch

Scratch is a free programming environment which enables young learners to create interactive stories for themselves and then share them with other online users (Brennan & Resnick, 2012 ; Lye & Koh, 2014) (see Figure 2.2). Scratch has been popular with young people and has been used by those new to programming with good results (Grover & Pea, 2013 ; D. Wang et al., 2014). Learners can build their programs using the graphical blocks provided and create simulations, animations and even games.

Scratch is aimed specifically at young learners and provides a valuable resource in schools (M. Resnick et al. 2009; Brennan and Resnick 2013). Its effectiveness like Storytelling Alice, however, depends on the learner and their learning style. If a learner cannot see the value in what they are learning then motivating them to use the tool can

become a problem (Chang et al. 2012; Y. Lee and Lee 2014). Additionally, Scratch and Storytelling Alice are aimed at teaching children to program and less suitable for engaging adults. In summary, the results of the study found that it is necessary to evaluate their problem solving and logical thinking skills through Scratch and the students had also displayed their creativity through the process. In another study conducted by Sofian Azmi, head of sector, innovation, information and communication technology at the Education Ministry of Malaysia, a Microsoft survey involving 1,850 students from Asia showed a growing interest in coding among students – 88% of students in Malaysia wanted to know more about coding and 68% wished it would be offered as a core subject in schools (BERNAMA, 2015). As a result of this initiative teachers will undergo a three month online course to teach coding in Scratch because the government will not incur costly licensing fees as it is free.

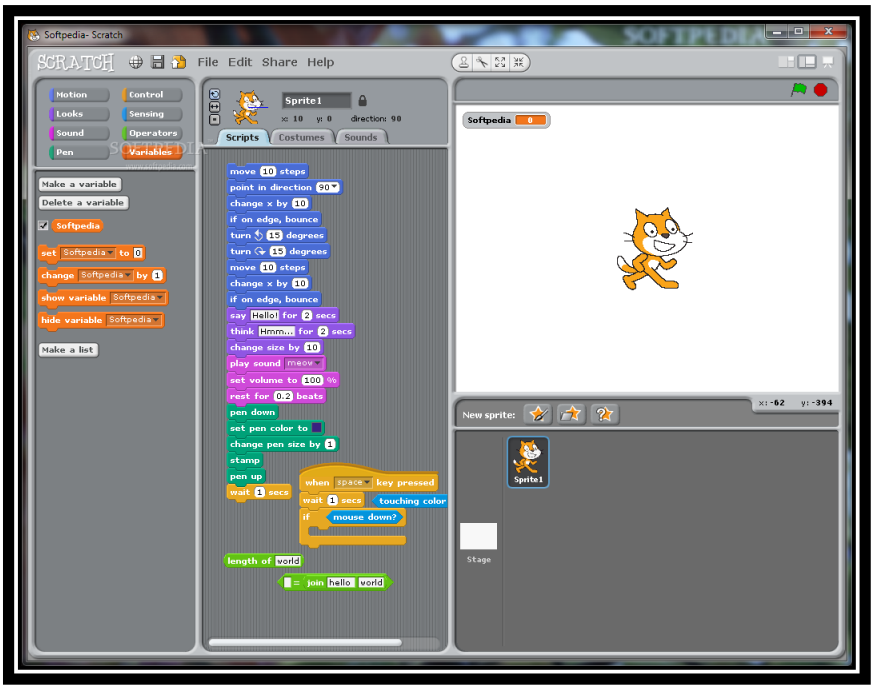


Figure 2.2: *Scratch Interface*

(source: ***scratch****.mit.edu)*

2.3.1.3 Text-based Programming Language - Python

Functions in Alice are supported through the underlying Python language. In Alice, functions are primarily used in the implementation of recursion/looping, and in the implementation of interactions via events (Pausch & Dann, 2000). Python is a freely available open source high level computer programming language, developed at the end of the 80's in the Netherlands, by Guido Van Rossun, and was published for the first time in 1991. Python has some characteristics that make it a good language for learning programming: its syntax is clear and objective and its dynamic typing makes variables automatically determined at execution time (Begosso 2012). Python is also used to introduce coding concepts with older students. Research conducted in Finland by Begosso (2012), emphasized the use of Python to introduce programming concepts and the results were very motivating: high school students reported feelings of success when writing and reusing structured programs they worked with. The language was adopted to strengthen object oriented concepts. But the students were all from a computer science course, in this study Python was not considered suitable because the level of language used is high for the non- computing background users (Aiken et al., 2013) .

Aiken et al. (2013) used VPython (http://vpython.org/) a high-level programming environment for a 9th grade conceptual physics course in the USA. Computational assignments followed in-class experiments and problem solving sessions. After instruction, roughly a third of the students were able to successfully complete an individual assessment in which they constructed a model of a new physical system. In addition, the SECANT (Science Education in Computational Thinking) project (http://secant.cs.purdue.edu/) (Ahamed et al., 2010) conducted at Purdue University used Python focusing on computational methods and visualizations. This project collaborated with two high school physics teachers to incorporate selected material of the 23 Matter & Interaction Curriculum with computational thinking principles into high school physics.

StarLogo the Next Generation (Klopfer & Yoon, 2000; Muratet et al., 2009) is an environment which uses similar principles to Scratch in that learners can place graphical

blocks together to create programs. Students are motivated to use the system by the opportunity to design and build computer games. Users can visually learn programming through StarLogo (Busaki, Iijima, and Konno 2014). The use of 3D graphics and agent-based programming allows them to create simulations and observe complex systems. The tool also allows them to develop CT skills such as algorithmic thinking, generalisation and evaluation since they have to select and put together blocks providing functionalities which build on other functionalities. However, there are some restrictions to the use of Starlogo because programmers must understand the unique languages of the systems as well as structures and the concept of agents (Busaki, Iijima, and Konno 2014).

Encouraging engagement with an environment is key to its success (Maloney, Resnick, and Rusk 2010; Brennan and Resnick 2013) and games are one such approach to achieving this with users with different ability levels. Games that are designed to teach are discussed in the next section. Further tools which use agent-based programming and robots are discussed in Section 2.3.3.

2.3.2 Serious Games

Serious games are games in which "a mental contest, played with a computer in accordance with specific rules that uses entertainment to further government or corporate training, education, health, public policy and strategic communication objectives." (Zyda 2005). They have been used with varying degrees of success in teaching environments and much research has gone into the effectiveness of these games for learning. In order to be effective the game must firstly encode the learning outcome in the game mechanic (Habgood 2010) and secondly, there needs to be the motivation to learn through games (Vs, Building, and Hamzah 2011). The development of games therefore takes time, attention to detail, and can be costly (Bond 2010).

The National Science Foundation is an organisation which has funded projects in areas such as the Innovative Technology Experience for Students and Teachers (ITEST), Academics and Young Scientists (AYS) and Research and Evaluation on Education in Science and Engineering (REESE)(I. Lee et al. 2011). As part of this, research has been

carried out into the design of games to teach programming and associated skills (Kazimoglu et al. 2012).

iGame is an after-school program funded under this initiative with the aim to engage students in CT. In this program students used Storytelling Alice to design and build their own games (I. Lee et al. 2011). As discussed above, Storytelling Alice allows the relatively easy creation of 3D animations and therefore allows students to develop and test complex abstractions quickly and precisely (Kelleher, Pausch, and Kiesler 2007). Students used the tool not only to design and build games but also to define their own methods to create new behaviours within the game using existing behaviours built into Storytelling Alice. This process involved students creating their own abstractions. They were then able to analyse whether or not their abstraction was correct and efficient by observing whether the game was played by others in the way it was intended. The code could then be edited to adapt the game or correct any semantic errors.

Students also play-tested their peers' games and analysed them with respect to playability, engagement and fun as well as being able to test the quality of abstractions created. However, whilst the purpose was to build an understanding of CT it was observed that most game programmers did not use the environment as intended. The intention was that they would scale up their abstractions and use programming constructs to allow these abstractions to be used in different situations, but instead students were observed to just repeat commands that they understood, thus demonstrating a lack of CT skills.

Prog&Play (Muratet et al. 2012) is aimed at first year undergraduates learning the basics of programming. The tool is designed to encourage students to formulate strategies for a real-time strategy game and evaluate the effectiveness of those strategies on the game outcome. Students write programs (instructions) to manipulate game units according to their strategy. The tool is designed to teach CT skills and the use of game design is considered as a motivating factor for students. The tool has undergone several iterations, a later one of which was designed to enable teachers to build the game themselves for their courses. Teachers' initial reports indicated that they were encouraged to continue using the tool. There was a lack of evidence,

however, of teachers using the tool with the strict intention of learning programming or CT skills themselves.

A study in Malaysia examined the game-based approach technique used to motivate students to learn object-oriented programming (Wong, Yatim, and Tan 2016). Odyssey of Phoenix is a computer game that allows players to learn object-oriented programming concepts and paradigms in an easy and motivated environment. The limitation of this game is that the Odyssey of Phoenix is a stand-alone game in which each player plays and learns in their own time. The game helps users to understand the concept of object-oriented programming. The gaming part aims to increase and nurture the users' curiosity in learning the knowledge. The study took place in one of the private colleges in Kuala Lumpur Malaysia. A group of 35 students in the first year of Bachelor of Game Development were offered to test the game. The results from the test showed the overall average of students who strongly agreed with the Odyssey of Phoenix being effective was 30% and those who agreed 55%. Even though the results showed positive output of the study, however not all game genres are suitable to be developed as learning and teaching tools (Rajaravivarma 2005; Muratet et al. 2012) and the Odyssey game was limited to teaching the object-oriented programming subject only and not to other programming subjects.

2.3.3 Robotics and Agents

Programming robots can be an effective way of learning CT skills (Kazimoglu et al. 2012; Catlin and Woollard 2014) because behaviours need to be programmed and the programmer needs to understand logic, reason and plan those behaviours. Numerous tools and simulations have been built over the years which use Artificial Intelligence (AI)-based robots to explore environments or play games and for response planning which uses CT skills (Catlin and Woollard 2014). Thus this section examines how Robotics has been used to teach CT skills and programming.

Logo (Pea 1983; Kahn 1996; Mayer 2004) was a simple, very early scripting language, the purpose of which was to teach programming concepts. It is perhaps more well-known for the application known as Turtle Graphics (Mayer 2004) which first used Logo to instruct a "turtle" to draw lines on a computer screen display. Logo has been used in

schools to teach pupils to break down desired behaviours into individual instructions (decomposition) which can then be input (algorithmic thinking) and evaluated against what was planned (evaluation). Many subsequent versions of Turtle Graphics have been developed (C. Jenkins 2015), including more sophisticated implementations using tangible objects on a table top which are used to program the turtle (see for example TurTan, discussed in Section 2.3.4 (Gallardo et al., 2008)).

Wumpus World (Stansfield, Carr, & Coldstein, 1976; Sardina & Vassos,2005) is a game which involves an agent exploring a virtual world grid in order to collect gold bars without being caught by the "wumpus". The original world used a text-based representation and the agents moved about the world and responded to various "precepts" or stimuli in the virtual environment. The game therefore involved logic and reasoning and players could program agent behaviours using AI scripting or programing languages. It proved a popular game and versions of Wumpus World, with increased sophistication, have been implemented in various languages over the years since its initial creation in the early 1970s (Stansfield, Carr, and Coldstein 1976).

WISE (Wireless Intelligent Simulation Environment) (Cook et al. 2004) is one such implementation that builds on Wumpus World by modelling a real physical environment through which physical robots explore and respond to real stimuli. WISE was designed to teach and strengthen understanding in Computer Science areas such as AI and wireless mobile networking through simulation. The tool was not designed specifically to teach programming, but to be used in laboratory exercises where students extended or tested simulations.

Wumpus World, WISE and similar agent-based games strengthen CT skills through the need to reason, evaluate and plan behaviours (Sardina and Vassos 2005). However, the need to program behaviours can prove challenging for those who don't have a programming background. For example players with a Social Science background were observed to struggle with understanding scripting and programming the behaviours for WISE (Sardina and Vassos 2005) resulting in a reduced ability to learn CT skills.

Robocode (Muratet et al. 2009) is a programming game through which players program the AI of a robot which is used to play battle tanks. It has been used to teach

programming (Java, in particular) to students at schools and universities but again is not suitable for novices who have no background in programming at all. As with other agent-based games, it does, however, strengthen skills in algorithmic thinking and problem solving through the need to plan tactics and behaviours of the robots.

Middle and high school students aged sixteen years old or younger in Colorado, USA, took part in the iRobot project (in which programs comprise of chains of wooden blocks shaped like jigsaw puzzle pieces) in order to learn to program robot behaviour. The students had to consider real world input (sensors) and output (actuators) and design how the robots interacted within the physical environment. They then evaluated the robot's behaviour to determine whether their implementation was correct (I. Lee et al. 2011).

In Malaysia, teaching CT is still a relatively new development, formally introduced at the time of writing, in 2017, despite the fact that learning CT through robotics has been practiced at selected schools prior to this. These selected schools (*Sekolah Bestari*) have been using robotics to teach students since 2003. However, cost has played a large part in the success of this and has meant that implementing more widespread teaching of CT has been impaired. Teachers or schools that have adequate facilities (e.g. computer labs) are normally in urban areas or have been selected as high-performing schools and have been given the provisions to teach CT using the latest technology (Abas, 2016; Azlee, 2016). There are certain initiatives for teaching CT such as PERMATA pintar Negara, a holiday club subsidised by the Government of Malaysia, which provides an enrichment program for young, gifted children to learn science and technology using robotics (Ramli, Yunus, and Ishak 2010). During the experiment, students use LEGO© Mindstorms NXT bricks to enhance their creativity and to learn engineering skills and computer programming (see Figure 2.3). LEGO© Mindstorms NXT has been used successfully to motivate and teach robotics and programming, whilst being inexpensive (Ramli, Yunus, and Ishak 2010). The approach is still in its early stages and is being explored further (Mariappan, Sing, and Nadarajan 2015).

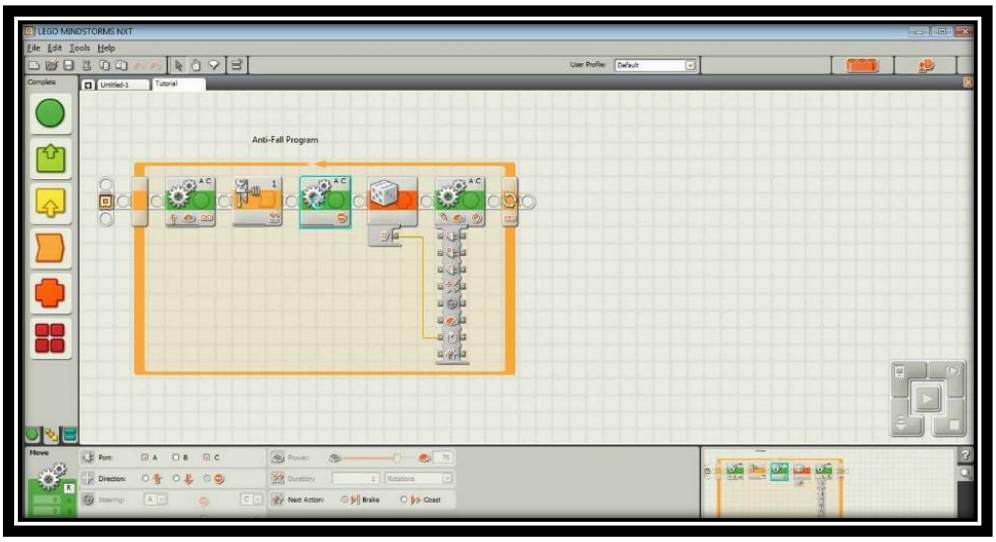


Figure 2.3: *Lego© NXT Programming.*

A study conducted in a school in Johor, Malaysia involved groups of students engaging in educational digital game activities using robologic (K. Tommy, 2014). In this study the content was related to programming and appropriate to the respondents' backgrounds. The students were all upper secondary school. The results from the experiment showed that more than 85% of agreed that learning using digital games was fun and interesting (L. Lehat et al., 2015). Moreover, 75% of the respondents strongly agreed that that digital game helped them to think critically. From the test, 87.5% said the "Robologic Games" made the problem solving process more interesting and 89.5% found it challenged their understanding of the subject.

As has been discussed, robots have been programmed by writing code but other methods have been developed which take tangible objects and combine them using a similar idea to Scratch. This again, abstracts from the complexities of programming syntax whilst enabling users to learn CT skills. Tangible programming has been used to control robots (D. Wang, Wang, and Liu 2014; Kazimoglu et al. 2012) but also to teach programming and CT skills in general. This is discussed in the next section.

2.3.4 Tangible Programming

It has already been established that one approach to teaching programming to novices is to abstract from the syntax and allow the learner to use higher order thinking skills

(CT skills) to formulate and put together a sequence of objects representing functionality. Section 2.3.1 discussed this in particular and presented GUI-based tools such as Scratch and Storytelling Alice which use this principle.

Tangible programming makes use of this approach but uses real physical objects rather than GUI-based objects so that learners can physically grasp those objects and place them together to create a meaningful program (Horn et al. 2009). Tangible objects therefore involve electronic components including sensors, actuators and processors. The idea is that the objects represent some meaning or function (in this case programming constructs) and there can be interaction between the objects if they are placed together or in particular places resulting in some behaviour or output. The advantage of tangible objects is that they can appeal to the way young children learn due to the fact that they use active manipulation of real objects (Tarkan et al. 2010).

Electronic Blocks (Wyeth and Wyeth 2001) are Lego Duplo Primo^{TM} blocks that have been designed to allow young children (aged 3–8) to use their existing skills of construction with the aim of teaching programming skills. The blocks have been adapted by incorporating electronics inside them so that each block provides a particular functionality. This might be an action, such as lighting up or emitting a sound, sensing the environment, such as detecting sound or touch, or providing some logic to adapt the behaviours of other blocks. Children can combine blocks and the output of one block feeds into the input of another block. This enables the children to plan outcomes, in a similar vein to designing a program, and experience interesting but sometimes unexpected results. Initial results indicated that the blocks were popular with both children and teachers alike and children were observed to create stacks of blocks to perform certain functions and to "debug" those stacks when not working. They also reused working combinations of blocks in a way akin to code reuse. There were a few children who did not understand the functionalities of the blocks and others understood how to use the simple logic blocks but not how these might be built upon further to extend to more complex scenarios.

TurTan (Gallardo et al., 2008) is based on Logo and involves manipulating tangible objects on a table top interface to perform Turtle Graphics. Tangible objects represent

programming instructions and the aim is to enable learners to use these to create and build functionality and therefore learn programming concepts (see Figure 2.4). Whilst the expense is still an issue, table top computing offers strong affordances for collaboration (Gallardo, Julià, and Jorda 2008) and so can enable scaffolding which promotes learning.

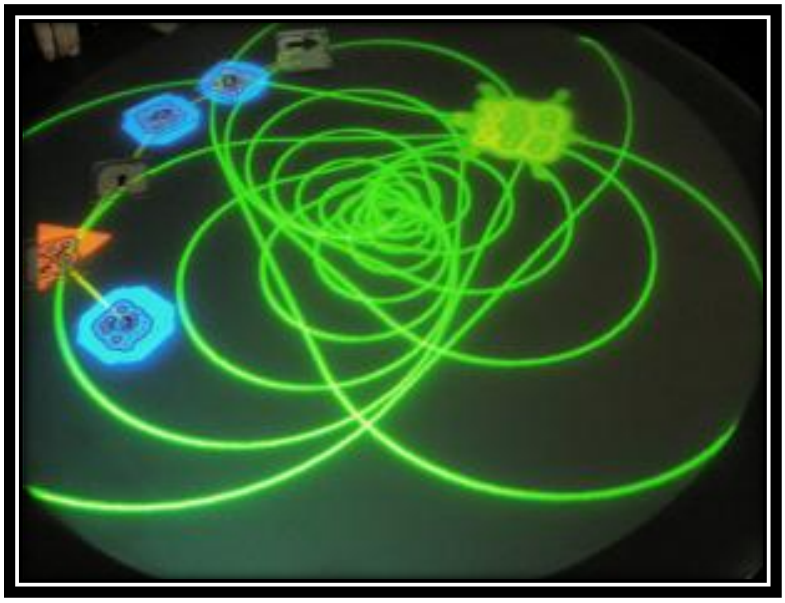


Figure 2.4: *Tur Tan Interface Design*

(Source from (Gallardo, Julià, and Jorda 2008)*)*

Plenty of tools have been built which have the potential to develop programming and CT skills in children, see also, for example, Tern (Horn & Jacob 2007; Horn et al. 2009) Toque (Tarkan et al. 2010) and T-Maze (D. Wang, Wang, and Liu 2014). However research is limited in terms of assessing the effectiveness of these tools for nurturing CT skills (Grover and Pea 2013). A disadvantage of tangible tools is the expense due to the electronic circuitry involved, which means they are not easily available for children in developing countries. Table top interfaces are particularly expensive and unwieldy and so even less suitable, although, as mentioned, the collaboration they afford is valuable in the learning process (Gallardo, Julià, and Jorda 2008; Ahn et al. 2014). Tools which encourage collaboration through conventional desktop interfaces are cheaper and are discussed in Section 2.3.5.

2.3.5 Collaborative Learning Approaches

Collaboration between peers has been widely acknowledged as a powerful teaching aid as it encourages peers to explain concepts to each other and therefore for scaffolding to take place (Vizcaíno et al., 2000; Meluso et al., 2012).

One such system that takes the collaborative approach is Cleogo (Cockburn and Bryant 1998; Muratet et al. 2009) through which several learners can collaborate in the development of programs to solve problems. Learners can take different approaches to do so according to their preference and skill level and the use of 3 different programming paradigms are provided, between which they can mix and match. For example, learners can program a turtle by entering text-based (Logo) commands, clicking icons or dragging and dropping the turtle and they can collaborate and work together on this, with different learners adding behaviours using different programming paradigms. Actions initiated by a learner in any one programming paradigm, are displayed simultaneously to learners using the other two paradigms. Therefore a learner who uses direct manipulation is able to see the corresponding text-based commands that would cause the same effect. The tool therefore engages learners with others in their group and their collaboration provides the opportunity for them to explain to and learn from each other. The interface, seen in Figure 2.5, is very bright and stimulating as it is intended to appeal to younger learners and not *pre-service teachers.*

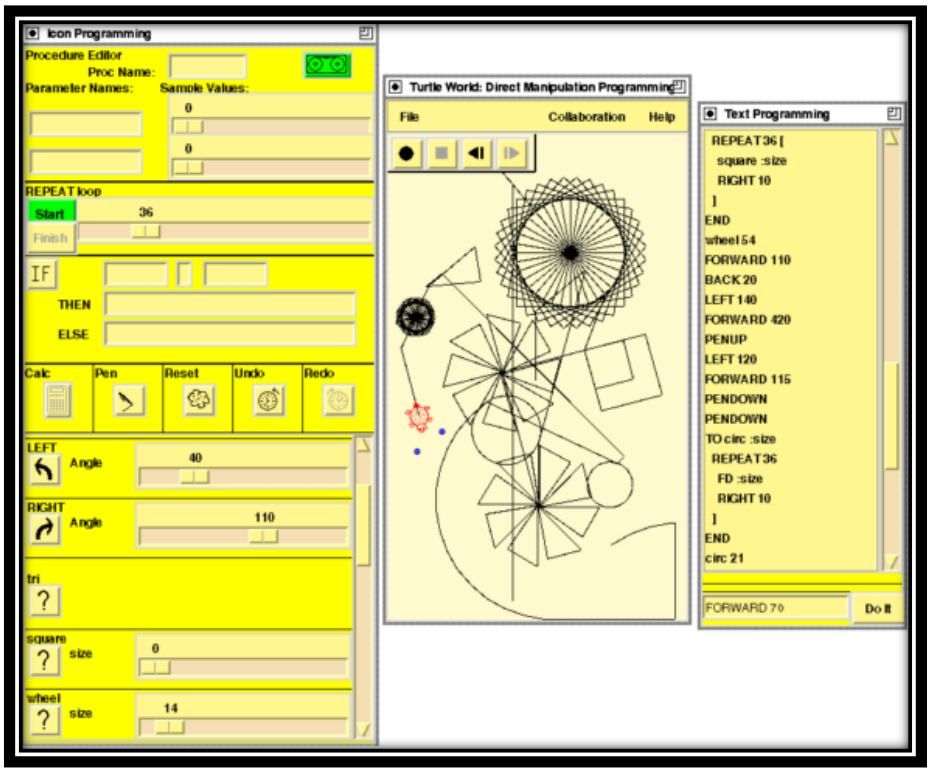


Figure 2.5: *Cleogo Screen Layout*

(source: Cockburn & Bryant 1998*)*

Col-E was developed to teach the early stages of programming to novices (Phuong et al., 2009) with the aim of engaging online peers with others who may be more knowledgeable, but also motivating them to overcome their programming difficulties. It aims to do this using a voting mechanism and the award of points for contributing. After preliminary sessions using the tool, the points awarded by learners to their peers and the grades awarded for their programming by the teacher were used to group students who then continued to use the tool in their new groupings. This enabled learners to be matched with others who could complement their learning.

Teachers also benefited from this approach since, whilst the learners were scaffolding each other, teachers could concentrate on supervising more complex learning. The aim was to help tackle demotivation of students that can occur in a learning environment with an already stretched teacher.

What the tool did not target specifically was the teaching of CT skills in a way which encouraged higher order thinking. Students used the tool to learn to program but this

did not mean they were able to transfer their understanding to tackling other problems. In other words, it could not be assumed that they learned to generalise using the tool. Additionally, there were limitations to the study since the tool was evaluated in a learning environment where several members of staff (Teaching Assistants) were present and so pupils were already well supported. The authors acknowledged that this environment did not provide the programming conditions that would be experienced in a more typical situation and that the tool needed to be evaluated further.

HabiPro (Vizcaíno et al. 2000) was an adaptive collaborative system which was designed to teach good programming practices, but not specifically to teach programming. It did this by developing skills needed to become a good programmer, namely observation, reflection and structure; in other words skills which can align to CT. The tool presented exercises to learners but adapted what it did according to the behaviours of the learners. It built up a model of the group which characterised it in terms of abilities, preferences for types of exercise and mistakes made, as well as motivation (determined by the type of help selected by the group) and participation. Different versions of the system were developed with different floor control mechanisms (no floor control through to turn taking and requesting the floor). A disadvantage of not having floor control was observed in that some students wrote answers to the problems in the chat window without a discussion with other group members, thus removing some of the mechanisms which could enable knowledge construction. Another issue was group size; it was found that if a group had more than 3 members then communication and negotiation between the members became difficult and collaboration decreased.

This section has presented a number of different types of tools which have been designed to teach programming and CT skills. Whilst the tools have provided different benefits to learners, such as interesting and engaging interfaces, collaboration and game-play, a significant limitation is that most of the research into these has concentrated on teaching children and not adults such as pre-service teachers. Additionally, there is no strong evidence as to the effectiveness of these tools or approaches. Measuring their effectiveness requires the ability to measure and assess CT skills which can be difficult in studies involving few participants or children. The next

section moves on to a body of research which has looked at how to assess CT skills and how to measure them quantitatively.

2.4 ASSESSING COMPUTATIONAL THINKING

It has been shown that the ability to program (not necessarily code) relies on CT skills but in order to effectively teach programming and CT skills there needs to be a reliable way to test for the presence of these skills. Researchers have recognised the lack of assessment mechanisms that have been developed for this (Werner et al., 2012; Zhong et al., 2015) although over recent years some efforts have been directed at developing assessment approaches and these are presented here. Approaches have taken the form of games and stories, assigning specific tasks designed to use CT skills without needing to program, coding behaviours and video analysis of young children partaking in problem solving. Numerous authors emphasise that knowing how to assess a persons grasp of CT concepts and practices is essential for full and effective integration of CT in education (e.g. Grover et al., 2014). Assessment is a crucial aspect of CT in education that is still underdeveloped (Bocconi et al. 2016). Kolodner (2011) believes that CT is a set of skills that transfers across disciplinary domains (NRC 2011, p. 54). He added that "the CT community needs to identify exactly what is meant by CT to decide what learners should learn and to assess and evaluate what learners know, what they can do and their attitudes and capabilities with respect to CT" (NCR, 2011, pg. 61 - 62). Kolodner (2011, as cited in NRC, 2011) identified three reasons for assessing computational thinking: first is to judge the curriculum and related materials and pedagogy, second is to judge the progress of individuals, for example, for giving grades, and third is to manage instructor training and support. The following sections look at how CT can be assessed in various ways.

2.4.1 Project Portfolio

Frequent use of computational constructs is favoured, but their effects on final artefacts and the relation between final artefacts and pre-defined learning goals are rarely considered (Basu, Kinnebrew, & Biswas, 2014). Use of a Project Portfolio is one of

the methods used in assessing the development of computational thinking in young people who are engaging in design activities with Scratch. Brennan and Resnick (2012) said that this approach involved using a "scrape tool" which is used as an analysis tool. Scrape tool is normally used by Scratchers to learn new blocks, the layout of the scenario is appealing and beneficial for the learner especially for the useful assessment. This method is totally product-oriented, and shows nothing about the process of developing projects, and nothing about the particular computational thinking practices that might have been employed.

2.4.2 Artefact Based

Secondly, Brennan and Resnick (2012) mentioned that another method of assessing CT was by "analysing the artefacts" for example games or models that are developed by students as indications of their CT capabilities (Bocconi et al. 2016). For instance, the students were asked to modify the code of a program so that the objectives will be achieved. Then, they need to troubleshoot the program. They suggest that such a debugging skill could be an efficient way to assess the effectiveness of the learners problem solving in programming skills.

Besides that, authors like Yadav et al. (2015) reported that they made use of "multiple-choice assessments and attendant rubrics" to assess the CT skills of middle school students. Recently a study from the UK Computing At School (CAS) initiative has started a project called "Quantum" on assessing computing in primary and secondary schools by providing a free access online assessment tool that helps teachers to check their students' understanding and support their progress (Bocconi et al. 2016).

2.4.3 Design Scenarios

"Design Scenarios" are another method proposed by Brennan and Resnick (2012) which they used exclusively in classroom settings. There are three main strong points for this method. First, the scenarios can explore different ways of understanding about knowing, such as critiquing, extending, debugging, and remixing, as well as the concepts

and practices. Second, the design scenarios are planned to be used at three waypoints over time, a method that focuses on a developmental approach. Finally, the scenarios highlight "process-in-action", rather than "process-via-memory". However the limitations of design scenarios are that they are time consuming, particularly in the case of the debugging and extension activities. Their conclusion of the three methods of assessment (project portfolio analysis, artefact-based and design scenarios) was that none of the three approaches were particularly effective for understanding changes in computational thinking perspectives (Brennan and Resnick 2012).

2.4.4 The Fairy Assessment

The Fairy Assessment (Werner, Denner, Campe, & Kawamoto, 2012), is a Storytelling Alice program which was designed to assess the ability to think algorithmically and to effectively use abstraction and modelling. An interesting contribution regarding the measurement and the assessment of computational thinking was their attempt to measure the understanding and use of different computational concepts that students utilize to solve problems. Students were set tasks within the flow of a story which required them to use drag and drop techniques to alter behaviours of the story characters. These tasks were designed to fit the CT skills being assessed and the solutions produced were analysed and assigned points according to the level of understanding and the ability to abstract that was evident. The main limitations to this study were that paired programming was used and so assessment of the students in pairs meant that the validity of data from these measurements was affected by the influence of one student on the other and vice versa. It was therefore difficult to ascertain the extent to which the assessment really measured aspects of CT.

2.4.5 Progression of Early Computational Thinking (PECT)

The PECT model is a framework for assessing CT skills in primary school pupils (Grades 1 to 6) (Seiter and Foreman 2013). The paper tried to give answer to the acknowledged lack of effective methods for assessing CT skills in children of this age and a lack of data concerning how CT skills develop in young people. The PECT model is comprised of

three fundamental components of decreasing abstraction, firstly, computational thinking concepts, secondly, design pattern variables and finally, evidence variables which together provide a practical framework by which CT in the primary grades can be observed and analysed (Seiter and Foreman 2013; Weese 2016). The model believes that every student has an underlying proficiency in CT that reveals itself in the student's talent to design and execute programs to complete specific tasks (Seiter and Foreman 2013).

Seiter and Foreman (2013) tested this approach by creating the PECT model for evaluating Scratch programs. PECT would seem to be an "artefact" based approach. The CT concepts analysed by Seiter and Foreman are a subset of the computational thinking concepts proposed by the CSTA (2012).

In a study by Brennan and Resnick (2012a) results from interviews with the Scratch contributors revealed that the existence of a block (Scratch programming language is via visual block-based (VBBL) artefacts) in a project is not verification of learning. Learning progressions have been proposed by Franklin et al. (2016) that illustrate a path that students go through in understanding a particular topic. They added that the programming in VBBLs may need different ideas and skills thus the process of learning becomes more complicated than the linear learning progression model for instance in a text based learning language. In fact, other scholars have made the argument that learning progression models, in general, are too simple to capture the learning process. PECT has been applied for CT assessment and, according to Weese (2016), gives a practical definition and view of CT.

In another study Duncan, Bell and Tanimoto (2014) found that in a Scratch module being studied grade 1 students were able to work with patterns, which in a Scratch program is called Animate Look (changing the look on sprite) and Animate Motion (changing location with possible synchronization). However, repetition skills (collide, user interaction and score) could only be learned by students in grade 4 and above. The conclusion for their experiment was that certain patterns are best suited for certain grades.

2.4.6 Work with First Year CS Students in South Africa

Gouws et al. (2013) used questions taken from the South African Computer Olympiad "Talent Search" papers (http://www.olympiad.org.za/talent-search/) to test the aptitudes of first year Computer Science students. The questions used were designed not to rely on knowledge of any programming language or paradigm and so could be used for students who may have CT skills without yet knowing how to program. Students completed the questions using pen and paper and, by looking at specific answers to problems and their workings the researchers were able to infer the techniques used by the students and the thought processes involved.

Based on the questions used and on a survey of CT definitions in the literature, the authors proposed six classifications of CT skills, namely Processes and Transformations; Models and Abstractions; Patterns and Algorithms; Tools and Resources; Inference and Logic and Evaluations and Improvements. They then scored the students according to these CT skills and examined correlations between this and the students' class tests results. It was found that high scores in the CT test correlated to high marks in the class tests but the results of dependency tests varied. The authors acknowledged that this research was at an early stage and proposed that the work should be conducted across different institutions and through more longitudinal studies.

2.4.7 Video game creation

(Koh et al., 2010) identified several computational thinking patterns that young students abstract and develop during the creation of video-games in a controlled environment; they created an automated tool that analysed the games programmed and represented graphically how far each game had involved the different patterns when compared with a model.

2.4.8 Peer Video

Portelance & Ave (2015) used peer video interviewing of young children to assess their learning of CT skills. Having been taught foundational CT skills using ScratchJr pupils were able to use what they had learned to create animations, stories and games. They then interviewed each other about what they had created and recorded these using iPads. The resulting videos enabled the researchers to gauge the CT skills that were present through the explanations given by the pupils. This technique was deemed to be particularly appropriate for capturing thought processes and assessing these in young children but the study was only exploratory and further work is needed.

2.4.9 CTSiM

Another approach of assessing computational thinking is CT-based science learning that is through using a CTSiM – a Computational Thinking based science learning environment (Basu et al. 2014). In this environment, students first construct a conceptual model and then design a corresponding computational model for a given science phenomenon. The authors designed an independent system of pre-post assessments for science and Computational Thinking and developed vector- distance, effectiveness, and consistency measures to characterize student models. Using these assessments, they showed that students gained significantly on both science and Computational Thinking content in Kinematics and Ecology subjects. A study conducted using CTSiM skills involving a learning by modeling approach (Wilensky, Brady, & Horn, 2014; Sengupta et al, 2013). CTSiM is proposed as an open-ended learning environment (OELE). OELEs have the potential to provide meaningful learning opportunities to students. While working with an OELE, students usually construct solutions to authentic problems (Ningyu ZHANG and Institute 2017). In CTSiM, students use block-structured constructs to model scientific scenarios using an agent- based framework (Wilensky, et al., 2014). The advantages and feasibility of integrating CTSiM in everyday STEM (Science, Technology, Engineering and Mathematics) learning contexts even for teachers with little experience with computer- based learning environments (Ningyu ZHANG and Institute 2017). Finally, the teacher herself gained programming experience with the system. The study helped teachers become more comfortable with

programming. Moreover, CTSiM is effective in helping middle students learn and improve their understanding of CT concepts and skills. In addition, CTSiM fitted well into the science classroom and helped students learn their science content better (Basu et al., 2016).

2.4.10 Dr. Scratch

Dr. Scratch is a tool that had been created to support an educator's evaluation of student programming and assess the development of CT (Bocconi et al. 2016). It executes automatic testing of Scratch programs by perceiving the presence/absence of conditional statements of students' tasks. The CT Score assigned by Dr. Scratch is based on the capability expressed by the developer of using the seven concepts: abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of flow control, user interactivity and data representation. Moreover, the tool provides feedback to learners and assigns a CT score to the analysed projects. The feedback report provided by Dr. Scratch was especially useful for secondary students with an initial medium (developing) CT score (Moreno et.al., 2015). The CT score range awarded was from 0 to 21 points, if the CT level is low it is assumed that the user is a novice programmer and, consequently, the tool will only show basic information of the most important improvements to perform in the code. As the score increases, Dr. Scratch will show more information of the analysed projects. Thus, advanced users receive a feedback report with all available information both in terms of CT skills and bad programming habits. A limitation of this assessment method is that it is a web application and thus requires internet access for it work (http://drscratch.org/). The research group behind Dr. Scratch has developed a computational thinking test (Román- González, 2015a, 2015b) and has showed its strength with respect to other traditional software quality and complexity metrics (Moreno-León, Robles, & Román-González, 2016).

2.4.11 Motion Charts

For it to be possible to assess the use of Computational Thinking by students and judge the validity of the assessments it is, first of all, important that the students engage in the use of computational tools. The various methods of assessment are designed to ensure CT is happening in school STEM classrooms (Weintrop et al. 2014). Each assessment will therefore comprise of interactive tools that focus on the practice of scientific and mathematical analysis. The computational tools used in the assessments must enable students to analyse data with dynamic visualization and explore concepts with computational models. The idea of Motion Charts is designed to measure students' mastery of skills such as manipulating, analysing and visualizing data from the data skills category in the STEM taxonomy. Weintrop et al. (2014) state that the assessment has students investigate a dataset containing life expectancy, average income and population for six countries over the span of 60 years. Using Google Motion Chart, students can see the data in three different formats (bar chart, line chart and bubble chart) and make a selection of data to render and execute the chart for the results.

2.4.12 Climate Change Model

In this assessment method, an agent-based model enables students to explore the fundamental processes behind climate change (Weintrop et al. 2014). The model measures students' system thinking and modelling and simulation skills. The model is powered by NetLogo and allows users to configure the inputs that define features of the model (such as cloud coverage and the amount of CO_2 in the atmosphere). In addition, the model calculates the position and behaviour of each element and students have the ability to experiment with different settings to determine the cause of climate change. The students need to run several trials and clarify the results. Moreover, they must assess the overall validity of the model and identify assumptions built into the model and simplifications it makes. These assessment sets have students use computational tools that let them explore and engage with the concepts in a dynamic way. This method gives a more reliable form of assessment of CT skills as it positions the use of these skills within a computational context which the students can directly

practice. Weintrop et al. (2014) stated that this assessment method supports and measures the computational ideas and practices that are central to CT.

2.4.13 Principled Assessment of Computational Thinking (PACT)

The intention of the design of PACT is to design, develop and validate CT assessment (Gouws L. et al., 2013). In their study the authors use their own CT test to assess the correlations between computational thinking and critical thinking. This testing is aimed at both computer science majors and non-majors, with early results indicating little correlation between the critical thinking and computational thinking scores (Gouws L. et al., 2013). PACT is a proposed research grant that collaborates with Exploring Computer Science (ECS) developers and teachers to design, develop and validate assessment of CT for ECS validation. Applying the ECS curriculum for researching assessments of CT will deliver direct value for ECS teachers and students. In addition, it can transform computing in high school (https://pact.sri.com). PACT offer CS teachers with accessible resources for assessing their students' CT knowledge and skills. The PACT project represents an important part of efforts to fill the computer science pipeline with a more diverse population of students. High-quality assessment tools and resources will lower barriers for adopting and using the ECS curriculum, as well as other CS curricula, and readiness to engage in further learning. The PACT project offers both new and experienced CS teachers with adaptable resources for assessing their students' knowledge and skills in computational thinking (http://pact.sri.com).

2.4.14 Creation of a Framework of Specially designed Tasks

Another approach taken has been the creation of a framework of specially designed tasks (Zhong et al. 2015). Work in this area is still in its infancy and there is no evidence that particular techniques are more appropriate for assessing CT skills than others. Additionally, as has been discussed in 2.4.5, techniques need to be designed that are appropriate for the type and age range of learner being assessed.

The next section moves on to examine the area of motivation and the role this plays in effective learning.

2.5 MOTIVATION

We have seen in Section 2.3 that, whilst there is a range of tools available to teach programming, many of them are either designed for children or require some basic knowledge upon which to build. For adults who are new to the domain, teaching programming can be challenging due to having to build CT and programming skills with a teaching method or tool which is age appropriate, and due to the fact that often there is a fear of computing and programming seen in these learners (Radošević, et al., 2009; Lin, 2012; Tondeur et al., 2012).

The importance of programming knowledge has rapidly increased in education in recent years *(CAS website; June 2016).* The term "digital native" was coined by Prensky in 2001 to refer to young people who were born in the digital era and who he therefore considered to be naturally technologically competent (Prensky, 2001). Whilst there is some dispute in the literature over this black and white division of abilities (Ng 2012), use of technology and coding is becoming increasingly considered as an essential skill in the digital age (B. D. Barr, Harrison, and Conery 2011). However, contrary to the idea of all children being digital natives, some do not adapt to the thinking style needed to be able to program. For example some have difficulties in visualizing the concepts behind programming, while others can build a picture of what a program is doing just by looking at the code (Williams & Kessler, 2000; Resnick,2007; Portelance &Ave,2015). A consequence of this, which adds to the challenges for teachers, is that students develop different learning rates.

Teaching a practical discipline like programming requires the teacher to persuade students to agree to practice by engaging in writing programs (T. Jenkins 2001a). It has been acknowledged that, in particular, it is not easy to motivate non-technical students to learn programming because programming is not a primary interest of theirs (Takemura et al., 2007). Indeed, teaching computer programming to non-computing or social arts users has been found challenging for both teachers and students alike (Li,

Xiao, & You, 2010; Raadt, 2008; Rudder et al., 2007). Boulton (2008) stated that "there needs to be an awareness that younger students of full-time compulsory school age do need training in using e-learning materials and developing independent learning skills". Such training, however, needs to motivate learners and for adults many tools designed to teach programming may not motivate in the same way that they do for children. The next section examines motivation in more detail and discusses types of motivation and how to engender it in teaching environments.

2.5.1 Motivation to learn

Learning must not only be the process of knowledge transfer from the expert to the novice, but it must also include the construction of knowledge by the learner when cooperating within their environment. People have diverse ways of learning. Some learners can adapt better when the knowledge is received visually, verbally or through another sense (Najjar 1996) and this has consequences for the instructional methods used which need to be varied for different learners.

In 1990, Franzoni et al. published a paper in which they described numerous studies on the efficacy of combining multimedia with learning styles in education systems (Franzoni et al., 1990). Interactivity, control of learning pace and novelty are all alternative ways to create better learning environments and computer-based multimedia can be used to improve the level and rate of learning (Najjar 1996).

If the educational standpoint defines what current learning is needed, then the psychological perspective can contribute some understanding on the motivational factors for successful learning. Motivation to learn depends on the learner's capability, the nature of the skill to be learned and the learner's perception of the value and difficulty of learning the skill (Frith 1997). Psychological factors also include the emotional experiences while learning and much of the psychological theory seeks to better understand human thoughts and behaviours to provide a better understanding on how students learn (Poulou 2005).

Motivation is important for students' success (Frith 1997), but it is an abstract concept that is hard to measure. Whilst a person's behaviour can allow others to infer their

motivation it is never possible to be certain (Jenkins 2001b). Motivation has different characteristics depending on the type and these types sit on a continuum along which some researchers believe is motivation to perform specific tasks (R. Ryan and Deci 2000). The following section examines these types of motivation.

2.5.2 Types of Motivation

People may be motivated through their underlying attitudes and the goals that they have. In other words, according to Ryan and Deci, it concerns the "why" of actions (R. Ryan and Deci 2000) and the "why" is significant in terms of the type of motivation that results. The type of motivation that is present in learners has significant implications for their education (R. Ryan and Deci 2000).

The key to inspiring students through truly motivational instruction is to maximize the positive effects of each of the motivational factors. This can be described more in terms of the types of motivation and what influences them:

- Extrinsic – This derives from the desire to perform an action in order to attain some expected reward or avoid some negative outcome (T. Jenkins 2001a) 2001a). Students can perform extrinsically motivated actions with resentment, resistance and disinterest (if the result of not doing so will result in a negative consequence) or with an attitude of willingness that reflects an inner acceptance of the value or utility of a task (R. Ryan and Deci 2000). An extrinsically motivated student may for instance do very little for which there is little perceived reward. In students a primary motivator can be the career and associated rewards that will follow from the successful completion of the course (T. Jenkins 2001b).

- Intrinsic – This derives from an interest in the task or subject (T. Jenkins 2001a). Students with intrinsic motivation would be anticipated to read about the subject, perform more using their own initiative and from their own views on the material they were taught. Intrinsic motivation is therefore a better driver for learning and creativity and is especially important in education (R. Ryan and Deci 2000).

Knowing the difference between intrinsic and extrinsic motivation and what engenders them is an important concern for educators as they cannot just depend on their students having intrinsic motivation to learn. It is not uncommon for students to be given tasks that they do not inherently enjoy or find interesting and so teachers may need to find ways of extrinsically motivating students instead. It is essential, therefore, that teachers use more active approaches to extrinsically motivate their students and giving them choice can enhance this (R. Ryan and Deci 2000). However, as mentioned in Section 2.3, teachers need to have the knowledge to prepare their teaching material using tools and approaches that they feel comfortable with. The issue is therefore not just about motivating students to learn, but motivating teachers to learn to use technology to enhance their teaching methods and this is discussed in the next section.

2.5.3 Motivating In-Service Teachers and Pre-Service Teachers

As discussed, one principle of motivation is to be more dynamic or adaptive in the way that content is presented to learners in order to stimulate curiosity (Frith 1997). This is the most effective approach to teaching students in that students are more likely to be rapidly familiarised with the content, and the provision of examples and then learning exercises enforces this. In order to maintain or raise students' motivation it is also important to allow students to enjoy seeing the final product (Takemura 2007). One train of thought is that use of computer games in learning environments is intrinsically motivating (Prensky 2001). However, as seen in Section 2.3.2, the effectiveness of games for teaching is dependent on game design and this is far from straightforward. Empowering teachers to design and create their own teaching material enables content to be created in a way that suits teachers and learners. Lin (2012) emphasizes that educators can become creative courseware designers and certain environments have been used by educators to do just this.

Adobe Flash and Action script is a programming environment that allows users to, relatively easily construct interactive animations (Routledge 2007) which embed elements such as graphics, audio and video (Miao & Zhong, 2011; Lamb & Johnson, 2006; Sorapure, 2006; Holzinger & Ebner, 2003;). It is frequently used in educational

software (Miao and Zhong 2011) to produce dynamic learning materials that appeals to learners of all ages (Grover and Pea 2013).

Technology in education can afford different teaching approaches. However, if in-service teachers and pre-service teachers enjoy taking these approaches and enjoy using technology in support of this then teaching will be made easier. It has been established that to use technology in teaching ideally requires pre-service teachers to not only learn how to use the technology but to enjoy doing so (Boulton & Hramiak, 2013) and, in turn, motivating pre-service teachers, and particularly those from a non-technical domain is important. The other challenge is convincing students that they are capable of programming to solve problems (Martins et al. 2010). If a programming environment motivates users then they are more likely to keep working (D. Wang, Wang, and Liu 2014) but only if they perceive that ultimately they have the capability to overcome the task they have been given.

Section 2.6 moves on to examine the global growth of digital technology and the consequences this will have on the provision of teachers who are able to teach computing, programming and CT.

2.6 GLOBAL RECOGNITION OF COMPUTATIONAL THINKING

Developing digital economies throughout the world have prompted a growing awareness of the need to improve computing skills and consequently address how these are taught. In September 2014 the UK announced the need to improve the education of computing and stated that computing would be a statutory subject for all children from age 5 to 16 (Department for Education 2013). Since then a number of countries have shared this view, including several from Europe, Japan, USA and Malaysia, and are taking actions to address this.

2.6.1 Digital Economy Initiatives

The focus on growing digital economies has resulted in the formation of a number of groups worldwide to develop and share best practice in support of these economies.

D5 is one such group which considers itself to be a group of digitally advanced countries and is comprised of Estonia, Israel, New Zealand, South Korea and the United Kingdom. The group has put forward 9 principles of digital development, one of which is "Teach children to code" in order to build "the next generation of skills" (Mitchell B. 2015). London hosted the first meeting of D5 during which they looked at ways to support computer coding in schools. The meeting acknowledged that coding is only part of a bigger picture and that teaching CT is of paramount importance (Mitchell, 2015). D5 is helping various organisations to work at an international level, including Computing At School (CAS) in the UK.

CAS was established in 2008 (Crick and Sentance 2011), and is a school-led, foundation with around 22,000 members as of March 2016 (*http://www.computingatschool.org.uk*) and is one of the driving forces behind the new computing curriculum in England *(http://govenmenttechnology.blog.gov.uk*). CAS provides a forum for collaboration between school teachers, academics, corporations such as Microsoft and the Department for Education, enabling the sharing of good practice and techniques for teaching programming. Local CAS communities meet regularly to allow networking among teachers and promote awareness of computing to those both with and without a computing background (Crick and Sentance 2011; Bradshaw and Woollard 2012). Specifically, CAS aims to promote the establishment of computing in UK schools and to increase knowledge and confidence among teachers to teach it (CAS, 2015).

Singapore established the Infocomm Development Authority of Singapore (IDA) to *"drive initiatives for teaching coding and computational thinking from an early age"* (J. Tan 2015). CT is considered a fundamental skill for future workforces in Singapore *(http://www.ida.gov.s; 2016)* and there is a growing interest among students in coding and an appreciation of its value to future careers, irrespective of the area of specialisation or gender (Tan, 2015). Despite this, among the Asia-Pacific countries Singapore was found to have one of the lowest percentages of students who consider themselves to have had the opportunity to learn coding at school *(http://todayonline.com/Singapore, 2015)* highlighting the necessity for improving the provision of programming education in this country.

In the USA, efforts to promote the importance of computing and CT education include degree programs at Carnegie Mellon University which integrate CT with a variety of disciplines. Jeannette Wing is the Director of the Centre for Computational Thinking, established at Carnegie Mellon University to explore and promote CT and its value for different domains.

Implementing CT in school education is not without its challenges because this has to begin with the provision of opportunities for teachers to learn CT and an overhaul of the school curriculum. Compounding this is the lack of knowledge of how to recognise the presence of CT in young people (Grover and Pea 2013), as discussed in Section 2.3 above.

Malaysia has faced similar problems and, realising this, implemented an Information Communication Technology Learning (ICTL) program in secondary schools from March 2002 and in primary schools from April 2005 (Curriculum Development Division, 2005) which was later extended to most schools in 2007 (Osman et al., 2012). Teaching ICT was introduced in schools with the expectation that this would enhance creativity and innovation in education but a lack of external teaching material meant students and teachers had difficulties with this subject (Hassan and Noor Asmalinda Masrom 2010). Additionally, the uptake was poor due to lack of budget and thus equipment. The teaching of ICT therefore tended to focus on the fundamentals of computers and computing, with programming more commonly being elective, and even then only in certain schools (Ministry of Education Malaysia, 2016). The situation prompted serious consideration given the global move towards digital technology and CT (Yadav et al. 2014; Grover & Pea 2013b; Henderson et al. 2007). Schemes such as the joint public-private-academia initiative mydigitalmaker, intended to encourage Malaysian youth to move from being digital users to digital producers, have been created (Azlee 2016; Lin 2012) and the Ministry of Education (MOE) has taken steps to widen the usage of ICT among pupils by providing schools with ICT infrastructures (preschool to post-secondary education, 2013).

In July 2016 the Ministry of Education (MOE) in Malaysia updated the syllabus taught in primary schools to include coding, following a proposal by the Malaysia Digital Economy

Corporation (MDEC) (Azlee 2016). The CEO of MDEC acknowledged the importance of coding as a “driver towards higher thinking skills”, as have some other researchers (Mahmood Y, 2016) and it was intended that new syllabuses would be in place from January 2017 (Azlee, 2016). MDEC supports The Beaver Computational Thinking Malaysia initiative (http://beaver.my) which encourages students to get involved with logical problem solving. This is part of a wider initiative, the Bebras (meaning “beaver”) effort, which was organised in Lithuania in 2004. Bebras’ aims to align with CAS and IDA; to stimulate interest in Computing and CT in teachers, students and the general public. In Malaysia, the first Beaver competition was held in 2015 and the objectives are (source: http://beaver.my) to cultivate students' creativity, information culture, algorithmic and computational thinking. Second, to facilitate a deeper understanding of information technology. Third, to encourage students' use of information technologies in their learning activities more enthusiastically. Fourth, to engage children in information technologies, computers and their application from the very beginning at school. Fifth, to reveal to students the advantage of information technologies, computers and their application from the very beginning at school. Finally, to help students take advantage of information technologies that are helpful in learning various subjects.

In Malaysia, however, despite Beaver promoting CT awareness, the limitations are still present. Whilst it has been acknowledged that coding and CT should be taught in schools, there has been a lack of technology and teachers themselves have shown concern that they lack the skills to teach programming (Azlee 2016). Early studies have also observed primary school teachers struggling to integrate technology in meaningful ways (Hassan and Noor Asmalinda Masrom 2010; Osman et al. 2012). In a study in 2010 the MOE found that around 80% of teachers spent less than one hour per week using ICT and only a third of students thought their teachers regularly used such technology. Additionally, a 2012 UNESCO review found instruction using ICT tools to be limited mainly to word processing applications (Malaysia MOE, 2012). Whilst ICT has a significant potential to promote and accelerate Higher Order Thinking Skills (HOTS) (Malaysia MOE, 2012), use of ICT in the classroom has not been common practice. This highlights that the work developed in this thesis has come at a timely juncture where it

is seen that ways of encouraging teachers to learn to think computationally so that they can teach children programming are needed. Overall, there is a need for the Malaysian education system to realign its curriculum with its assessment system in order to ensure effective implementation and assessment of HOTS (Tan & Halili, 2015; Malaysia MOE, 2012) and a need for teachers to be able to teach these skills. In July 29, 2017, Deputy Education Minister Datuk Chong Sin Woon stated in BERNAMA (BERNAMA, 2017) that the special training is conducted with the assistance from 100 lecturers from teacher training institutes, ministry officers who are British Computer Society-certified master trainers, and also 90 lecturers from Institut Aminuddin Baki (IAB). The objective of the this is to teach CT among teachers. He added that CT and computer science would not be standalone subjects in primary schools but would be integrated into core subjects. Moreover, with the help of MDEC, the syllabus was benchmarked against courses offered by Britain's Computing at School as well as the United States' Computer Science Teachers Association. MDEC vice-president Ms. Sumitra Nair stated that students will learn Computer Science in secondary school, later gradually being introduced to coding languages, including Scratch, Java and Python. Chong stated that eventually the programme would be developed after a comprehensive survey of stakeholders, including parents and teachers so that it would succeed very shortly afterwards.

The next section goes into more detail regarding efforts taken by the MOE to enhance Continuing Professional Development (CPD) of its teachers.

2.6.2 Continuing Professional Development (CPD) in the Malaysian Education System

In order to answer RQ 6 – positively – it will be necessary for the right environment to at least have begun to exist in Malaysia. The Malaysian Ministry of Education (MOE) recognises the importance of Continuous Professional Development and over the past few years has promoted the CPD of teachers through the introduction of the School Implementation Specialist Coaches (SISTCT) scheme (Malaysia MOE, 2012), which aimed to provide training "on the ground" for teachers. The scheme, which is not ICT specific, involves subject professionals, who are teachers, in a knowledge-sharing exercise whereby expertise and experiences are shared with teachers in other schools.

Additionally, the Performance Development Program – High Performing Schools (HPS), another initiative, aims to improve student performance in under-performing schools. Since 2013, the MOE improved CPD further with a scheme to provide teachers with greater support through an individualised CPD programme in which they engage with an assigned supervisor. This programme, run by the Teacher Education Division, includes training relevant and common to all teachers as well as an elective component that can be tailored to the specific teaching domain and individual developmental needs. The programme is run at schools and uses a network of peers to teach and circulate best practices.

CPD is not limited to teaching-only roles. A new Principal Career Package has recently been rolled out by *Institut Aminuddin Baki* (IAB) to enhance the performance of school leaders (principals, assistant principals, subject leaders and department heads) (Malaysia Education Blueprint, 2013-2015). The intention of this is to give greater support to heads of schools to help them achieve their full potential, but then to hold them more accountable for delivering higher student outcomes. This includes those about to be appointed as principals, and the package includes a new "on-boarding programme" of mentorship whereby the soon-to-be principal is mentored for one month before the outgoing principal leaves. Once they have formally started their new role the new principal is also given a period of coaching from an experienced principal or district School Improvement Partner (SIPartner+) (Malaysia Education Blueprint 2013-2025).

Such coaching and mentoring continues beyond a new principal's first year to enable them to continually develop in their profession. The aim is that school leaders will be prepared for, and have the flexibility to, make decisions regarding school improvement planning, curriculum planning and also administrative leadership matters such as allocation of school funds.

A new common civil service track (Malaysia Education Blueprint 2013) gives school principals the flexibility of working at primary or secondary schools.

Principals are now assessed annually using a new evaluation instrument. This is designed to evaluate their leadership abilities as well as their effectiveness in improving

student outcomes. If a principal performs to a high level then they will progress faster and be given more career opportunities. Such high-performing principals will also be given new, revised incentives to move to rural or underperforming schools. Principals who do not meet the minimum standard will receive extra coaching support and those who consistently underperform will be redeployed back to a teaching position in another school.

As with teachers, therefore, the aim is for school leaders to mentor and train one another, leading to the dissemination of best practices.

The MOE acknowledges the need for effective teaching of HOTS in the classroom. However, ICT usage in schools, in terms of both quality and quantity, has been continuously below its expectations. A recent initiative has involved the development of CPD centres and, in 2017, MDEC established myDigitalMaker CPD centres, specifically for teaching Computer Science to teachers, at six universities in Malaysia. These centres (at USM, UiTM, UTM, UNIMAS & UMS) are located regionally in order that teachers can easily access CPD in preparation for teaching the new ICT curriculum. MDEC has been working specifically with faculties of Computer Science, Education and Engineering to establish these centres with more planned in the near future.

It is clear that the Malaysian government is making strong moves towards improving ICT resources and knowledge, but this is at a very early stage and, at the time of writing, the CPD centres have only very recently been put in place. The MOE has spent more than 6 billion Malaysian Ringgit (MYR) on technology over the past decade in education initiatives such as smart schools (Umar and Hassan 2015) and it has recognised that teaching expertise needs to be in line with these developments in order to meet the ambitious goals that Malaysia has set itself. However, there is a lot still to be done.

2.7 CONCLUSION

The fundamental problem being addressed in this PhD is that, in Malaysia, teaching provision of coding and CT in schools needs to increase and improve to allow the country to stay a part of the growing global digital economies. It is also becoming more widely recognised that CT skills are valuable throughout life and in different disciplines.

Enhanced teaching provision will require pre-service teachers themselves to have access to an education in programming and CT before they can go on and teach it. However they also need to be motivated to learn CT which requires overcoming fears and prejudices. This PhD investigates how that could be achieved and the feasibility of that achievement given the current situation in the Malaysian education sector. The next chapter details the methods used in this study before moving on to Chapter 4 which presents Study 1, the initial phase of work towards this goal.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 INTRODUCTION

Chapter 1 introduced 6 research questions each seeking to address the goal of investigating the feasibility of improving the ability of Malaysian primary teachers (principally) to be able to think computationally and thus be more able to teach their students to think computationally and thus to be enable them to learn to code. Chapter 2 reviewed a number of tools that have primarily been used to teach programming and CT skills to children or young people (pre-service and in-service teachers) but there was a lack of tools or methods that had been used to teach primary school students to program (Seiter and Foreman 2013). The same is to an extent true of teaching non-computing background teachers to be able to teach CT or to be able use this to teach coding in primary schools. Most of the techniques described in chapter 2 were seen to be focused on motivating children to learn. There was a focus on what makes good motivators for children. This included using game design and graphical interface based (drag and drop) languages like Scratch. Teachers on the whole are less likely to be motivated by this and more likely to be motivated to learn to use a tool that they could use to develop their own teaching artefacts.

It was thus decided to tackle the research questions by undertaking three further phases of research study. The first of these was to undertake a more formal data gathering exercise on the students at UPSI before and after taking the standard Flash Action Scripting class. This would provide data for RQ 1 to RQ x. The second phase was designed to begin answering RQ y – z. The hoped for outcome from this phase was the information needed to develop a standalone software that would embed training in CT skills and assessment of whether they had been achieved. This is in line with RQ y. The rest of this chapter is devoted to describing the research methods that were used for the study and why they are appropriate.

3.2 OUTLINE OF WORK

The progression of the three phases and the methods used in each are shown in Figure 3.1 below.

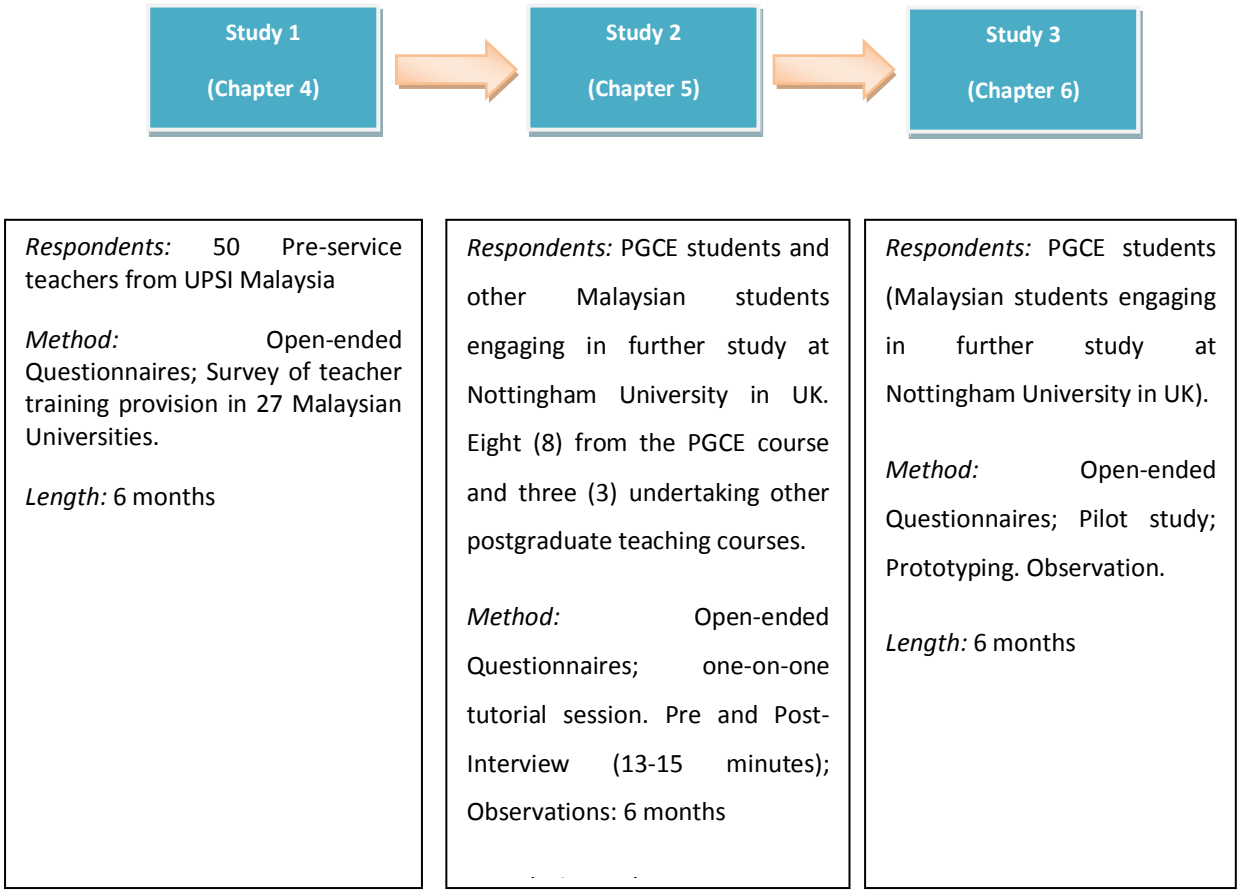


Figure 3.1: *Outline of Research Study.*

The methods used are now described in more detail in Section 3.3 along with a discussion of the rationale behind each method. The tree different phases were designed to refine the approach generated from the literature review, compare and contrast how the approach can be applied and finally contribute to theory and practice.

3.3 METHODS

Prior to any research directly involving users, a survey of in-service teacher-training provision at Malaysian universities needed to be performed to identify where gaps

might exist. This survey involved Internet-based research into courses that were provided for pre-service teachers, and any relevance of that content to learning programming and CT skills.

A User Centred Design (UCD) (Maguire 2001; Willis and Willis 2016) approach was taken throughout the rest of the studies. Willis & Willis (2016) added that *"user-centred design is often promoted as 'socially responsible' because it focuses on how people actually interact with a specific product".* This was not only important because the tool was being designed for users, but also because the tool had to be usable and appropriate for pre-service teachers to use to develop their own teaching content for other learners. Additionally, the tool and experiments needed to be designed in a way that allowed for the presence of Computational Thinking to be observed and recorded.

A number of methods were used to conduct these UCD studies including questionnaires (pre- and post-), observations, interviews and prototyping. These are described in more detail in the following sections.

3.3.1 Questionnaires

Questionnaires are particularly beneficial for fact-finding (Bell 2005) but also allow researchers to measure responses from a large cohort in a short period of time with minimal cost (Atkinson, 2008). They can also be a valuable tool in the assessment of changes that might have occurred through participation in the studies. In particular, attitudes and motivation needed to be assessed before the start of the Initial phase experiment and then again at the end. The purpose behind using this method was to be able to present subjects with identical questions and the same approach to asking those questions, so that differences in the answers can be used to infer whether an effect has taken place from their participation in the experiment. Questionnaires lend themselves to this because it is easy to keep consistency between what is asked in the pre- and post- conditions and this gives confidence that confounding variables have not been introduced unwittingly (Kathy Charmaz 2014; Yin 2009). The disadvantage of questionnaires, however, is that it is not subsequently easy to get participants to

expand on answers that prove interesting to the researcher. Open-ended questions can be included to get more qualitative data but their usefulness depends on the participants' willingness to write longer answers and to explain them clearly (Papastergiou 2009). Open-ended questions should be used carefully said (Smyth et al. 2009). Basically coders code the open responses on the core of a predefined categorisation scheme (Züll 2016). Open-ended questions may request textual information (for example "Can you tell me what you associate with the term 'programming'?") or numerical information (for example, "How many minutes per week do you do some programming?"). Because numerical information differs from textual information, the present involvement focuses only on aspects of the collection and analysis of textual responses to open-ended questions (Züll 2016). Open-ended questions can be used both to gather information and to motivate respondents (Züll, 2016). They are used for information-gathering purposes in the following situations such as firstly, knowledge measurement - Züll (2016) stated that open-ended questions are more appropriate than closed-ended questions for measuring knowledge, because they not only minimise the possibility that respondents will try to guess the right answer but also often produce more reliable and valid information. However, when respondents cannot immediately recall the correct answer, open-ended questions also elicit more "don't know" responses or refusals than closed questions do (Krosnick and Presser 2010). Secondly, unknown range of possible answers - The use of open-ended questions is recommended if it is not yet possible to clearly restrict the subject of inquiry, or if one expects new topics to occur. Thirdly, extremely avoid from long lists of response options - a further field of application for open-ended questions is when the number of possible answers is so long that the question cannot be asked with predefined response categories (Züll 2016). An example of this is the motivation to learn question. Although researchers know what motivates them, respondents cannot, of course, be presented with a list of several hundred response options. Fourthly, avoidance of directive questions - open-ended questions can be used to avoid navigating respondents in a particular direction. Response categories comprised a selection of the best-known bands and reflected only the researcher's own ideas (Züll 2016). The selection would navigate respondents in a certain direction and there the categories would be a danger to be picked because that were only choices given.

Finally, cognitive pretesting - open-ended questions are normally used to check whether the meaning of a question was understood as intended (Lenzner, Neuert, and Otto 2016).

Moreover, using open-ended questions for information-gathering purposes, should also be used to motivate respondents by giving them an opportunity – in between the many closed-ended questions with predefined response categories – to express their opinions freely and in their own words (Lenzner, Neuert, and Otto 2016). In order to give respondents an opportunity to express criticism or make comments, the following open-ended question is often asked on the last page of the questionnaire: "Could you please give any of your opinions about today's survey?".

Answering open-ended questions gives a greater burden on the respondent than does choosing a response category in closed-ended questions (Lenzner, Neuert, and Otto 2016). The answer must be expressed in the respondent's own words. They should able to express their responses freely. Numerous researchers have studied which respondents are even capable of answering open-ended questions. A major function in answering open-ended questions is played by respondents' level of interest in the topic of the questions (Groves, Presser, & Dipko, 2004; Holland & Christian, 2009).

In the case of open-ended questions asked during face-to-face interviews, an important role is played by the interviewer. By giving extra information and asking follow-up questions, interviewers can help respondents and motivate them to provide better and more comprehensive answers (Smyth et al. 2009). However, interviewers are also responsible for recording the responses – that is, they must enter the respondents' expressions into the questionnaire (as completely and faithfully as possible).

The details about coding can be found at the following section 3.4.

3.3.2 Observations

Study 2 of this research involved user observation. Observations form a vital component of many user studies because they enable the recording of qualitative information about behaviours, approaches, feelings and dialogue which is not necessarily possible using other methods (Cohen et. al, 2007). Observations can be

conducted on users in their natural working environment ("in the wild") or in an environment set up by the researcher. Data can be recorded through note taking, video, audio or other means, such as recording actions through software.

In these phases participants were required to use a tool as part of the experiment and it was imperative for the researcher to be able to gain a thorough understanding of the users' interactions with the tool and to appreciate the difficulties that may have been present. Such interactions and difficulties could only be understood fully by observing real time use of the tool. Additionally, the rich data set which observations can produce is recorded within the context of the task being performed and the environment within which it is being performed, thus enabling the researcher to consider not only the progress of the user in completing the task but also the influences of other factors and experimental conditions. Observations therefore result in a large quantity of qualitative data which then has to be analysed for trends and interesting or unusual behaviours.

Participant observations are a good method to use when the researcher needs to ask more about a subject's actions or to ask them to explain their thinking. Observation (Kawulich 2005) is therefore used for gaining greater understanding of phenomena from the point of view of participants, and can be used in diverse areas like sociology and education (Kawulich 2005; DeWalt & DeWalt, 2011). The disadvantage is that the presence of the researcher or the fact that subjects are aware they are being observed can influence their behaviours and hence the data recorded (K Charmaz 1990; Kathy Charmaz 2014).

In this study observations were recorded using Camtasia Studio, a screen capture program, which is a low-cost approach that allows recording and replay of experimental sessions and the interactions that take place within these (Arnold et al., 2008). Camtasia Studio software suite includes a screen recorder (Camtasia Recorder) and a video production tool (Camtasia Producer) (Arnold et al.). Audio can be recorded separately or at the same time as the screen activity. Camtasia was used as a way of capturing activity in a less obtrusive way than the researcher making constant notes. Ethics approval was given for both studies and this is discussed in detail in section 3.6. The same data analysis methods were applied to analyse the data to keep consistency.

Data from the sessions were collected using audio recordings and screen capture. Camtasia software was used to record the audio and screen activity (such as mouse clicks). After completion of the tasks, respondents were interviewed to explore ambiguous actions and to solicit additional feedback (Novotny E., 2004). The software is as simple to use as a tape recorder. Firstly, participant sat in front the computer and display Camtasia Studio. Participant were explained on how to handle a task for example he/she need to work alone without any guides from a tutor. All the activities on the screen will be recorded (screen and audio recordings) - a region of the screen, a particular window or the entire screen. All screen activity will be recorded until the stop button being pressed. These videos could record research consultations or reference encounters and send the files to the users for later review. These videos could also be used for assessment purposes along the journey of the studies. All-in-all, Camtasia Studio is an easy, inexpensive tool that anyone can use to create streaming media without the stress or cost of a big-time video shoot (Charnigo 2009). Moreover, this software works best when users wish to demonstrate how something works rather than to explain (using text) how something works (Charnigo 2009).

Researcher will code the videos for the types of interactions participants had with the software. The unit of analysis was the participant act at a computer station (Davis et al. 2015). Camtasia Studio 8 used to capture and to record the programming writing steps and also any tasks that the respondents works on the screen. After completing the tasks, participant discussed few questions in an exit interview as shown in Appendix G.

3.3.3 Interviews

Interviews are another approach to gathering qualitative data about users' thoughts and feelings but they allow a deeper understanding of these than can be captured through observation (Patton, 2005). They allow the researcher to ask questions around a topic guide and expand upon answers of interest. In other words, there is no fixed set of questions that have to be adhered to. Interviews were conducted pre- and post-experiment in Studies 2 and 3 and this method was chosen over the use of questionnaires. Research suggests that the way questions are asked can influence what participants reveal (Charmaz, 2006) and it was felt that the researcher could put

participants at ease through creating more of an informal feel and speaking to them in Malay, their native language. Merriam (2009) supports this view that the method of interview can help researchers gather specific kinds of information and explore participants' thoughts more fully. Additionally, whilst questionnaires can be used to ask about areas of interest, interviews enable more qualitative data to be gathered and this was considered important when examining the use of software by novice programmers.

The flexibility of interviews also allows the researcher to follow up on observation or questionnaire findings (Cohen, Manion, & Morrison, 2007; Heck, 2006). There is enough control to allow for comparability of results while permitting respondent freedom and interviewer queries.

However, unlike questionnaires, interviews are much more time consuming and therefore costly because they require the researcher to be present while the questions are being answered but the intended sample size was not unrealistic for this.

3.3.4 Prototyping

Prototyping is an important component in User Centred Design as it allows rapid iterations of versions of the system or tool being prototyped. The advantage of prototyping is that it is quick and easy and changes can be quickly made to the design following user feedback. This also means that cost is kept to a minimum in terms of time and people working on the project. Designers can prototype using different fidelities with different tools and this allows the gradual increase in realism and of working features that can be presented to the user once initial designs have been agreed.

Prototyping was done using the Flash Action Scripting environment as it is designed to allow the quick and easy production of multimedia material. It was therefore considered an environment appropriate for pre-service teachers and in-service teachers and which could be used to observe CT (Angeli et al. 2016) skills in use as well as enable the production of course material. Prototyping was prior to Study 3 where it was used to find out whether participants could demonstrate CT skills.

The tool was piloted with HCI experts (Computer Science post-graduate students with the experience in Human Computer Interaction, system development and multimedia) over a period of 5 weeks. The experts were given a system prototype to use for about twenty to sixty minutes each time. They then completed questionnaires as shown in appendix C. Once the tool had reached a stable version it was deemed to be ready for use in Study 3. The main objective of the piloting of the tool was the validation of the language used in the interface, testing of the functionality of the tool and examining the overall design of the tool and the scenarios that it would present to users.

Prototypes had been used as a way to try out ideas for the design of programming tools. A prototype is "the first form of anything from which all later forms develop, sometimes with improvements" (Long Dictionary, 1983). High fidelity (HiFi) prototype had been used Please refer to the Appendix J). Fidelity is how truthful or close a copy of something is to be original. (Longman Dictionary, 1983). In this case the 'original' is an imaginated artefact (a programming tool). A prototype that has high fidelity can be very close to the final artefact. Prototype can be varying in depth, they can model a few aspects in depth, in a fully developed way. Hifi prototype are created using the medium of the final artefact. Computer-based HiFi prototype was also used extensively in this study. The advantage with HiFi prototype is that they can provide a user experience that is more or less identical to the experience of using the final version of the artefact. The prototype that looks finished can also be drawback; users may be reluctant to suggest alterations to something that appears to be finished. Furthermore, it might be easier to imagine alternative designs if the prototype looks unfinished (Kindborg, 2003). This prototype was used in the HiFi sessions in the studies reported in chapter 6.

The prototype is a design experiment that was done after the Study 1 and Study 2 but prior to Study 3. The purpose of this prototype was to get feedback from the users in terms of the interface design, usability and the functionality. Another purpose of having prototyping was to look at how Flash Action Script programming can be easily understood by a non-computing background users and adopt CT skills.

The prototype was designed in Flash Action Script software. The development of the prototype did progress relatively smoothly. Prototyping was conducted at the same

venue that was at Technology Research Lab at Jubilee Campus with two experts. They were two individual sessions took place at the computer science lab with the duration between thirty to sixty minutes

After all the data had been analysed (the respondents will suggests some improvements should be made in certain sections), the result from it will be reviewed and all the positive feedback and suggestions will be amended from the original version. Therefore, the improved version of tool will be produced and the real experiment (Study 3) will take place based on the time scheduled. The details on prototyping can be viewed at chapter 6.x.x

3.3.5 Pilot Study

The Implementation of pilot studies are a crucial element of a good study design (van Teijlingen and Hundley 2001). It will aid the research in shaping if there are mistakes or any other weaknesses within the interview design and enable him or her to do some revisions earlier to the implementation of the study (Turner, 2010; Kvale, 2007). A pilot test should be conducted with participants that have similar interests as those that will participate in the implemented study. The pilot test will also assist the researchers with the improvement of research questions (Turner 2010). Pilot studies fulfill a range of important functions and can provide valuable insights for other researchers (van Teijlingen and Hundley 2001). However, a pilot study can be the pre-testing of a particular research method (Baker 1994: 182-3; Turner 2010).

One of the advantages of conducting a pilot study is that it might notify researcher about where the main research project could fail, where research protocols may not be followed, or whether proposed methods or instruments are inappropriate or too complicated (van Teijlingen and Hundley 2001). As stated by De Vaus (1993: 54) “Do not take the risk. Pilot test first.” These are important reasons for undertaking a pilot study, but there are more reasons behind it, for example convincing funding bodies that the research proposal for the main study is worth funding. Therefore for this study, researchers will start with “qualitative data collection and analysis on a relatively topic” (Tashakkori & Teddlie 1998: 47). Pilot studies may also try to identify potential practical problems in following the research procedure (van Teijlingen and Hundley 2001). This is

particularly important because pilot studies can be “time-consuming, frustrating, and fraught with unanticipated problems, but it is better to deal with them before investing a great deal of time, money, and effort in the full study” (Mason and Zuercher, 1998).

Before a Study 3 begins, a pilot study took place. The process took place about four to six weeks. This study was conducted with five (5) respondents from the University of Nottingham. Three of them were from PGCE trainees programs here whilst the remaining two were from postgraduate students (existing teachers) from Malaysia who further their studies at School of Education University of Nottingham. The details about this piloting process will be explained in section 6.3.

3.4 QUALITATIVE ANALYSIS

3.4.1 Content Analysis

Content Analysis as a technique, has been extensively practised to analyse transcripts from Computer Mediated Communication (CMC), to Computer Supported Collaborative Learning (CSCL) research (De Wever et al., 2006; Laat et al., Newman, Webb, & Cochrane, 1995; Rodriguez, 2014; Schrire, 2006). Meyer (2004) stated that it is hard to justify the quality of an interaction by only showing the number of participants’ contributions to a discussion. The main purpose of content analysis is to give a deeper understanding of situations that may not be immediately obvious by reading the transcript (De Wever et al., 2006).

In section 3.3, justification was developed as to why and how the mixed methods (both qualitative and quantitative methods) approach was adopted in order to test the research questions at different stages. In this section the various data analysis techniques used in order to obtain the results for this research are justified.

Thematic Coding Analysis (Braun and Clarke 2006) was used to analyze the observation and interview data in each phase of this study. Charmaz (2009) suggests that there are at least two primary phases of coding in grounded theory: the preliminary phase where

data is scrutinized by segment; and the second focused or selective phase where large amounts of data are organised by the most frequently occurring codes.

3.4.2 Thematic Analysis

The technique of inductive thematic analysis was applied to compose new themes from data collected from the observations and interviews (Braun & Clarke 2006). The process involves 6 phases of thematic analysis including firstly familiarisation with the data before generating initial codes and examining the data for themes based on these. Themes (Braun and Clarke 2006; Murphy et al. 2004) need to be reviewed in order to ensure that the codes and themes link together reliably, and thus relationships between themes are examined with the potential to integrate themes or create sub-themes.

Using these methods allows quantitative values to be extracted from qualitative data to enable further analysis and to give a clearer picture of the results from the experiments.

Inductive thematic analysis (Thomas 2003; Fereday and Muir-Cochrane 2006; Styles and There 2012) was chosen because no existing coding scheme or model was found that could be used appropriately. It therefore required an exploratory process to define the categories and themes which emerged from the data. This is revisited in Section 3.5 which discusses measures taken to ensure the reliability and validity of the data.

3.5 RELIABILITY AND VALIDITY OF DATA

Ensuring the validity of data is dependent on a number of considerations which range from the sampling method used to enlist participants in the study to using the correct approach in collecting and analysing the data.

Data collection requires appropriate methods that will allow researchers to answer research questions and therefore the data collection instrument must measure what it is designed to measure, with the conclusions following on precisely from this (Bell, 2005). Risks to the reliability and validity of data collection instruments comprise dishonesty, human error, bias, lack of nuance, and the inclusion of a limited number of respondents (Boeije 2002). Additionally, the interviewee and/or the interviewer may

not reveal all that they know in relation to the questions. The most correct way to increase validity is to control for bias in the design and content of the questions and the attitudes and behaviors of the interviewer (Cohen, Manion, and Morrison 2007). Using a mixed methods approach is additionally important to allow triangulation of data and thus enhance validity and reliability (G. W. Ryan and Bernard 2003; Kolb 2012)

This study used purposeful sampling (Kathy Charmaz 2014) due to the fact that a very specific niche of the population was to be studied. Cohen, Manion and Morrison (2007) stated "There is little benefit in seeking random sample when most of the random sample may be largely ignorant of particular issues and unable to comment on matters of interest to the researcher, in which case a purposive sample is important" (2007, p.115). Participants were recruited through educational organisations with which the researcher had links so that it could be ensured that participants met a strict set of criteria, i.e. they were Malaysian pre-service teachers. Whilst this sampling technique limits the number of participants that take part, the data gathered are from relevant, information-rich sources and it also enables the most effective use of limited resources in terms of number of researchers and time available.

The participants in this research were teachers or pre-service teachers and they were all from a non-technical or non-computing background. However, they were all aware of the need to start to teach Computer Science (Secondary level) and/or Computational Thinking (CT) (Primary and Secondary level) due to changes being introduced by the Malaysian Ministry of Education (Azlee 2016; Singh 2016). The requirement to teach coding, programming, computational thinking and related topics is an ongoing process currently in Malaysia (Azlee 2016) and participants were required to learn to teach these in the near future.

No absolute guarantee can be given as to the reliability of the data gathering and analysis, and therefore the validity of the data, but efforts can be taken to reduce the likelihood of spurious effects on the data. Such efforts include a scientifically rigorous design of the experiments (Morse et al. 2002) using data gathering approaches which are appropriate for the type of experiment and the type of data to be gathered. Section 3.3 has already outlined the methods used and reasons for this. Each experiment and

the data gathering technique used was piloted to determine where there were any problems, and those problems corrected before re-piloting.

In thematic analysis the identification of themes from data can be subjective and so care needs to be taken that the analysis isn't influenced by the researcher. Coding was done by more than one person so that the codes that resulted were free from bias before themes were generated (Braun and Clarke 2006; G. W. Ryan and Bernard 2003). Any statistical analysis performed should be appropriate to the type of data and the number of participants. In this research a limited number of participants were studied and so only basic statistics were appropriate for the analysis as more complicated techniques would not have produced any meaningful interpretations and may have been misleading.

Finally, the methods used need to be written up clearly and precisely, thus allowing others to criticise the approaches and determine whether the researcher's inferences are appropriate and correct.

3.6 ETHICAL CONSIDERATIONS

As with any study involving users, there were a number of ethical considerations which had to be addressed. These included not only how the data were collected, stored and analyzed but also that participants gave their permission for this and were correctly informed of how their data would be collected and what would be done with it. As such, all studies involved participants being given information and consent forms (see Appendix IV) and being reassured that they could withdraw from the study at any time and that their data would then be destroyed. Measures had to be put in place to accurately and appropriately record the data and store it securely. Reassurances (Cohen, Manion, and Morrison 2007) had to be made that data would be stored securely and for a limited time period only and that participants would remain anonymous. Confidentiality was ensured and participants were given the researcher's contact details should they have any questions. The designs of the experiments were submitted to the School of Computer Science Ethics Committee, University of Nottingham and approval was obtained before any data were collected.

As the participants were not considered vulnerable (i.e. no pupils would be involved in the studies) and there was considered to be minimal risk to those taking part, it was not necessary for additional safeguards to be put in place such as security checks.

3.7 CONCLUSION

This PhD involved studying users in three experimental phases or studies, through which it was intended to gain an understanding of the role of CT skills in learning to program. A UCD approach was taken using a number of research methods which have been outlined in this chapter with justifications for their selection. The three studies in this PhD are described in more detail in the following chapters, each of which gives a detailed description of the experimental procedures used in a study and presents and discusses the results. Study 1 is described in Chapter 4. Chapter 5 reports on Study 2 before leading on to Study 3 which is reported in Chapter 6.

CHAPTER 4

STUDY 1

4.1 INTRODUCTION

As was stated in chapter 1 the initial motivator for this research was lecturing experience in a teacher training university in Malaysia. The context of this, as brought out in Chapter 2, was the increasing international interest in making sure a greater supply of coders is produced. At the time the research study started Malaysia had not recognised that this was a crucial and growing need. This situation has changed dramatically over the time of the research and awareness of the need to prepare particularly primary school teachers has grown. This has been highlighted in chapters 1 and 2. It was thus valuable first of all to see what efforts are being made to introduce teaching of how to program to pre-service teachers. The first part of this chapter presents the results of an internet based search of primary (or elementary) school teacher training provision in Malaysia to find out how much is being done to introduce teaching of computer programming within the sector. The second part of the chapter details a more formal evaluation of the attitudes of pre-service teachers in Sultan Idris Education University (UPSI) towards learning to program and of their reaction to the current teaching they were getting.

4.2 PROGRAMMING TEACHING PROVISION IN MALAYSIA TEACHER TRAINING COURSES

This section presents the results of a review of teacher training provision in Malaysia aimed at identifying whether any components were provided specifically for teaching programming or CT skills. It was anticipated that this would give an overview of the level of importance of such provision placed by the Malaysian government.

Twenty-seven Malaysian universities were surveyed online and, whilst most of these had some form of ICT training, none were found to be providing specific programming or CT skill training.

Since provision of programming training was found to be so scarce in Malaysian universities the same provision, that of teaching programming or CT skills, at Malaysian primary and secondary schools was reviewed. It was found that, prior to August 2016, teaching programming skills had not been on the curriculum (source: *Statistic division Malaysia of Higher Education*, 2015), and only very recently, subsequent to this PhD study, has the Malaysian Prime Minister announced that coding should be taught at primary schools from 2017 (Abas, 2016). Leading up to this, only ICT skills such as use of Microsoft Office were taught (Umar & Hassan, 2015).

It is clear that, in order for programming to be on the curriculum at schools in Malaysia, there has to be some provision for those in a teaching role to gain skills in programming, and this provision would be most suited to university study. UPSI was found to provide a technology module that involves teaching the use of Flash Action Scripting but it did not support this with deliberate teaching or assessing of computational thinking (CT) skills. The students were expected to learn the subject by themselves or explore the software in order to learn the subject. Thus, students developed negative feelings towards the course as can be seen later on in this chapter.

It is worth noting that at the time of writing, in Malaysia the situation has now improved with two of the largest universities (The National University of Malaysia and University of Science Malaysia) offering teaching masters' courses with majors in ICT and three others now offering an ICT element in a more general teaching masters. All 27 of the specifically teacher training institutions now offer ICT as at least an elective in their teaching degree programmes with the Education Institute of Penang now offering a major in ICT.

However, the situation is still one of needing more provision for teaching programming. Part of tackling this problem is not just teaching the subject but also

enabling students to overcome any doubts and insecurities regarding learning to program. In order to do this, it was necessary to identify any negative feelings that students might have. The next section describes an initial experiment conducted with pre-service teachers in order to determine their levels of motivation to learn to program.

4.3 ATTITUDES TOWARDS PROGRAMMING IN UPSI TEACHER TRAINING STUDENTS

In order to answer the first two objectives of Research Question 1 (RQ1) a study was carried out with pre-service teachers to judge their motivation and attitudes towards the idea of learning programming skills and the potential approaches to this learning. Students completely new to programming took part in an experiment where they attended a tutorial which taught Flash Action Scripting. Questionnaires were used to gauge their attitudes to programming before and after the tutorial. This is detailed in the following sections.

4.3.1 Participants

Fifty pre-service teachers aged from 21 to 26 years old from 6 different majors in UPSI, Malaysia, took part in this experiment. Participants were selected as those who intended to begin a career in teaching after graduation, and so would have a greater possibility of teaching computing or related subjects. This was determined so that the sample would be made up of those who had no IT background but who had chosen to enrol in a Multimedia Project Management course out of interest. Students were taken from this course who had a range of backgrounds in the social arts and humanities fields. Non-computing students were chosen because there is a literature on the underrepresentation of non-computing students in learning computing and the likely reasons for these students being generally willing to learn how to program provided they are given appropriate teaching methods (Suliman, Hawari, and Othman 2011; Wong, Yatim, and Tan 2016).

4.3.2 Method

Students attended a face-to-face tutorial (a lecture followed by a laboratory session) lead by a lecturer which aimed to introduce them to Flash Action Scripting. Two sessions with 25 students and 1 teacher were conducted in a computer laboratory. The teaching approach taken was that of student-centred learning (Levin, Buell, and Levin 2000). Some students had started to explore the topic on their own but all of them needed direct teaching from the lecturer in order to begin to learn to program.

A questionnaire was designed for use before and after the students had taken part in the tutorial in order to determine their attitudes to and level of motivation for learning to program. It was intended that this would give an idea as to whether the tutorial they engaged in had changed their views of programming, either for better or worse. The questionnaire can be seen in Appendix II. The questions asked about the students' background experience in use of computers and centred on the students' attitudes to learning a technical subject and Flash Action Scripting in particular. The questionnaire contained open-ended questions which allowed the students to provide free responses relating to their expectations and experiences. It should be reiterated that at the time of the surveys the Malaysian education system had no formal requirement to teach programming at any level, something which has changed more recently.

Before the experiment the questionnaire was piloted with 5 students who had a similar background to the target audience. The aim of this was to check that the questions would be understood by the target audience and that the answers would give indication of the types of response that would be expected in the full experiment. It was then possible to discuss their responses with the pilot group and make changes to the questionnaire to deal with any issues. The questionnaire in Appendix X to is the final one after changes. The changes were only in requiring the clarification of some terms where too great an understanding of subject specific language had been assumed. The survey and tutorial were taken by the 2013/2014 class at UPSI Malaysia in the second semester.

4.3.3 Thematic Coding

Following the study, the open-ended answers from the questionnaire were collated and a thematic analysis (Fereday and Muir-Cochrane 2006) was conducted. This type of analysis involved coding the data which allowed themes to be identified from this.

As an elective these students might have been more inclined to be positive towards technology, however, a lot of negative attitudes and fear were evident before the teaching**.** This is discussed later on in this chapter.

Figure 4.1 below shows the initial thematic map that was derived from the questionnaire results. These were condensed further to identify positive and negative attitudes as shown in Figure 4.2.

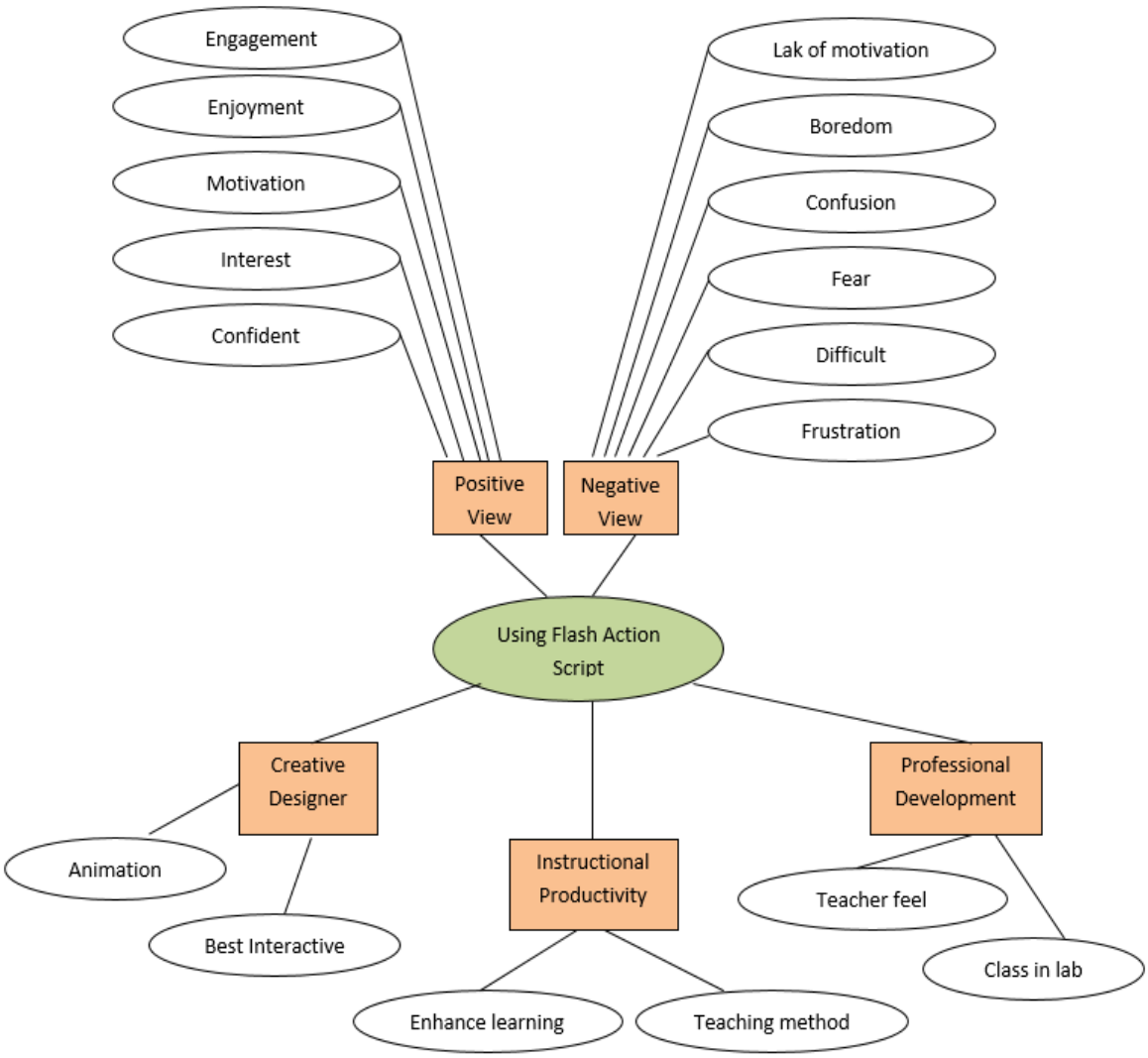


Figure 4.1: *Initial thematic map, showing five codes* (analysis as presented in Braun & Wilkinson, 2003).

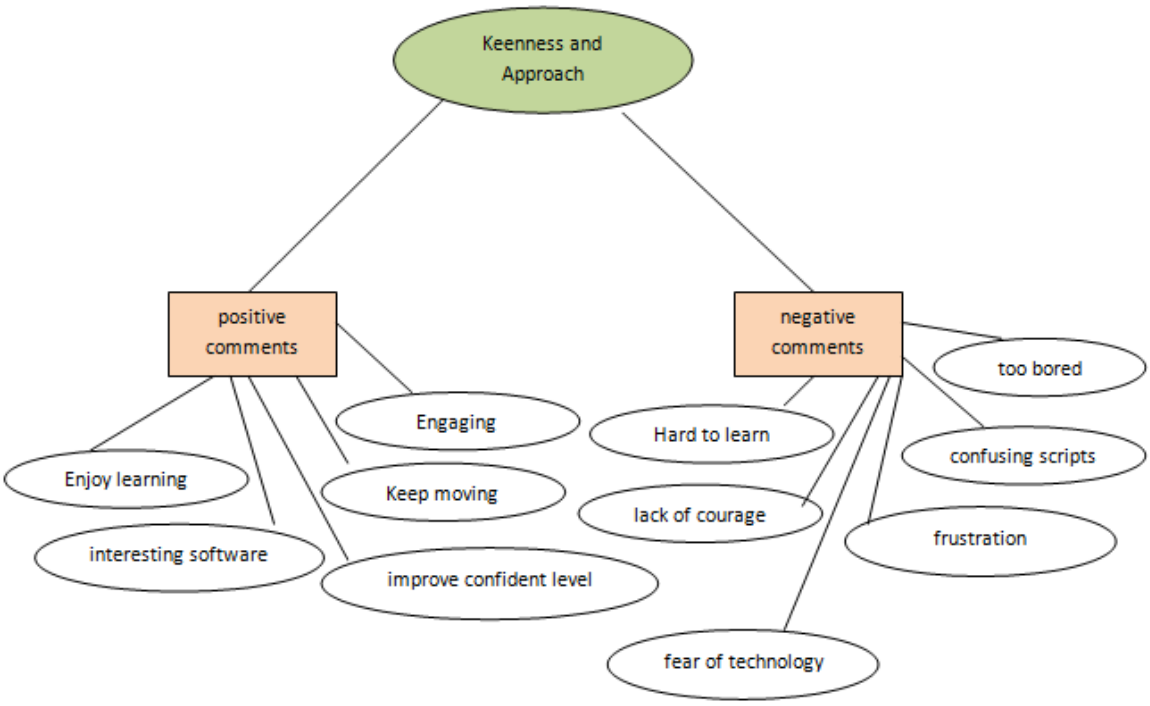


Figure 4.2: *Thematic map, showing one theme and a few codes* (analysis as presented in Braun & Wilkinson, 2003) *towards respondents' attitude to using Flash.*

As can been seen in Figure 4.2, codes were assigned for both positive and negative responses. There were 5 positive codes (engagement, enjoyment, motivation, interest and confidence) and 6 negative codes (lack of motivation, boredom, confusion, fear, difficulty and frustration) extracted from the data when looking at the attitudes towards learning how to program and how the students started to develop computational thinking (CT) skills. Additionally, 3 codes relating to their professional development goals emerged. All these codes were based on statements made by the participants and a description of the codes along with some corresponding statements are shown in Tables 4.3, 4.4 and 4.5 below.

An example of the inductive coding (Thomas 2003; Fereday and Muir-Cochrane 2006) of data can be seen in the following segment of data from respondent TT2:

"Programming is too boring. I am sure I will not able to do it." This response was considered to contain two different meaning units and therefore was counted as

two different categories. The first text segment: *"Programming is too boring."* was coded as *boredom*. The second text segment: "*I am sure I will not able to do it."* was coded as *difficult*. Thus 2 codes were extracted from these statements.

Table 4-1: *Example of Positive Themes in Coding Enthusiasm and Attitude*. The example for each quote was selected from respondents' feedback. *(data extracted with codes applied from Clarke, Burns & Burgoyne, 2005)*

Open Code	Properties	*Example Quotes*
Engagement	Expressions concerning successfully completing the tasks given	*"I feel comfortable and keen to use this software now because…"*
Enjoyment	Willingness and ability to focus on tasks given	*"I enjoy exploring the software..."*
Motivation	Feeling positive/ competent in using Flash and eager to learn more	*"Although I am not that good at using the computer, especially programming, I think the proper teaching technique will definitely motivate me to learn computing …"*
Interest	Learning to program is exciting and interesting and encourages more exploration	*"It's really good to be able to use this so that we can produce something at the end of this semester."*
Confidence	Feeling they understand the concept	*"I can now understand the concept of computing and comfortably handle the tool..."*

Table 4-2: *Examples of Negative Themes in Coding Enthusiasm and Attitude. The example for each quote was selected from respondents' feedback. (Data extracted with codes applied from Clarke, Burns & Burgoyne, 2005)*

Open Code	*Properties*	*Example Quotes*
Lack of Motivation	Less interest in learning how to program.	*"Although I know how to use multimedia to create picture books, classroom decorations, or work sheets sometimes I feel lost if I get stuck at some points." Respondent TT2*
Boredom	General negative attitudes such feeling programming is boring and not engaging with the subject.	*"Programming is too boring. I am sure I will not able to do it" Respondent ET3*
Confusion	Confusion about how programming works or how to use Flash Action Scripts to create programs, including understanding of the syntax and the importance of layer function.	*"The script looked confusing because I don't understand what and how those statements work." Respondent TT1*
Fear	Respondent expresses nervousness or fear in exploring programming codes, syntax or other aspects of the technology.	*"I am afraid of technology" Respondent ET1*
Difficulty	Phrases related to struggling with the practice of producing code.	*"I need to learn more especially the scripting…" Respondent TT8*
Frustration	Feelings expressed about the impact and progress of their learning.	*"I don't think I can proceed with this because I did not perform very well in ICT in the previous semester." Respondent TT5*

Table 4-3: *Examples of Codes Related to Professional Enhancement. Each example quote was selected from respondents' feedback. (Data extracted with codes applied from Clarke, Burns & Burgoyne, 2005).*

Open Code	Properties	*Example Quotes*
Creative Designer	Animation in learning	*"I am sure students will be more impressed with the subject…"*
	Interactivity is the best way of teaching	*"Students enjoy learning by doing."*
Professional Development	Cooperation between in-service teachers and pre-service teachers	*"…we can work together in producing something really good ..."*
	Feeling able to conduct a class using a computer lab	*"I will definitely use computer lab maybe for the final project..."*
Instructional Productivity	Flash could enhance helpful instruction	*"I believe that it will help me to increase my productivity in producing teaching module for instance… "*
	Flash will help them to integrate new teaching methods	*"I know that it is not easy to learn new technical stuff like programming but with the right way of teaching or technique, it will not cause any difficulties"*

Following coding, the frequency of each code was recorded and these are shown in Tables 4.6 and 4.7 below.

4.3.4 Results

Observation during the lesson showed varying levels of interest in the subject material as witnessed by their comments. It was also noted that the knowledge and skills in multimedia varied from student to student.

It was important to gain a picture of the attitudes of the respondents towards Flash Action Scripting and whether programming competence was considered by them as important for their future jobs. Therefore, each participant's viewpoints were examined qualitatively and are presented here. The quantitative data from the coding process is presented later in Tables 4.4 and 4.5.

4.3.4.1 Findings

There were positive and negative feedback received from the respondents in this phase of analysis. The following statements show the detail on the positive comments:

- ***Engagement*** - All of the respondents stated that using Flash Action Script could help capture attention from their students. Participants felt that the graphics, audio and video elements and the ability to provide animation and interactivity was useful (Kalganova 2001b; Routledge 2007).

- ***Enjoyment*** - All of the respondents believed that the use of suitable software such as Flash for developing teaching material would help them to enjoy learning the subject (Routledge 2007; Fitriza 2012). They agreed that if the subject was made more enjoyable in this way it would help concentration.

- ***Motivation*** – All respondents agreed that Flash Action Scripts held their attention and had the potential to motivate them in their learning (Kalganova 2001b; Fitriza 2012; Lin 2012).

- ***Interest*** – The students found the Flash Action Script software entertaining and they commented that it would be a positive instrument for teaching programming at beginner-level like them. They expressed strong opinions that the interactivity brought the learning material alive and that all pre-service teachers should learn how to use Flash Action Script. Most of the them (fifty respondents) claimed that they are quite interested in learning new technologies and are willing to take the challenges (Lin 2012).

- ***Confidence*** – Most of the participants, however, were positive about the use of flash action scripting regardless of the learner's background. However, two of the respondents disagreed that Flash Action Script should be learnt by students with a non-computing background (Lin 2012; Tsukamoto 2012) because they themselves considered it difficult to learn. From their point of view Flash Action Script is aimed at users with an ICT background.

- ***Creative Designer*** – The majority of the respondents agreed that the elements used in the software could be useful for trainee or in-service teachers to be more creative in their teaching or learning (L. Wang, Ertmer, and Newby 2004; Lin 2012). They felt that the animated characters could be designed easily without having to produce difficult code and so these could help them to produce exciting material for their learners.

- ***Professional Development*** - Respondents believed that an appropriate method of teaching Flash Action Script could help to enhance their future teaching careers. For instance they felt that teachers would feel more self-assured in their teaching (Lin 2012; Routledge 2007; Garofalo and Summers 2004; Duncan, Bell, and Atlas 2017) as they could make their material more interesting and this would give them a tool and confidence to get better results and hence progress professionally.

- ***Instructional Productivity*** - Respondents commented that using Flash Action Scripting would enhance their instructional productivity. They felt that if a module could be developed using this tool then their teaching of programming could be augmented with this as well as using more traditional teaching approaches, which would benefit their students (García et al. 2007a; Routledge 2007; Lin 2012; Fitriza 2012; Matthews 2014). Additionally, they felt that using this tool could enhance their own productivity when producing the teaching and learning material.

Besides the positive feedback received from the respondents, there were also some negative comments. Examples as follows:

- ***Boredom*** – Due to the difficulties in understanding the concept of programming, some respondents expressed their feelings of boredom in trying to learn programming. The feelings was so bored and hard (K. Koh, Repenning, and Nickerson 2013; Routledge 2007) moreover, respondents' feelings can be looked through their face expression during the session.

- ***Lack of Motivation –*** All of the respondents mentioned that they did not understand the basic concept of programming at the early stage of learning.

They just know about Flash on the day of testing phase. The feel so down and demotivation about learning how to program (Tsukamoto 2012; Martins, Mendes, and Figueiredo 2010; T. Jenkins 2002).

- ***Confusion*** – Some of the instructions given to them were so confusing especially when starting to works on more than one scene. Furthermore, some scripts were so hard to understand and make them confused on the lesson they had learnt (Sang et al. 2010). For example at the section on adding navigation scripting.

- ***Fear*** **-** The fear expression (Radošević, Orehovački, and Lovrenčić 2009; Balan and Kalavally 2012; Lin 2012; T. Jenkins 2002) had been expressed by the respondents due to the use of the software and computer itself. They were so afraid to even touch the keyboard and explore the software itself. However the fear feelings was decreased dramatically from twenty-seven number of frequency to zero number after the Study 1 had been completed.

- ***Frustration*** **–** Respondent felt less confident once they did not get what they expected from the task given (Kalelioğlu, Gülbahar, and Doğan 2017) or from what they had learnt. Eight of them felt upset when the lesson learnt did not met their target for instance they did not perform well even for the basic understanding on the programming concept. However, after the study 1 completed, only one of them still had that frustration feeling.

- ***Difficulty*** **–** As a first time person facing with the programming environment, coding looked so hard and difficult to learn (Matthews 2014; Lister and Leaney 2003; T. Jenkins 2002; Kalelioğlu, Gülbahar, and Doğan 2017). Without proper method of teaching, programming could become very difficult for the students to learn programming. All of them said that programming was so hard and difficult subject to be leant however, after they had learnt the task at Study 1, their perception on the difficulties in programming reduced to only 3 person were still said that programming is very hard to learn.

4.3.4.2 Analysis

Tables 4.4 and 4.5 below present the frequencies with which the extracted codes appeared in the data, before and after the tutorial, based on the responses from the pre- and post-questionnaires. Table 4.4 specifically presents the positive and negative codes discussed earlier and Table 4.5 presents the frequencies of codes related to professional enhancement. Differences in code frequencies that were considered particularly notable are highlighted in the text immediately following the tables.

Table 4-4: *Attitudes and Emotions Before and After the Module.*

Frequency Responses of Attitude Codes in Study 1				
	Before taking the Action Scripting Module		**After taking the Action Scripting Module**	
Codes	**Frequency**	Normalised	**Frequency**	Normalised
	Summary of Positive Codes			
Interest	**50**	1	**58**	1.16
Engagement	**39**	0.78	**53**	1.06
Motivation	**42**	0.84	**81**	1.62
Confident	**8**	0.16	**42**	0.84
Enjoyment	**23**	0.46	**41**	0.82
	Summary of Negative Codes			
Boredom	**14**	0.28	**2**	0.04
Lack of Motivation	**21**	0.42	**2**	0.04
Confusion	**59**	1.18	**23**	0.46
Fear	**27**	0.54	**0**	0
Frustration	**9**	0.18	**1**	0.02
Difficult	**58**	1.16	**10**	0.2

Of greatest note in this comparison is the large reduction in *fear* – from 27 responses (over half the respondents) before the tutorial reducing to zero such responses afterwards. Also of particular note is the corresponding increase in *confidence* of the respondents after the tutorial – 8 comments were noted prior to the tutorial and this number rose to 42 afterwards. There was also a general reduction in frequency of all

the negative responses after the lesson – an average of nearly 4 per respondent before the tutorial reduced to less than one per respondent afterwards.

Table 4-5: *Responses Relating to Professional Skills.*

Codes	Relating to	Frequency (Before)	Frequency (After)
Creative Designer	Animation in learning	20	32
	Interactivity is the best way of teaching	29	43
Professional Development	Teachers will feel empowered	12	27
	Feeling able to conduct a class using a computer lab	15	32
Instructional Productivity	Flash could enhance helpful instruction	14	36
	Flash will help them to integrate new teaching methods	21	42

It is clear from Table 4.5 above that the tutorial on programming in Flash Action Script increased participants' positive opinions of the tool's potential benefits to their future teaching careers. The frequency change in the *creative designer* codes illustrates that the facility for developing animation and creating interactivity notably impressed the students with an approximately 50% increase in the codes from before the tutorial to afterwards. Additionally, empowerment increased by more than double, as did *instructional productivity*. From this data it was evident that Flash Action Scripting could provide intrinsic value to their future lessons.

4.3 CONCLUSION

The respondents involved in this exercise had all chosen an elective that led to them to taking the lessons in Flash Action Scripting. It was assumed that this indicated that

they had an inclination to think the subject valuable, though they may have taken it for other reasons. However, the responses regarding professional enhancement possibilities tend to reinforce this view. The lesson they were exposed to was of a conventional nature (lecture followed by lab class) and no special attention was paid by the teacher to the area of Computational Thinking. Whilst the importance of developing CT skills when learning to program is widely acknowledged, as reported in Chapter 2, no effort was made to specifically correct any non-computational thinking during the programming lab. This phase was purely to determine whether Flash Action Scripting could engage and motivate students to learn to program.

It can be seen that the lessons were generally effective in improving student attitudes towards their ability to learn to program and in reducing the fear levels of the students. There was a reasonable understanding of the value of learning this subject to their professional skills and RQ1 asked whether understanding such will increase the participants' engagement with learning a programming language and thus help them to acquire CT skills. Study 1 has demonstrated that using Flash Action Scripting can enhance enthusiasm and engagement in learners, even those from a non-technical background. Of all the positive codes identified, all of them showed an increased frequency after exposure to the tutorial and all the negative codes showed a reduction in frequency. Participants' comments provided some useful illustrations of their views that Flash Action Scripting was a tool that could really help them learn to program as well as allow them to generate interactive multimedia for their own future programming lessons. Additionally, considerable benefit was seen in terms of professional development. Flash Action Scripting was considered a tool that could help them feel more empowered and make a positive difference to their teaching in terms of enabling them to use different teaching methods and the participants considered this as being potentially important for the progression of their teaching careers.

Study 1 tested the three objectives related to RQ1.

- To more formally identify attitudes to programming among teachers and trainees in particular.

It has shown that originally they had high levels of fear/anxiety

- To identify the challenges for them in learning to teach programming.

Which were identified as related to their anxiety etc.

- To determine to what extent they already think computationally and identify ways in which this might be helping or preventing them meeting the challenges of learning to program.

This final objective was less well addressed. Clearly their attitudes improved as they found they could make use of Flash Action Scripting. However, more detailed analysis of their behaviour in learning was needed to understand how this was preventing them meeting the challenges of learning to program. The changes in attitude were indicators that Flash Action Scripting would be a useful motivator for them to learn how to program and consequently to see if they can develop computational thinking. This therefore began to respond to the second research question:

RQ 2 Can those without a CT background learn to think computationally?

With the objective of trying:

- To understand how teachers might be motivated to learn to think computationally.

The next chapter moves on to further address the third objective from RQ1 as well as give further answers to RQ2 and begin to address RQ3. It presents Study 2 which further considers Flash Action Scripting as a motivator for this group and whether the presence or absence of Computational Thinking (CT) skills can be identified in learners and whether incorrect thinking with respect to CT can be corrected.

CHAPTER 5

STUDY 2

5.1 INTRODUCTION

The previous chapter provided some practical evidence in response to the first two objectives of RQ 1 and began to provide answers to RQ2 as it concluded that Flash Action Scripting would be an acceptable motivator for getting teachers to engage with understanding Computational Thinking (CT) as well as how to program. Respondents in the experiment had expressed positive motivation towards learning it. This chapter looks at the second study which was designed to address the third objective of RQ1:

- To determine to what extent they (teachers and trainee teachers) already think computationally and identify ways in which this might be helping or preventing them meeting the challenges of learning to program.

Also it seeks to look at the objective of RQ3:

- To work out how to detect use or lack thereof of the five CT domains: abstraction, algorithmic thinking, decomposition, generalization and analysis.

This is addressed in this chapter by describing Study 2 in which one on one lessons in Flash Action Scripting were given to a small group of learners with the same general background as in the first study. In this case because the lessons were presented on a one to one basis between the learner and the researcher it was possible for the actions of the learners to be observed and recorded as they were taught and completed exercises in using Flash Action Scripting to perform a set of tasks.

Flash Action Scripting is considered to be a language that is really helpful for beginners (Sands, Moukhine, and Blank 2008; Fitriza 2012; Crawford and Boese

2006). As was shown in Chapter 4 the attitude towards Action Scripting was generally positive – after having studied the subject. However, there was little understanding of Computational Thinking (CT) and the associated skills. This also showed up when they were observed in their lab classes after the lessons (though this data was not gathered formally). Based on these findings it was clear it could be possible that use of an appropriate standalone software to learn Flash Action Scripting could enhance computational thinking of those without a computing background (RQ2).

In order to achieve this aim it was first necessary to study in more detail the attitudes and behaviors of students trying to learn action scripting; in particular motivation to learn to think computationally and the presence or absence of CT skills. Identification of attitudes would allow requirements for a standalone tool (see Chapter 6) to be gathered and gaining an understanding of incorrect CT would enable a process for correcting these errors to be identified and taught (RQ3).

In a number of studies described in chapter 2 (refs) the method of measurement of the presence of CT was the ability to produce the desired artifact.

5.2 METHOD

5.2.1 Participants

Given that the goal of this research was to support Malaysian trainee teachers to develop CT skills, it was important to select a group of test subjects who fitted well with the target audience. The study was undertaken in the UK but participants were recruited as Malaysians studying teaching related degrees in the UK.

The University Of Nottingham School Of Education runs Post-Graduate Certificate in Education (PGCE) courses in education and admits a number of students each year from Malaysia. It also runs PhD and Doctor of Education (EdD) courses for professional teachers and a number of Malaysian students take these courses. It was thus decided to recruit a group of pre-service teachers from the PGCE course as well as a group of in-service teachers from the doctoral courses. The major difference

between the two groups was their age and experience. In addition at least one of the doctoral students was not intending to return to teaching on completion of their studies but rather to a management role.

Eight pre-service teachers and three in-service teachers took part in the experiment. It should be noted that the small number of in-service teachers makes any findings relating to differences somewhat speculative. The main objectives of this phase were for participants to prepare a fundamental lesson task for their learners using Flash Action Script (see Appendix 1C) and to observe their behaviours whilst they did so.

5.2.2 Experimental Design

Each individual participant was given a short course of one-to-one lessons on the use of Flash Action Scripting (see Appendix 1B), with each task designed to introduce one stage of Action Scripting. The first lesson introduced the controls of the interface. Clear understanding of these controls and the role they play is necessary for the correct building of scripts. However an additional goal was for users to be able to generalize this knowledge for use with other software in the future. As was seen in Chapter 2, the ability to generalize is a core component of CT (Dorling M. 2014).

The tasks involved participants following instructions given by a tutor in order to complete a defined task. Initially the tasks were very explanatory, with later tasks being more hands-off by the tutor. The tasks were printed on paper (see Appendix D) and the expected output (artefact to be constructed) from the completed task was also shown to the participant before they began the task (see Appendix B). The purpose of this was to allow the participant to have a clear understanding of what to do and to enhance their motivation to do so (Tsukamoto 2012; Lin 2012).

Participants were given questionnaires before and after the tasks. The purpose of these questionnaires (please refer to Appendix L and Appendix M) was to identify their level of motivation to learn to program. Additionally the post-exercise questionnaire asked about their experience of using Flash Action Scripting and whether they believed that its use would be of value to their teaching (Chang et al. 2012; Sang et al. 2010; Umar and Hassan 2015).

Participants' working task were analyzed by the researcher during the session. The Camtasia studio software was also used to capture the activities (individual actions) to allow them to be replayed after the exercise so that the researcher could check the observations taken live. Camtasia Studio was used to capture screen shots and function as voice recorder as well (Charnigo 2009). It was not used for any other function. The participants were also interviewed (please refer to Appendix M for the interview questions) at the end of one of the sessions. The same ethics process was carried out as in Study 1.

The next section describes the tasks given in more detail in order to illustrate both the intended progression of learning and how the tasks were designed to investigate CT thinking.

5.2.3 Tasks Design

A series of tasks were given with each task being based around a learning outcome. (That is the participant should have been able to complete a similar task for themselves by the end of the session.) Each task involved the participant producing a set of simple artifacts or animation demonstrations, building in complexity with each task (see Appendix B for examples what was being asked in the tasks). For particular artifacts it was acknowledged that not many of the teachers would specifically want to produce this particular artifact but all of them could see the value in doing so.

An example of the initial steps taken to develop a task are listed below and shown in Appendix D. The example of output that should be produced based on the following criteria can be seen in Figure 5.1.

- Produce a simple scene for the artefact.
- Add controls and add actions.
- Introduce more complex actions.
- Extend the process to produce a final complete product.

The purpose of such steps was to allow the extent of the capability of Action Scripting to be explored, but also to allow a number of CT skills to be explored (Dorling M. 2014):

- The task must be broken down into a suitable set of sub tasks (Decomposition).
- Action generalization can be seen as activities have many similarities (Generalisation).
- Algorithms are seen being developed in the Action Script codes (Algorithmic Thinking).
- These can then be extended as the complexity of the task increases (Abstraction).
- Debugging was discussed by asking respondents to solve the problem thus errors were detected and solved (Analysis).

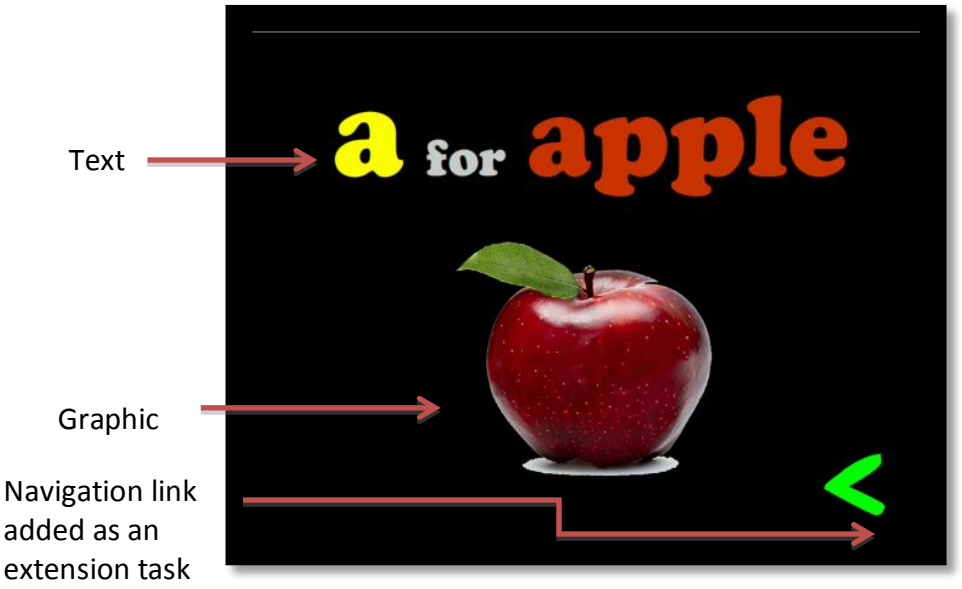


Figure 5.1: *Fundamental Step for Creating Text, Graphic and Navigation Link.*

When programming in Flash Action Scripting one of the main components of the interface is the popup window which shows the scripts produced by the system (see

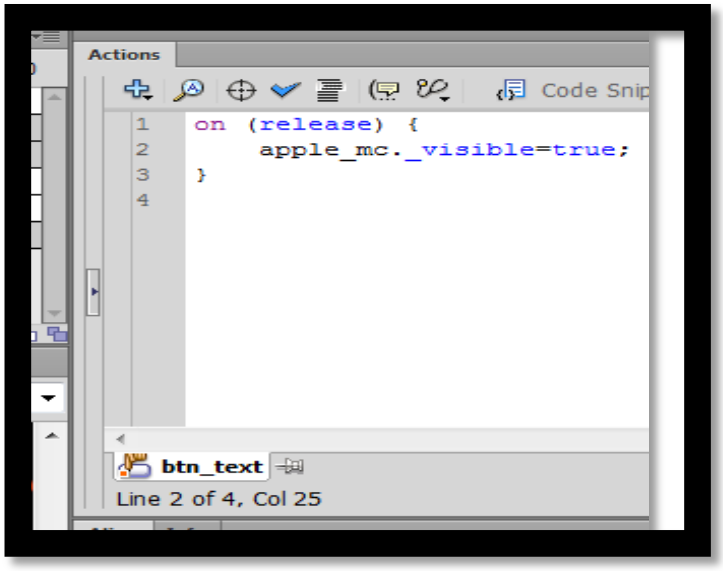


Figure 5.2: *The Action Script Screen Showing a Movie Clip Action for Animation*

Figure 5.2) and errors in syntax (Figure 5.3). In Figure 5.3 it can be seen that hints are displayed to allow the programmer to identify the location of syntax errors.

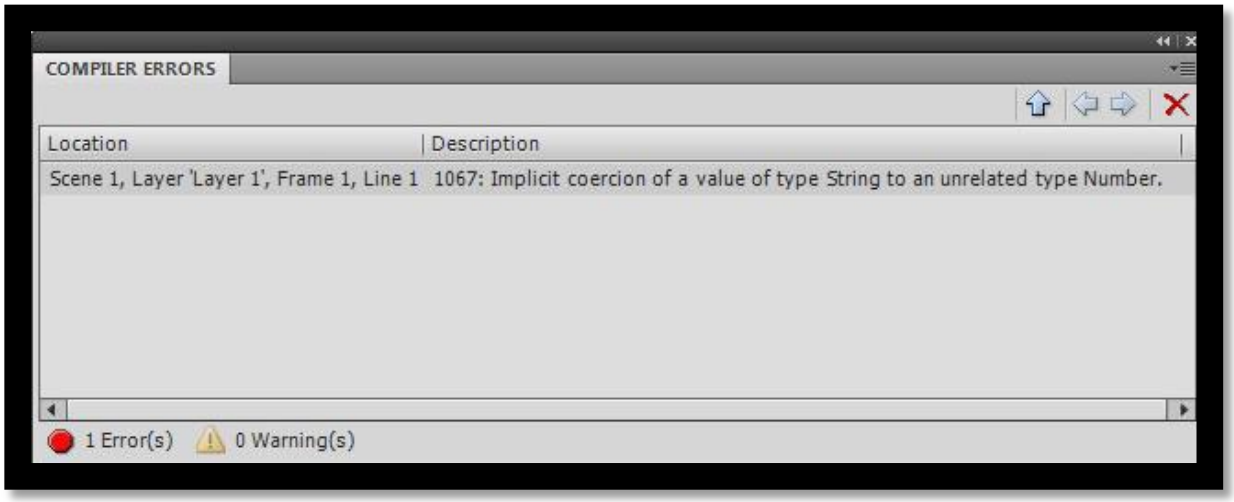


Figure 5.3: *The Compiler Error Log Screen*

Participants were exposed to the code at an early stage in order to reduce the "fear" they might feel at the idea of coding. Flash Action Scripting is considered to be a language that is really helpful for beginners (Sands, Moukhine, and Blank 2008; Fitriza 2012; Crawford and Boese 2006). One of the best ways to learn how to write Action Scripts is to read the code of an existing Flash project and then reproduce and

perhaps alter this code in one's own project (Sorapure 2006), thus learning by doing. Through the process of learning how to write programs it was envisaged that CT skills could be detected. Scripts such as the animation example in Figure 5.2 were used early on in the tasks in order to engage the user with something interesting to view and learn from.

The incorporation of animation (Diwakar et al., 2011; Sorapure, 2006) allows more complex activities to be coded but without requiring complex knowledge of coding. Use of animation and visuals can greatly aid the teaching of new concepts in programming (Routledge 2007) and Flash additionally provides a frame-based structure which makes it easier for inexperienced programmers to create ergonomic content (Reimers and Stewart 2007). The developed animations can be integrated into the code produced and can be used to create learning material for use by the teachers themselves when teaching others.

Figure 5.1 above shows one of the intermediate levels of tasks that participants were required to complete. Prior to this the activity was to add a script and pictures. Subsequent to this the activity was to add a navigation link to the graphic. Thus the task was decomposed for the user into sub tasks. As was discussed in Chapter 2, decomposition is one of the CT skills that helps programmers to simplify complex problems in order to solve them (Dorling M. 2014). It is a way of thinking about artifacts in terms of their component parts which can then be comprehended, answered, developed and evaluated separately. One of the aims of each lesson was to support the learning of this CT skill (LA Gouws et al. 2013; Grover & Pea 2013a).

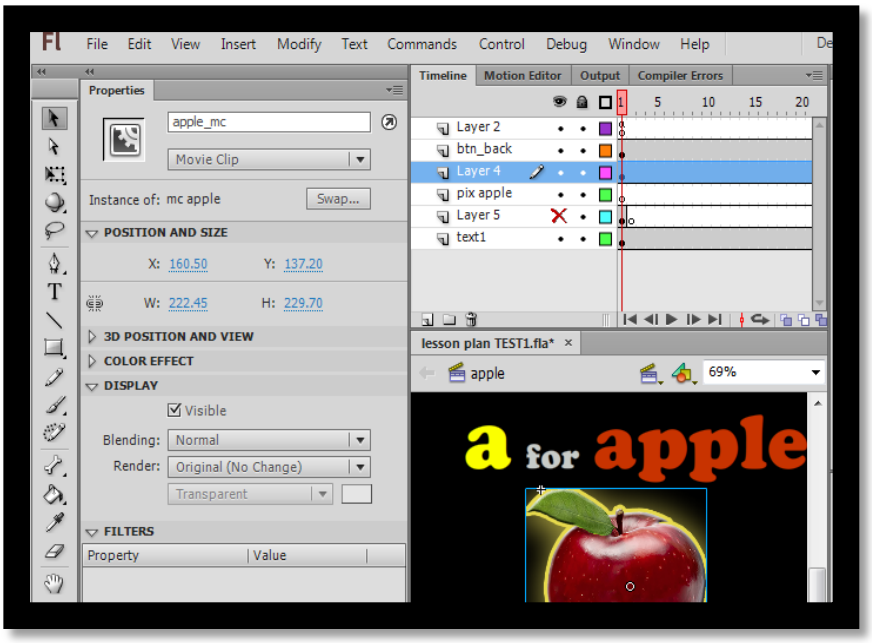


***Figure 5.4**: Complex Syntax for the Lesson Module.*

Figures 5.4 and 5.5 display more complicated coding that participants were exposed to later on in the task. As users progressed through the task it was intended that their algorithmic thinking would be developed and thus lead to persistence of this in thinking (Hu 2011). It was also intended that participants would develop stronger logic whilst learning to debug their code (*analysis* – CT domain) which in turn would boost their thinking skills (Van Dyne and Braun 2014; Brusilovsky and Calabrese 1997).

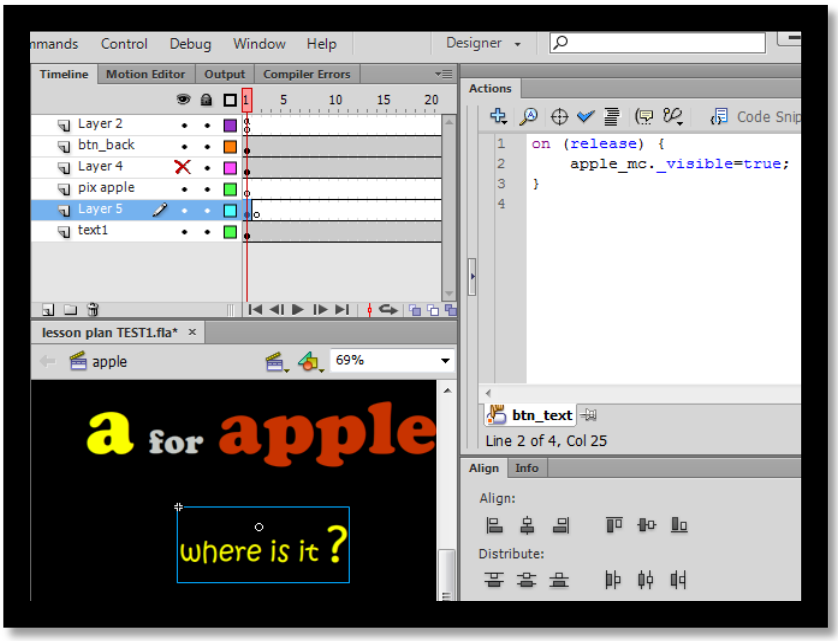


Figure 5.5: *Adding On Release Syntax for a New Function*

5.3 RESULTS

In this section the results of the pre- and post- questionnaires are presented followed by a discussion of the observation data and interviews. The activity in Study 2 involved 8 Pre-Service Teachers (identified as TT1 to TT8) and 3 In-Service Teachers (identified as ET1 to ET3).

5.3.1 Questionnaires

The questionnaire data were analysed and coded using thematic code analysis (Fereday and Muir-Cochrane 2006; Braun and Clarke 2006). Examples of participant statements before and after taking part in this activity are seen in Table 5.1 (Pre activity) and 5.2 (Post activity) along with their codes. The same codes as used in Study 1 were used in this stage so that differences in attitudes of the participants could be detected.

***Table 5-1**: Example Positive and Negative Responses from Participants before the Activity. The example for each quote was selected from respondents' feedback.*

Positive Quotes	Code
"I think this software really helps me in understanding the concept now..." (ET1)	Confident, Affordance
"...by learning this software we can produce a good teaching module for our students..." (TT5)	Motivation, Affordance
"I am confident now to start developing the advance stage of Flash Action Script..." (TT4)	Confident
"I want to learn this programming more and more so that I will come out with my own teaching modules soon!" (TT1)	Motivation, Affordance, Interest
"I knew that Flash can do interesting stuff..." (TT6)	Interest
"I am looking forward to learn and try something new today..." (TT1)	Motivation, Enjoyment
"I am ready to learn new lesson today!" (TT8)	Motivation, Engagement
Negative Quotes	**Code**
"I am not that confident because I am sure it will be very tough..." (ET3)	Fear, Difficult
"I'm sure it will be harder and so sorry if I can't finish the tasks for today..." (ET2)	Difficult, Frustration
"Are you sure today's lesson is not difficult for me?" (TT5)	Difficult
"More complex programming? I am not sure whether I can do it" (TT8)	Lack of Motivation

Table 5-2: *Example Positive and Negative Responses from Participants after the Activity.*

Positive Quotes	Code
"I feel more confident now..." (ET1)	Confidence
"Great lesson learnt today!" (ET2)	Enjoyment, Motivation
"I really enjoy the lesson, even though it is quite complicated and hard but..." (TT8)	Enjoyment, Engagement, Motivation
"I can say I am in love with Flash Action Script!" (TT2)	Motivation, Enjoyment
"I can now be a developer at my school..." (TT3)	Affordance, Motivation
"I am speechless because I wasn't thinking I can program at last…" (TT2)	Motivation, Affordance
"…exciting software…" (TT6)	Interest, Engagement, Enjoyment
Negative Quotes	**Code**
"Sometimes the scripts still confuse me..." (ET2)	Confusion, Frustration
"I have mixed up with some scripts… sometimes I feel so hard to follow…." (TT7)	Frustration, Difficult, Confusion
"I am not sure whether I can work on this independently after this session…"(TT6)	Lack of motivation
"I am afraid that when I move too much using mouse, the system will hang or something like that because…" (ET1)	Fear, lack of motivation
"I don't really like to do programming… quite hard for me as a business background teacher…" (ET3)	Lack of motivation, frustration

Some of the quotes recorded were given more than one code, as appropriate. The quotes shown came from trainee and experienced teachers alike. In the tables respondent ET1 means in-service teacher 1 and respondent TT8 means pre-service teacher 8.

The results in the tables below (Tables 5.3-5.5) give a comparison between the frequencies of codes in Study 1 and in Study 2. It can be seen that there was more interest and confidence overall in Study 2 (Tables 5.4 and 5.5) than there was for the

participants in Study 1 (Table 5.3). As was seen in Study 1, negative attitudes and responses were once again reduced during Study 2 (from an average of about 7 per participant to fewer than two per participant after the activity). Positive comments uniformly increased.

Table 5-3: *Frequency of Appearance of Code Phrases in Use by the Pre-Service Teachers and the In-Service Teachers* ***before and after*** *Taking Part in the Activity in Study 1.*

Codes	Frequency (before)	Normalised (before)	Frequency (after)	Normalised (after)
	STUDY 1			
	Summary of positive codes			
Interest	50	4.55	**58**	5.28
Engagement	39	3.55	**53**	4.82
Motivation	42	3.82	**81**	7.36
Confident	**8**	0.73	**42**	3.82
Enjoyment	23	2.09	**41**	3.73
Affordance	20	1.82	**28**	2.55
	Summary of negative codes			
Boredom	14	1.27	**2**	0.18
Lack of Motivation	21	1.90	**2**	0.18
Confusion	59	5.36	**23**	2.09
Fear	**27**	2.45	**0**	0
Frustration	9	0.82	**1**	0.09
Difficult	58	5.27	**10**	0.91

Table 5-4: *Frequency of Appearance of Code Phrases in Use by the Pre-Service Teachers (TT) and the In-Service Teachers (ET)* ***before*** *Taking Part in the Activity in Study 2.*

Codes	Frequency *(Pre-service Teachers) (TT)*	Normalised Frequency *(Pre-service Teachers) (TT)*	Frequency *(In-service Teachers) (ET)*	Normalised Frequency *(In-service Teachers) (ET)*	Total Frequencies	Normalised
Summary of positive codes						
Interest	52	6.5	22	7.3	**77**	**7.0**
Engagement	38	4.8	23	7.7	**61**	**5.5**
Motivation	62	7.8	14	4.7	**76**	**6.9**
Confident	29	3.6	15	5.0	**44**	**4.0**
Enjoyment	37	4.6	21	7.0	**58**	**5.3**
Affordance	31	3.9	18	6.0	**49**	**4.5**
Summary of negative codes						
Boredom	0	0.0	2	0.7	**2**	**0.2**
Lack of Motivation	0	0.0	2	0.7	**2**	**0.2**
Confusion	19	2.4	9	3.0	**28**	**2.5**
Fear	17	2.1	3	1.0	**20**	**1.5**
Frustration	0	0.0	1	0.3	**1**	**0.1**
Difficult	29	3.6	19	6.3	**48**	**4.4**

Table 5-5: *Frequency of Appearance of Code Phrases in use by the Pre-Service Teachers (TT) and the In-service Teachers (ET)* ***after*** *taking part in the Activity Study 2.*

Codes	Frequency *(Pre-service Teachers) (TT)*	Normalised Frequency *(Pre-service Teachers) (TT)*	Frequency *(In-service Teachers) (ET)*	Normalised Frequency *(In-service Teachers) (ET)*	Total Frequencies	Normalised
Summary of positive Codes						
Interest	59	7.4	24	8.0	**83**	**7.5**
Engagement	45	5.6	32	10.7	**77**	**7.0**
Motivation	54	6.8	27	9.0	**81**	**7.4**
Confident	52	6.5	27	9.0	**79**	**7.2**
Enjoyment	52	6.5	29	9.7	**81**	**7.4**
Affordance	47	5.9	28	9.3	**75**	**6.8**
Summary of negative Codes						
Boredom	0	0.0	0	0.0	**0**	**0.0**
Lack of Motivation	0	0.0	0	0.0	**0**	**0.0**
Confusion	8	1.0	4	1.3	**12**	**1.1**
Fear	4	0.5	2	0.7	**5**	**0.3**
Frustration	0	0.0	0	0.0	**0**	**0.0**
Difficult	5	0.6	2	0.7	**7**	**0.6**

Comparing Tables 5.4 and 5.5 directly, it can be seen that the frequencies of all of the positive codes in Study 2 increased after the activity and the frequencies of all of the negative codes decreased. The only exception to this was that motivation appeared to have reduced slightly in the pre-service teachers after the activity. This may be worth noting, but may have just been anomalous because the overall frequencies had risen slightly for motivation and the frequency for lack of motivation in the trainee teachers stayed at 0 before and after the activity. However, of

particular note is the increase in overall frequency, after the activity, of confidence (44 to 79), enjoyment (58 to 81) and affordance (49 to 75). Levels of confidence in programming had increased dramatically, with the frequency almost doubling, and levels of enjoyment and affordance had also increased markedly.

Looking at the negative codes in Tables 5.4 and 5.5 it can be seen that all non-zero overall frequencies had reduced after the activity, with confusion and difficult being dramatically lower. Frequencies of confusion dropped to less than half (28 to 12); frequency of fear dropped to half (4 to 2) and frequencies of difficult dropped to approximately one seventh (48 to 7). These reductions were considerable and demonstrated that use of Flash Action Script used in this way had a dramatic effect on both pre-service and in-service teachers. Examples of two of the comments related to the codes confusion and difficult are shown below with a short discussion of the implications.

Example of Confusion code extracted:

"I am really confuse with the syntax for instance, frame1. I am not really understand what is that mean by frame number 1." - Respondent TT3

The above quote clearly illustrates that Respondent TT3 was having difficulties understanding certain concepts and syntax. Such difficulties are known to cause students to give up trying to solve a problem (Sáez-López, Román-González, and Vázquez-Cano 2016) and capturing feedback like this is necessary to identify their difficulties and propose solutions (see Study 3).

In this circumstance the respondent was observed to copy and paste code from one part to another to try and solve the problem they were facing, but this approach was unsuccessful because they did not understand how the copied code would not apply to the current problem, meaning that they were *generalizing* incorrectly. This is discussed further in section 5.3.3 below.

Example of *difficult* code extracted:

"Personally, I feel very hard to follow the session because I cannot grasp everything within time period. If extra time will be given to understand and learn Flash, I think it should be better" - Respondent ET3

Although intrinsic motivation of a student has been identified as a critical success factor, careful design of instruction can help boost motivation amongst students (Balan and Kalavally 2012). This respondent clearly felt overwhelmed by what they were required to do given the time available, thus undermining their motivation. The quote shows that the respondent found it not only difficult but was also fearful of not being able to complete the task.

The questionnaire data showed that using Flash Action Script clearly increased confidence, enjoyment and affordance and reduced the difficulty and confusion participants experienced with programming. However, the study allowed further analysis of behavior in order to determine whether participants were using CT correctly (or incorrectly) and when. This was important in order to identify where participants might have been going wrong and how to try and remedy this. The study, therefore, were used to gain further understanding of what motivated participants and how, and also allowed the researcher to examine the presence or absence of correct CT.

All participants (both pre-service and in-service teachers) were novices to programming and many had very little experience of using computers. Several of the participants appeared unsure and nervous at the beginning of the session. One participant even expressed shock when asked to participate, one initially declined to participate but then later agreed. However, once the sessions had started several participants moved from appearing anxious to asking lots of questions and then appearing to understand and enjoy the activity. As each session progressed the researcher observed large changes in many of the participants' attitudes, with many expressing interest in how Flash Action Script could be used for their own lesson development.

A few participants appeared confused at the beginning and sat doing nothing. However most of these soon began to grasp some of the concepts and began to realize that the software was not as “scary” as they had first thought.

5.3.1.1 Motivation to learn programming

As was mentioned in the previous section intrinsic motivation to learn a subject will be affected not only by a participant’s enjoyment and interest in that subject but also by how difficult they find that subject (Seligman and Csikszentmihalyi 2000).

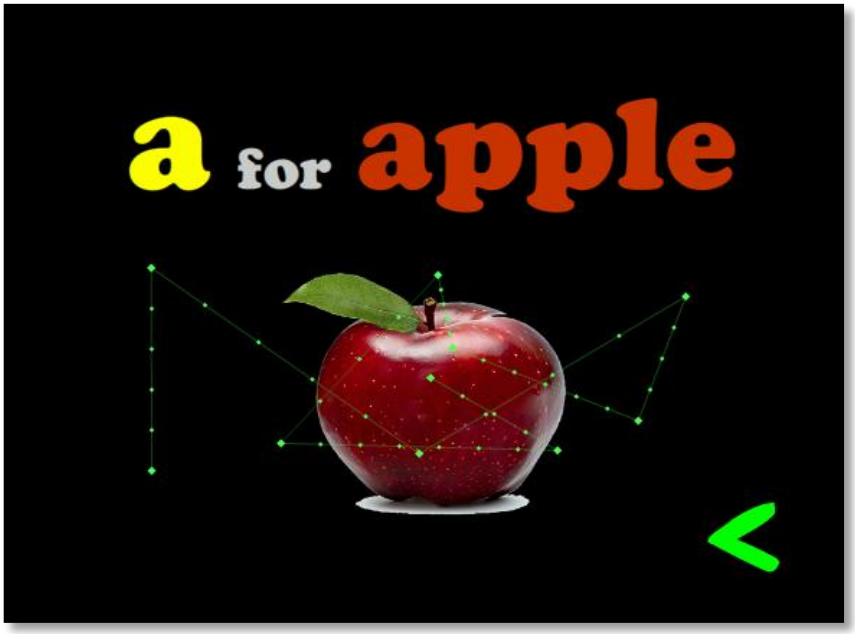


Figure 5.6: *Learning About Animation through Flash Action Scripting.*

During the observations several respondents appeared nervous of trying the technology but nearly all of them seemed ultimately to overcome this to some extent and progress to creating simple animations. Figure 5.6 shows one of the ways of producing an animation by adding a path for the animation object to follow.

Participants clearly used their creativity and understanding of programming concepts to design a module and expand the related screens. They were observed to engage with the activity and persist when difficulties arose which gives evidence of

sustained motivation to learn how to program. This approach of “not giving up” meant that participants were constantly evaluating and re-evaluating the situation, constructing ideas as they learnt (Holzinger and Ebner 2003b; Ainsworth 2006).

5.3.1.2 Detecting incorrect thinking

Correct use of CT involves a programmer breaking down a problem into smaller problems (*decomposition*) and understanding how to solve those problems algorithmically (*algorithmic thinking*). Programmers also need to be able to *abstract* from one problem to another; in other words to understand how one problem might stem from another; also programmers should know how to debug a problem and solve it (*analysis*) and to *generalize* so that they can apply a previous solution correctly to another problem.

In this study it became apparent that CT was not being used by many of the subjects. One set of behaviours that was frequently observed in the lessons was that of copying and pasting code rather than understanding the given algorithm and adapting it to the new problem (see Figure 5.7). This clearly showed that wrong – non-Computational Thinking had taken place.

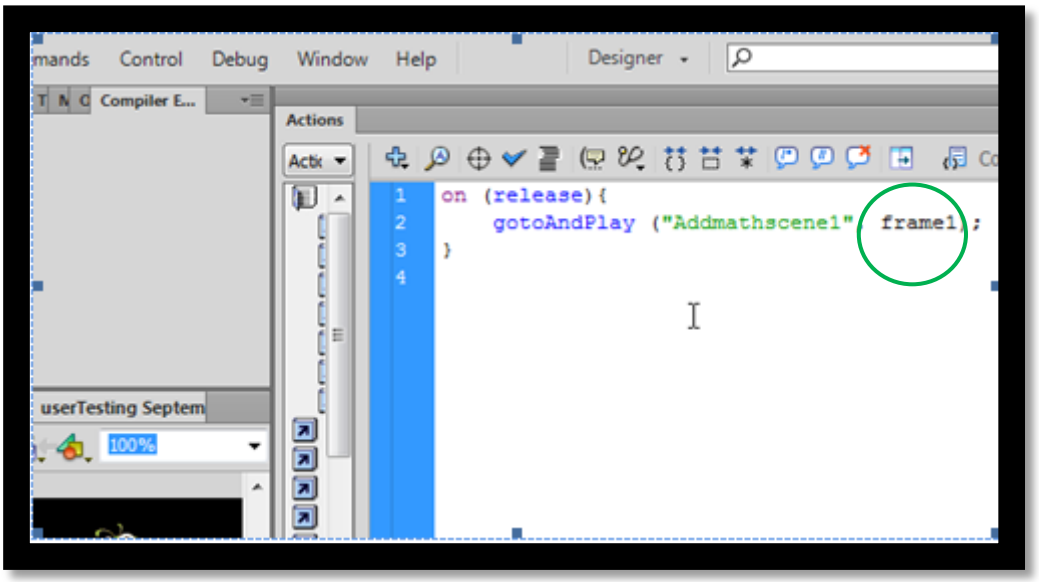


Figure 5.7: *Screen Captures on Wrong Thinking in Producing Scripts.*

Figure 5.7 gives an example piece of code produced by a participant who had copied and pasted from a previous section of code when trying to solve a problem. The correct version of the script should specify '*gotoAndPlay ("Addmathscene1",1)* 'and not "frame1". However the respondent copied the syntax from the example given demonstrating that they did not understand the function of the script or the concept of parameters in coding. Errors like these were able to be spotted where the output captured using the Camtasia software was analysed (the details about how it works had been explained at chapter 3.3.2.) The majority of the respondents showed this lack of algorithmic thinking and incorrect generalisation.

One advantage of using Flash Action Scripts is that any errors that occur are highlighted by Flash itself, at the place where the error occurs (see Figure 5.8). This is in contrast to other programming languages such as Java Script (Routledge et al. 2007; Sorapure, 2006). Whilst such error flagging seems a simple idea, it is important for a learner because the ease of tracing the errors can have a significant effect on the motivation or demotivation to continue with the task at hand.

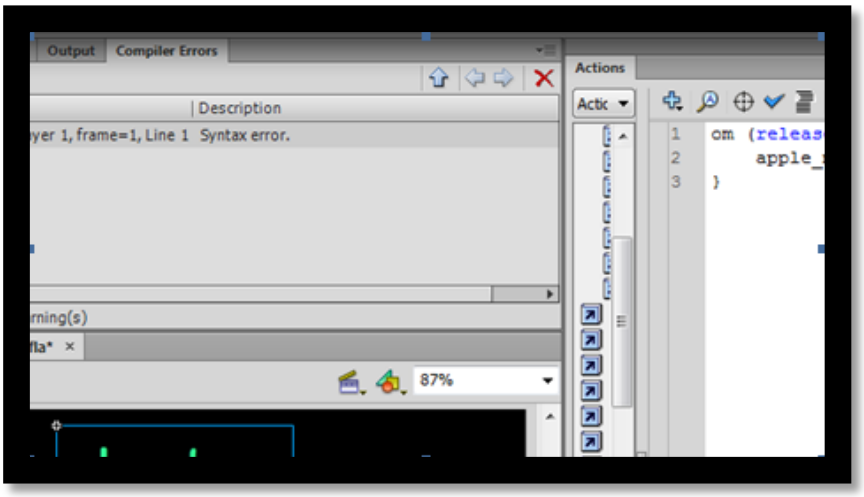


Figure 5.8: *Error Screen captured due to Copy and Paste Attitude.*

5.3.1.3 Adoption of CT skills

An important consideration for the observation sessions was also to examine not only whether and how incorrect CT was occurring but also whether and when correct CT was used. Some participants did demonstrate use of CT after an initial period of learning. At the beginning several participants clearly did not understand some important concepts such as parameters and the passing of variables, but by the end had grasped these concepts and were using them, demonstrating the beginnings of *algorithmic thinking*.

Another example is shown in Figure 5.9. Here the level of confidence of one of the respondents in the subject had increased as shown by his initiative to create additional buttons ('*Economics*' button) to explore more features later in the experiment. This positive signal was really encouraging and engaging. By looking at the layers created by respondent ET2 in Figure 5.9, it shows that the *algorithmic thinking* he had acquired was developing as was his understanding of *abstraction*. He organised his work systematically and effectively by putting different objects that appear on stage at different layers (Routledge, Aminaei, and Benachour 2007). These tasks were extremely important when it came to the revision or correction phases. The amendments would easily be done if the layers had been properly structured.

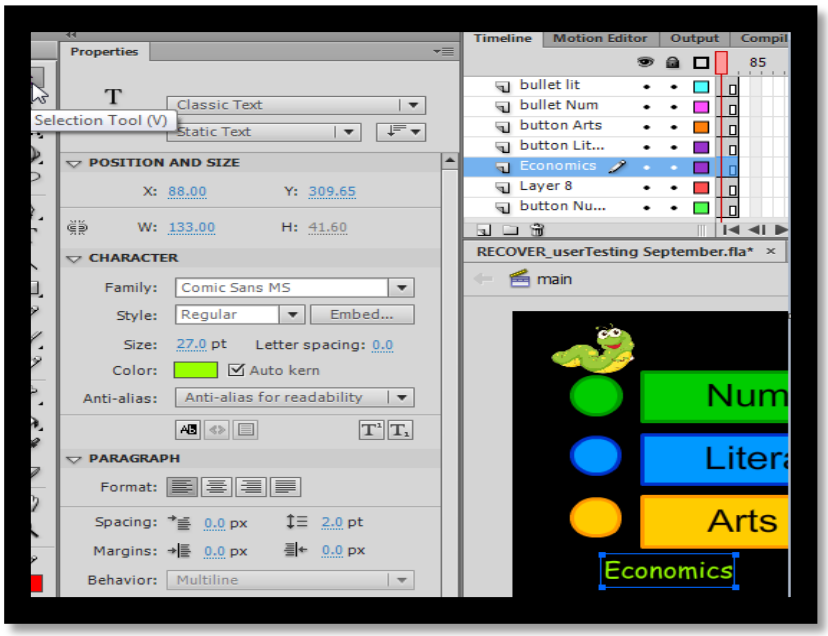


Figure 5.9: *Increasing Confidence Level and Improved CT Skills (Respondent ET2).*

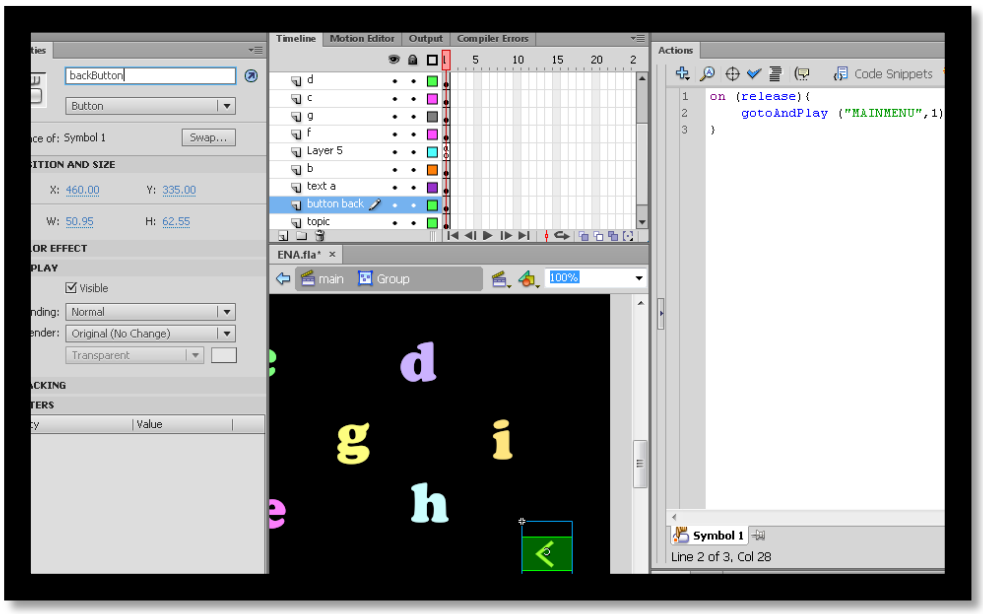


Figure 5.10: *Good Understanding of the Concept of Abstraction (Respondent TT2).*

Figure 5.10 shows another example demonstrating that some respondents started to understand the concept of abstraction. In this lesson the participant had grasped how the layers were organized and how screens would act and navigation links that could be integrated in the lesson module. Even though at this stage of study the five

domains in CT were not being fully seen, some of it already could already be seen by the end of the session.

5.3.2 Interviews

Once the data were collected and the interviews had been transcribed, it was coded to identify key themes. To avoid bias, the codes and themes were not only analysed by a researcher alone, but also from two other trusted persons in order to get the valid and reliable data (Fereday and Muir-Cochrane 2006; Strauss and Corbin 1994). The persons were selected not from the same area of study and they did not know about the respondents' background and also any details about the study. Therefore, the analysis and results are reliable for this study (G. W. Ryan and Bernard 2003). After that, then the final results will eventually gathered to get the vigorous data The analysis of the data focused on the key aims of the research that is in gaining an understanding of incorrect CT would enable a process for correcting these errors to be identified and taught. As discussed earlier, the data were coded to identify the key aspects which were then grouped to form themes. Voogt (2016) argues, there is a need to make room in teacher education programs for computer science specialists who can teach at least basic notions of CT.

Following each session the participant was interviewed about their experience of using the software. Several of the participants expressed enthusiasm for the software and acknowledged the benefits of using Flash Action Script in designing and developing their lesson material. They also stated that they were looking forward to the next session. Four of the participants (with backgrounds in mathematics, accountancy, economics and management and business, respectively) explicitly stated that they felt Flash would make their learning material more interesting for their students. They expressed that use of the software would provide more variety in teaching style for their students and would be more stimulating (Lin 2012). Another participant described its use as "value added" for their teaching. However two participants stated that time restrictions would prevent them from using or learning the software. They stated that it would add a burden, even though they could see the benefits of making interesting learning material, especially for teaching

"dry" subjects. Three participants commented on the benefits of learning Flash Action Scripting for their teaching careers and two subsequently asked the researcher for a copy of the software.

5.4 DISCUSSION

Overall, the results of this phase were very encouraging. Participants demonstrated incorrect use of, or lack of CT skills, but also the development of CT skills, by using Flash Action Scripting to create the desired animations. Most participants appeared nervous to start with (Martins, Mendes, and Figueiredo 2010; T. Jenkins and Davy 2002) but several were able to overcome this and openly appeared to enjoy the process. The software engaged participants and levels of motivation to try and program were improved after the participants had used it for a while. Based on the tasks and the responses of participants RQ2 can be confirmed; Flash Action Scripting can enable those without a computing background to develop CT skills. Identification of where participants were going wrong and using incorrect Computational Thinking was possible and this was most apparent by their copying and pasting behavior (E. M. Saari 2014) showing that before training they were not thinking algorithmically or generalize to other problems. A process for correcting these errors had begun to be identified and taught in answer to RQ3 as participants were able to learn from their incorrect CT and correct their mistakes because they understood why they were mistakes.

Based on the experiments in this study the requirements for the stand alone tool could be defined. Thus the next stage of this work was to develop and build that tool which could be used standalone to teach programming and CT. The tool would use Flash Action Scripting as this had been confirmed to stimulate interest, engage and motivate users to work through the problems set (García et al. 2007b; Lin 2012; Fitriza 2012). The ease of creating animations without requiring complex coding (Heins and Himes 2002; Crawford and Boese 2006) has been seen to encourage participants to persevere as they were rewarded with interesting and fun content once they succeeded.

There are other tools available which provide a simple interface, as seen in Chapter 2, for example Scratch (Utting, Cooper, and Kölling 2010; Brennan and Resnick 2013; B. Y. Resnick, Brennan, and Kafai 2009; Franklin et al. 2016) or Alice (Utting, Cooper, and Kölling 2010; Kelleher, Cosgrove, and Culyba 2002; Linda et al. 2009; Kelleher, Pausch, and Kiesler 2007; Wong, Yatim, and Tan 2016). However researchers have found that regardless of the lower barriers presented by these simplified visual environments some respondents still become unable to proceed further when asked to move on to more demanding problems (Csizmadia et al. 2015). Additionally, it was seen in Chapter 2 that environments such as these are considered suitable for children, but less so for adult learners (Cooper, Dann, and Pausch 2000; Daily, Leonard, and Jörg 2014; M. Resnick et al. 2009).

Other systems exist that use tools to create visual content which might be more suitable for adults. Examples of this include those which allow the integration of multimedia content in learning material (Lyons, Reysen, and Pierce 2012; Ronchetti 2010). However this study is concerned with teaching and developing skills rather than simply dealing with teaching principles and theories. This research is aimed at adapting Computational Thinking (CT) skills (Yadav, Hong, and Stephenson 2016) and not just learning how to create content by integrating multimedia elements. Instead the aim of creating that content is to harness an individual's intrinsic motivation to be able to develop their own learning material, which they themselves can benefit from using in their lessons (L. Wang, Ertmer, and Newby 2004; Lin 2012). Additionally the resultant development of CT skills and ability to program can also be of benefit to their teaching career, and this intrinsic motivation was apparent in several of the participants.

5.5 CONCLUSION

This stage of the research was principally carried out to answer the objective of RQ2:

Can those without a CT background learn to think computationally?

It identified where Computational Thinking was lacking in the pre-service and in-service teachers who took part in the process. A lot of knowledge had been gained in

Study 1 about the presence of wrong thinking and the reluctance to take part in learning Action Scripting and programming. Study 2 allowed this to be properly tested in a more controlled environment and opportunities for developing CT to be identified as well as places where CT was lacking. The main problem was again seen to be the sidestepping of the actual processes of CT with a tendency instead to attempt to copy and paste rather than to work out how to abstract the core of an algorithm and then generalize it. The opportunity to see algorithms developed was clearly useful in stimulating thinking amongst participants although one of the in-service teachers did drop out of the process. His reasons for this were apparent in that, as stated earlier on in this chapter, he was planning on no longer teaching. As a result the activity failed to motivate him. Of the others the majority had gained a lot of confidence and gradually could be seen to be using aspects of CT as they clearly decomposed the more difficult tasks allowing them to then generalize these and apply them later. As a result comments relating to difficulty fell dramatically, as seen in section 5.3.1.

The next chapter describes Study 3 which was carried out to address the research questions:

RQ 4 Can pre-service teachers who have adopted CT demonstrate this by applying it to undertaking “programming” tasks?

And

RQ 5 Does the use of a suitable tool in learning create users who will be motivated to go on programming?

CHAPTER 6

STUDY 3

6.1 INTRODUCTION

The previous chapter described Study 2 which involved teaching respondents to learn to program Flash Action Scripts through a one-to-one lesson set, combined with completing a set of tasks. The purpose of this phase had been to identify the design concepts for standalone software that would provide a self-learning tool to teach the programming of Flash Action Scripting. Study 2 identified ways in which a failure to use Computational Thinking (CT) had hampered the participants' learning. As the lessons were taught and problems encountered in doing the task the researcher observed the process and recorded what errors took place and the result produced. It was thus possible to design the standalone system to guide the participants to go through the process on their own. This chapter describes the system design and implementation and testing. One of the major objectives was to see whether participants could use the skills they had learned and apply them in using a different piece of animation software without having to be taught to use this specific software. This would be a strong indicator of their having learned to think correctly. The study therefore continued to include a stage where participants were invited to use a separate software tool for a similar objective – producing an artefact for a lesson.

In the study subjects were given incomplete code and were required to complete the program using their own creativity and thoughts. The ultimate goal for the study was to encourage learners to write pseudo code to solve problems then translate those pseudo codes into Action Script programming. Indirectly, the results would indicate whether the computational thinking skills were being applied or not. In addition, this study was conducted to provide a better understanding of CT for educators to support those who want to integrate CT into their own classroom content, teaching practice, and learning.

This chapter is structured as follows. Section 6.2 describes the design of the automated Flash Action script tool, including the design of lessons within it and how these were tailored to teach individual CT skills. Section 6.3 describes the methods used for studying the use and effectiveness of the tool. The results are split into two sections. Section 6.4 presents the results of the process of learning that the participants went through whilst using the tool. These results include observation and interview data. Section 6.5 presents the results for the final intended outcomes of the tool. In other words, it discusses whether the tool was ultimately successful in enabling participants to learn CT skills and whether they were able to move on to a new and different tool and transfer their learned skills to design new material with this tool. Section 6.6 draws conclusions from the work in this chapter.

6.2 TOOL DESIGN

The teaching tool for this study was developed using Adobe Flash Action Script software. In Chapter 2 the suitability of various tools was discussed and several researchers have argued that Flash Action Script software is particularly suited to novice programmers. Additionally, there is evidence of the suitability of Flash for instructors who wish to develop course material (Chang et al. 2012). In Study 2 evidence was presented that the pre-service and in-service teachers responded well to the use of Action Scripting. In this section the design of the standalone tool is described and following sections describe the experiments conducted to evaluate the effectiveness of the tool in terms of its ability to allow learners to learn CT on their own.

6.2.1 Architecture of the teaching tool

During Study 2 (Chapter 5) points at which the participants made errors were identified in order to come out with a precise teaching artifact and hence to motivate those existing respondents and future learners to learn to program. The tool was designed so that a user would pass through different stages, as shown in Figure 6.1 below:

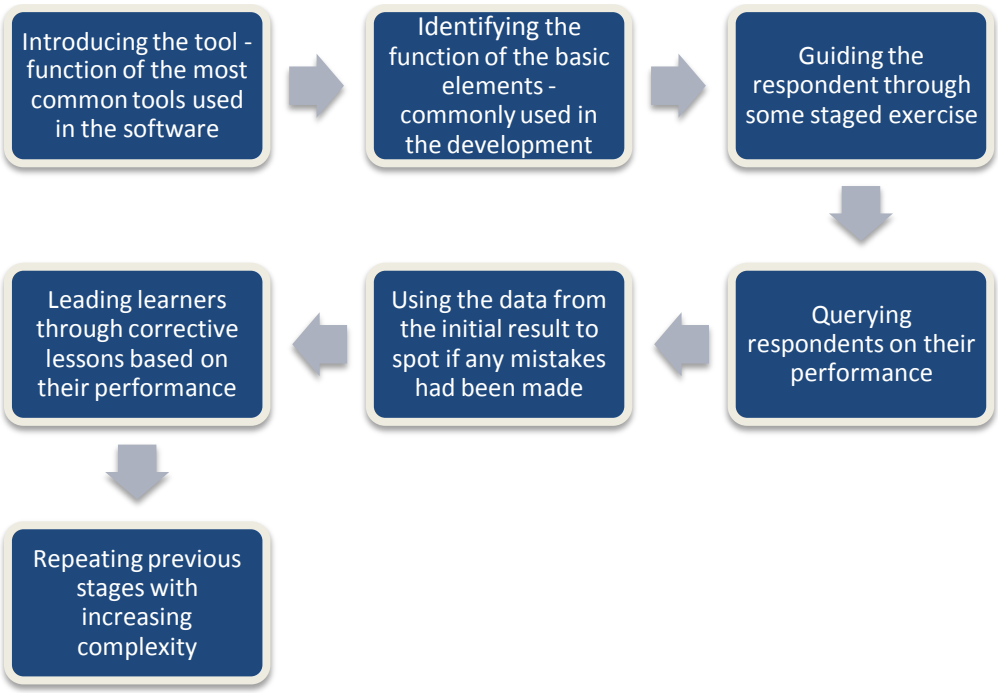


Figure 6.1: *Design and Basic Flow Using the Tool.*

Each block in Figure 6.1 above corresponds to a coded element in the tool. It takes the user through from launching the tool and the Flash Action Scripting software. Each step in the teaching has an associated exercise to be completed in Action scripts. Some templates are originally provided and these must be extended to complete the artifact that the students are designing. They are required to think through the sequence that their artifact will go through and produce a storyboard of the flow of activities – thus engaging them in algorithm building through an appropriate form of pseudo coding for this activity. This also allows them to break the task down into a set of steps thus decomposing the tasks. From these steps generalization can be developed once the original codes have been written. These all relate to aspects of Computational Thinking (CT) as described in Chapters 2 and 3.

6.2.2 Prototyping of the Tool

Before Study 3 began prototyping took place over a period of approximately five weeks. The prototype was tested on experts in designing multimedia user systems (Computer Science PhD students). They worked with the prototype for up to an hour after which they completed a questionnaire as shown in appendix C. Once a stable

version of the software had been designed the actual study began. The main objective of the prototype testing was the validation of the language and functionality of the system along with the design approach, more than evaluating the tool itself.

6.2.3 Design Tool

In this section the design of a lesson is described in order to illustrate the use of the tool. Camtasia Studio was used to capture the screenshots that are shown in the figures below. When the tool is opened, the home screen is displayed, as shown Figure 6.2.

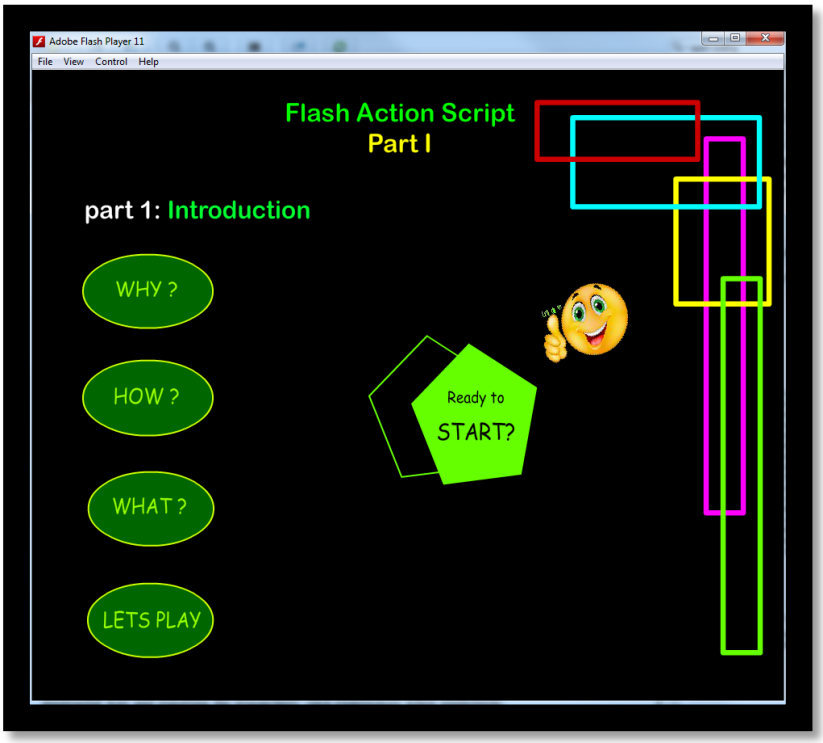


Figure 6.2: *The Home Screen.*

The design of the home screen is based around the theory of engaging students in CT, as argued by Mark Dorling, the CPD coordinator of Computing At School (CAS)

(Dorling, 2015). Dorling described the difference between the traditional curriculum and the curriculum when teaching CT, as discussed in Chapter xxxxx

The aim, therefore, of each lesson is to tackle the "Why" questions first followed by, "How" and "What" and to complete this by switching to the exercise to be followed by the participant. This is an introduction to thinking in the right order – starting with why the artifact is being made followed by how it will be done and then what will be done.

While working with the automated tool the aim was that the respondents would try to develop their own artifact using guidelines provided in the tool. Every lesson scene introduced programming functions and users were given hints and solutions for the relevant topics.

At the end of each lesson step there was a related exercise for the participant. On completing the exercise they were required to compare their product with the model given by the tool. An example of this is given below in Figure 6.4.

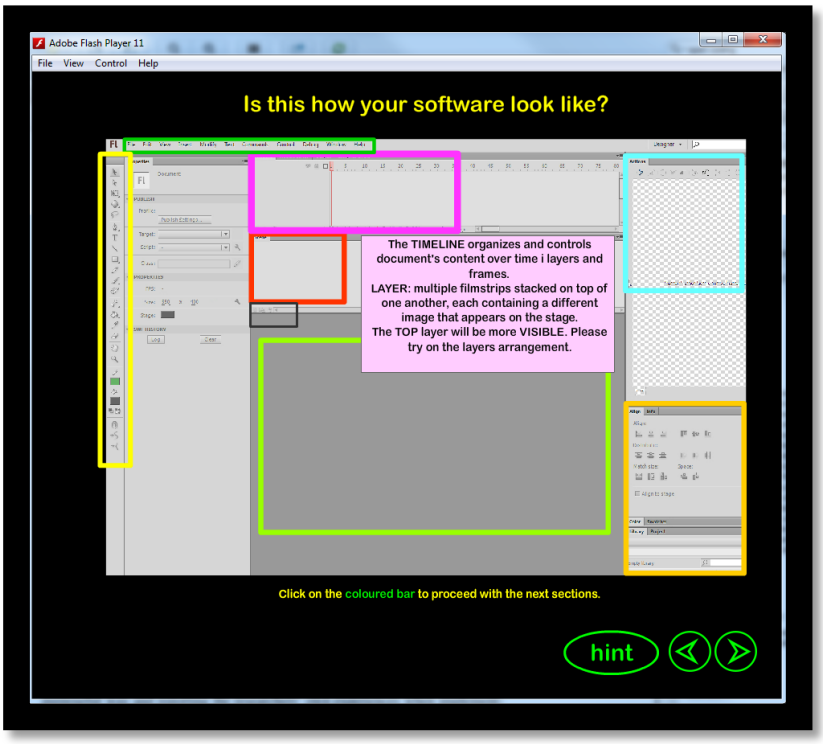


Figure 6.3: *Enquiry Screen from Training Tool. The Place Where the Participant Clicked Revealed Any Problems They Had.*

6.2.3.1 Design for Developing Computational Thinking Skills

Different aspects of the tool were designed specifically to relate to the development of particular CT skills. The following description and figures show how the corresponding screens aimed to adapt CT skills and section 6.5 shows some examples of screens that were developed by the respondents when using the tool.

- Layer screen (abstraction) (Figure 6.5) – At this stage, respondents would learn how to organize layers and the importance of labeling and controlling layers. Encouraging use of a systematic format at this stage was aimed to get participants into good habits for when they got to the correction or amendment stage of development. Each layer should represent each different object on each screen (Routledge, Aminaei, and Benachour 2007). "*Once you realize that computing is all about constructing, manipulating, and reasoning about abstractions, it becomes clear that an important prerequisite for writing (good) computer programs is the ability to handle abstractions in a precise manner*."

(Kramer 2007). Participants were encouraged, using this mechanism, to think about an object's role and then abstract from this to produce other objects elsewhere with similar roles.

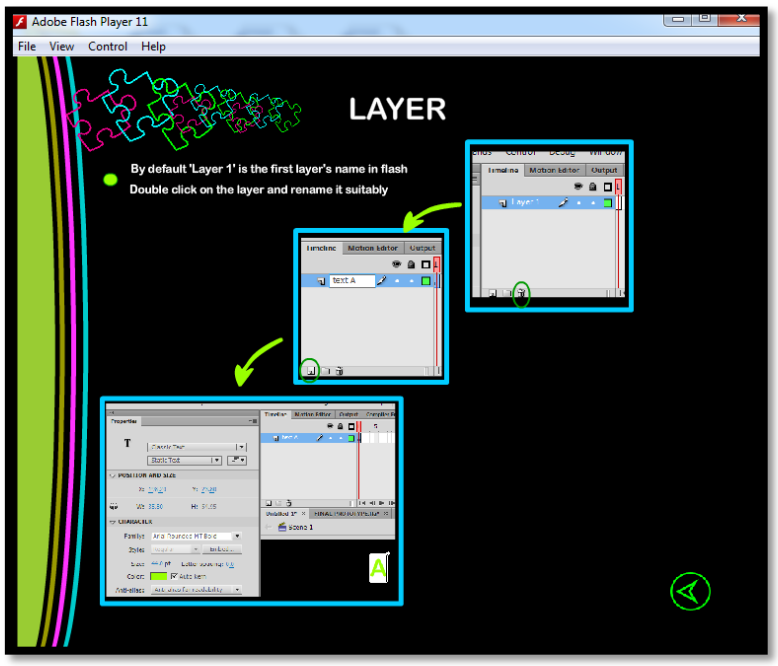


Figure 6.4: *Layer screen - Adapting Abstraction Domain from CT Skill.*

- Navigation screen (decomposition) (Figures 6.6 and 6.7) - At this screen, respondents were asked to learn about how to break problems into sub-tasks (D. Wang, Wang, and Liu 2014; Guzdial 2008; J. M. Wing 2006a). In this particular lesson tool, they were required to design a few related scenes so that the navigation buttons would be the main component that linked to each screen. An example of the navigation link from one scene to another can be seen in Figure 6.7. Participants therefore were required to think about the overall problem and break it down into smaller steps, due to the need to provide navigation between these steps.



***Figure 6.5**: Interaction Screen - Adapting Decomposition from CT skill.*

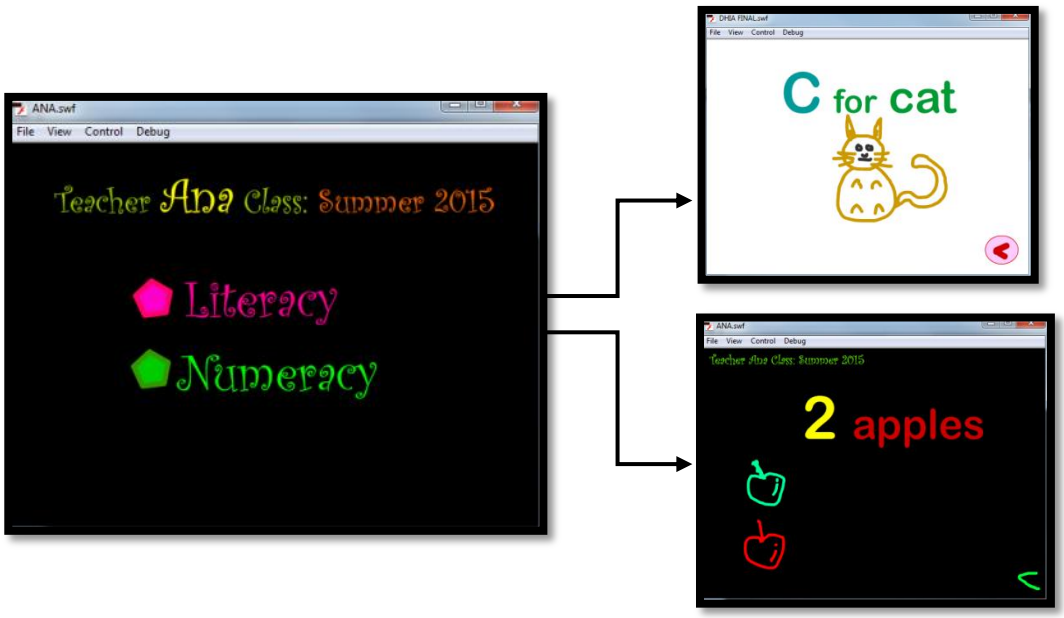


***Figure 6.6:** Navigation Screens.*

- Switch Case Statement (algorithmic thinking) (Figure 6.8) - As the lessons progressed the activity being taught became more complex and introduced more programming concepts. Figure 6.8 below shows the "switch case" statement. This coding exercise would test the understanding of more complex coding based

on the basic knowledge they gained earlier. Hence, algorithmic thinking is encouraged at this stage of the task. Respondents were required to use their algorithmic thinking skills to design the switch case statement based on the module given as shown in Figure 6.8. The use of this construct forces the participants to break down the problem to find a route through the problem and to map it out in code.

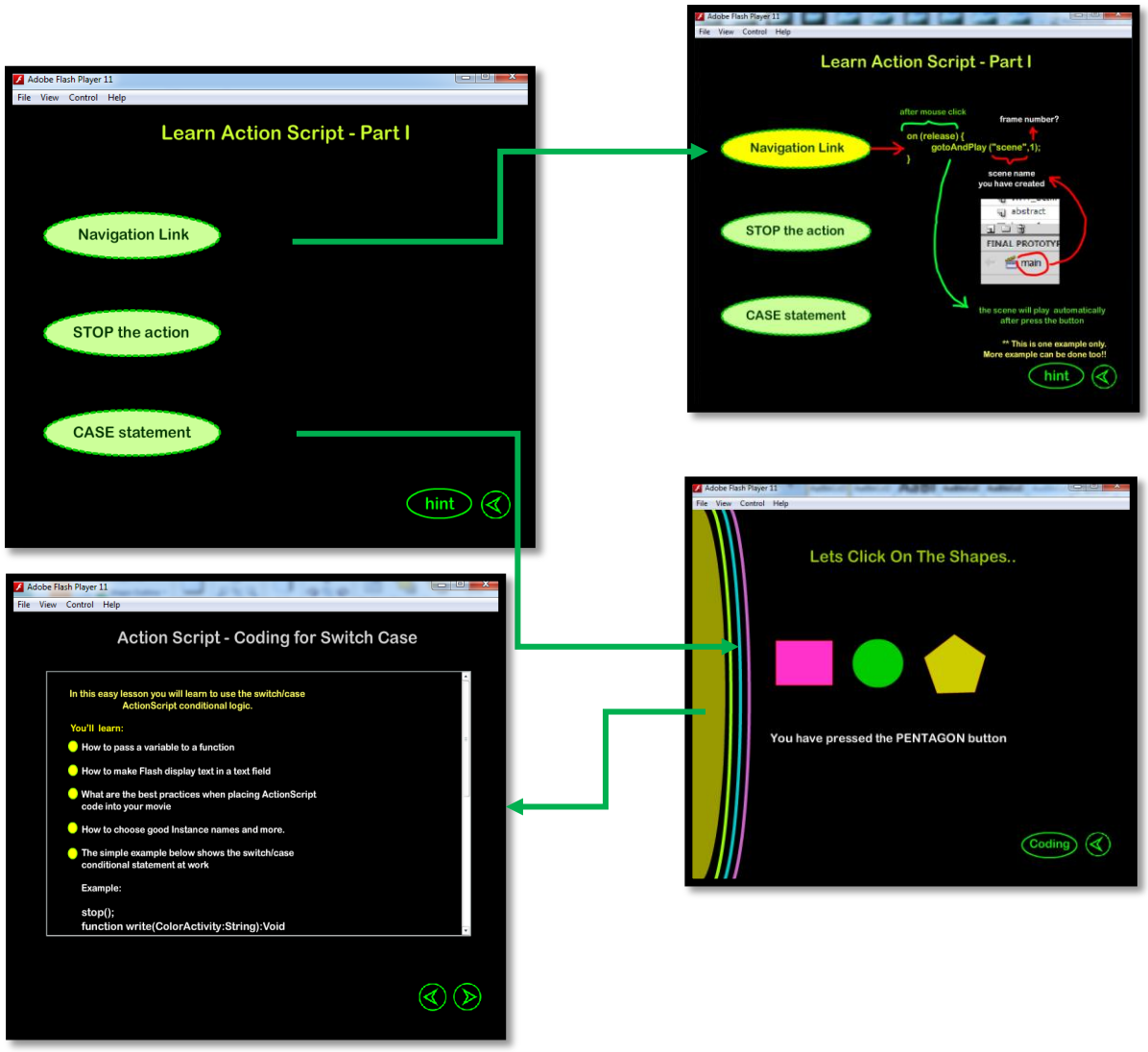


Figure 6.7: *Lesson Tool Switch-Case Screen*
Algorithmic Thinking Skill

- More complex coding (generalisation) (Figure 6.9) – Once participants had mastered the previous skills the automated tool provided them with more complex problems and problems which brought all the ideas together.

Participants then developed the ability to generalise from one problem to another. Figure 6.9 shows an example of artefacts which required more complex coding, as produced by one participant later on in the course.

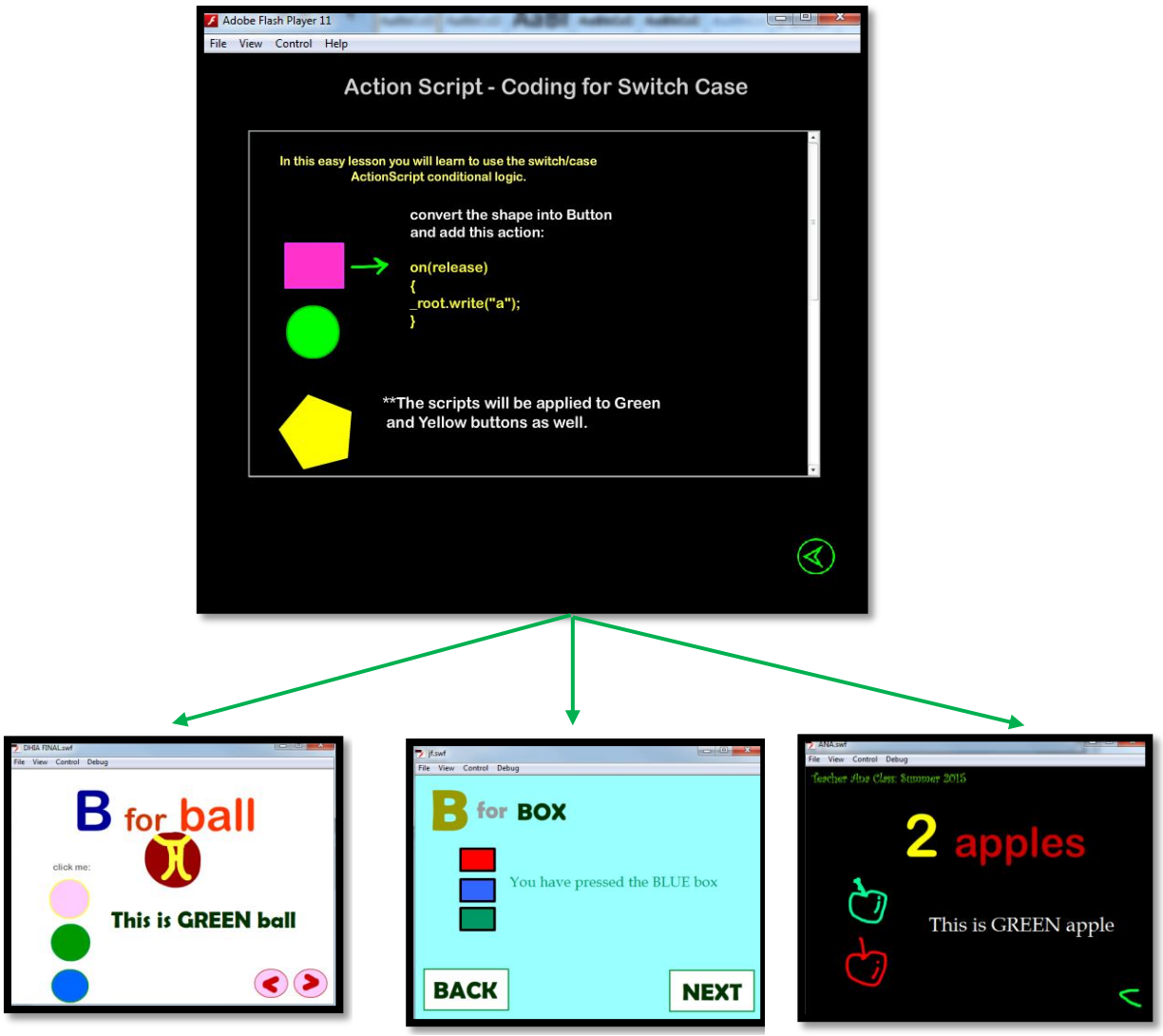


Figure 6.8: *A Respondent's Skill in Developing a Complex Coding Page*

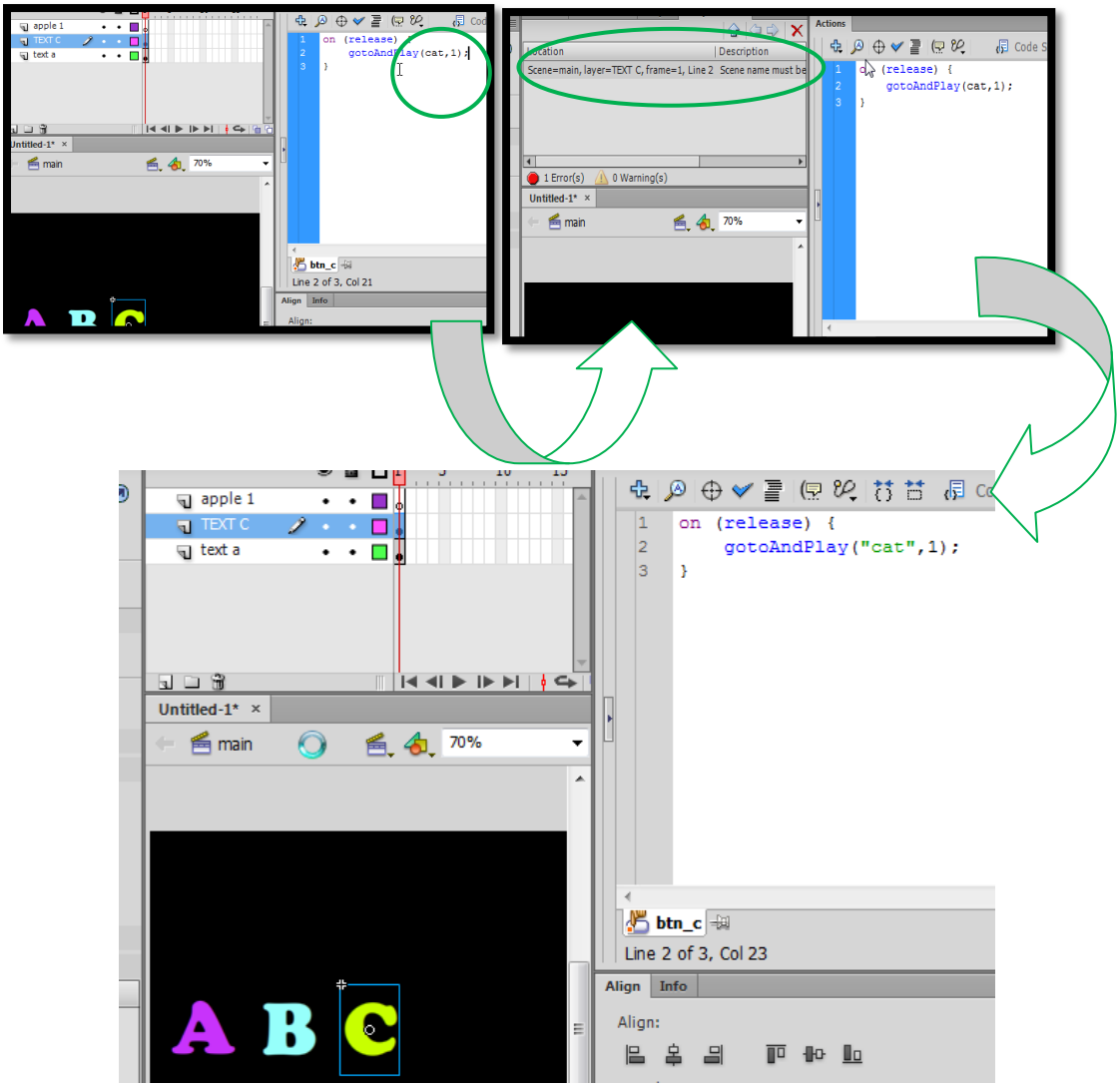


Figure 6.9: *Analysis – debugging errors*

- Debugging problems (analysis) – is about finding out what is wrong in a program and fixing it. Figure 6.10 above shows how a respondent (TT5) for instance, faced a problem and tried to fix it. The software (Flash) itself gave hints to the user in order to solve the problem(s). Getting users to take responsibility for thinking through their algorithms and code, to identify and fix errors is an important part of learning to think, and work, like a programmer (https://community.computingatschool.org.uk). The debugging process encourages independence for the user.

6.3 PILOT TEST

This study was conducted with five (5) respondents from the University of Nottingham. Three of them were from PGCE trainees programs here whilst the remaining two were from postgraduate students (existing teachers) from Malaysia who further their studies at School of Education University of Nottingham. Firstly, a respondent was told to develop a small task (Appendix A1). The session started with the brief explanation about a task, then the respondents started to develop a task using the tool. The respondents were informed of the need to run the tool and at the same time start to develop their task without any guidance from the researcher. When a respondent had completed the given tasks (which took between thirty to forty-five minutes), they ran their code to see their final product. The tasks includes creating texts, interactive buttons, colourful graphics and programming tasks using simple codes. The following sections discuss the results gained from the pilot study prior conducting Study 3.

6.3.1 Pilot Test Findings

Table 6-1: *Positive Responses from Participants after the Pilot test (The example for each quote was selected from respondents' feedback.*

Positive Quotes	Code
"I think this software looks interesting for me as a first timer and I love at it..." (PT1)	Interesting, motivation
"...It is very nice and I am sure student will like it software but..." (PT2)	Motivation, interest
"I think I can do this later because I like it..." (PT3)	Confident, interest ,motivation
"This tool look so engaging especially for students who don't like the specific subject like maths or history." (PT4)	Motivation, Affordance, Interest
"I don't believe that I can do programming even though not that much..." (PT5)	Affordance, motivation, interest
"I am looking forward produce something later..." (PT1)	Motivation, Enjoyment, interest

“It is so exciting to so this task. I am now feel confident using this tool even though I don’t know anything about programming before.” (PT4)	Motivation, Engagement
Negative Quotes	**Code**
“I feel quiet hard for certain parts ..." (PT3)	Difficult
"I’m not really understand about the scripting at first, but when looking at the other piece of paper, slowly I learn and get them right..." (PT5)	Difficult, Lack of motivation
"Sometime I feel quiet confuse…?" (PT1)	Confusing
"Sometime I feel hard to do it because I am not good in using software or learn programming…” (PT3)	Lack of Motivation, difficult

Table 6-2: *Code Frequencies After Completion of the Pilot Study.*

Codes	Frequency *(Pre-service Teachers)*	**Normalised Frequency** *(Pre-service Teachers)*	Frequency *(In-service Teachers)*	**Normalised Frequency** *(In-service Teachers)*	Total Frequencies	**Normalised**
		Summary of Positive Codes				
Interest	28	9.3	13	6.5	**41**	**8.2**
Engagement	21	7.0	16	8.0	**37**	**7.4**
Motivation	27	9.0	15	7.5	**42**	**8.4**
Confident	23	7.7	11	5.5	**34**	**6.8**
Enjoyment	24	8.0	16	8.0	**40**	**8.0**
Affordance	9	3.0	12	6.0	**15**	**5.0**
		Summary of Negative Codes				
Boredom	0	0.0	0	0.0	**0**	**0.0**
Lack of Motivation	0	0.5	1	0.5	**1**	**0.2**
Confusion	2	1.0	3	1.0	**5**	**1.0**
Fear	0	0.0	0	0.0	**0**	**0.0**
Frustration	0	0.0	0	0.0	**0**	**0.0**
Difficult	1	3.0	2	1.0	**3**	**0.6**

Of utmost note in this comparison is the summary of positive and negative codes received from the analysed data – from 5 respondents (3 of them were pre-service teachers while the remaining 2 were in-service teachers) which summarize the themes and codes from the respondents. Participants felt that the use of multimedia elements such as colourful images, texts and interactive buttons were interesting and would be fascinating to users (Kalganova 2001b; Routledge 2007). It is clear from Table 6.2 above that the tool created in order to increase motivation to learn programming by adopting CT had improved participants' perceptions towards programming that's is quotes from Respondent PT4 from Table 6-1. - *"This tool looks*

so engaging especially for students who don't like the specific subject like maths or history". It supports the points *(motivation, affordance and interest)* that the use of suitable software such as Flash for developing teaching material would aid them to appreciate learning the subject (Routledge 2007; Fitriza 2012).

The respondents found the tool was so entertaining due to the design and approach used to attract users. It looks so interesting thus the total number of frequency is the second highest after motivation theme that is 41 as compared to 42 for motivation. This tool was suggested to be applied to school so that the use of technology is adopted in education (Lin 2012).

There were 40 frequencies for the code type Enjoyment - All of the respondents believed that the use of this tool help them in learning the subject in a great and lovely environment (Routledge 2007; Fitriza 2012). They agreed that if the subject like mathematics and history would be practice in this interesting approach, it will definitely give more concentration for the students.

Motivating code were counted as the highest frequencies of all the positive codes that is 42 (and the normalised value is 8.2). All respondents really supports that the use of a tool could attract the students to learn programming or any other subjects as well. They (students) are not only to become a user, possibility to become a developer as well (Lin 2012).

Even though there still had feedback on the difficulties in learning to program, however, the confident level of the respondents still high that is 34 total frequencies (23 from the pre-service teacher and 11 from in-service teachers). Moreover, all of the 5 respondents from non-computing background users, they agreed that Flash Action Script should be learnt by non-computing background users (Lin 2012; Tsukamoto 2012).

Their effort and affordance towards a tool was positively accepted by the respondents. When affordances are taken advantage of, the user knows what to do just by looking: no picture, label, or instruction needed." (Norman 1988, p.9). The tool and functionality of objects inside helps them a lot for instance the navigation

link or clickable items were user friendly. From their point of view this tool was not confusing in terms of terminology used and user-friendly (Lye and Koh 2014).

Looking at the negative codes in Tables 6-2 it can be seen that all non-zero overall frequencies had filled up two columns that are “confusion” and “difficult”. The rest of codes were coded as zero value on the “’lack of motivation”, “fear” and “frustration”. Frequencies of confusion and difficult were detected based on the experience of the respondents as well. Looking at the number of frequency for both codes, they were slightly higher frequency numbers came from in-service teachers compared to the pre-service teachers. These values showed that the age and current profession were the factors that determine these because pre-service teachers aged from 21-24 and in-service teachers aged from 32-43 years old. The pilot test will definitely help researcher to upgrade and improve the quality and feedback of the respondents and to summarize the results gain at the end of the study 3.

Example of Confusion code extracted from one of the respondent:

"I am really confuse with the use of layer and frame name or frame number. Why I really need to add new layer every time I start with new object." - Respondent PT3

The above quote obviously illustrates that Respondent PT3 was having difficulties understanding certain concepts and syntax. Such difficulties are known to cause students feel less motivation to continue their tasks (Sáez-López, Román-González, and Vázquez-Cano 2016). Pilot studied fulfil a range of important functions and can provide valuable insights for other researchers (van Teijlingen and Hundley 2001). Pilot study would also try to identify potential practical problems in research procedure therefore for this pilot study, it really helps researcher to get the feedback from respondents on the practical session and the result shown none of them experienced any problem on the practical issue. Researcher described a pilot study on teaching computational thinking by cultivating the capacity for problem-solving skills of students using Flash Action Script software.

6.4 EXPERIMENTAL METHOD

Eight pre-service and three in-service teachers respectively from the PGCE and doctoral program at the University of Nottingham were enlisted. These were the same participants as those used in Study 2. The post-trial evidence from Study 2 was used as the pre-trial evidence for Study 3. Observations of participants using the automated tool were conducted and these were followed up with interviews.

Each task lasted between 25 and 45 minutes, depending on the complexity of the task. Each task involved the development of a complete program. Initially this involved the drafting of a storyboard (5-10 minutes) before beginning to program. Participants used the tool without any guidance.

Finally, after completing all lessons using the standalone Flash Action Script tool participants were given the task of developing lesson material using a different tool (Anime Studio). The aim of this was to see if they had developed correct CT skills, including the ability to generalize their understanding to another programming tool. During this session they were able to refer back to the automated tool for help.

The analysis of the information gathered took place manually from interview data using thematic codes as explained in Chapter 4. The interview data was manually recorded using Word List and Key Words in Context (KWIC) as supported by Ryan & Bernard (2003). In order to appreciate what people are talking about, look thoroughly at the words used. For instance in this study, in-service teachers and pre-service teachers very often used “interesting”, “motivate”, “engage”, confident”, “enjoy”, “afraid”, “demotivation”, “difficult”, “boring” and “confusing”. To generate word lists the researcher first identified all the unique words in the texts and then counted the number of times each occurred. The researcher also identified all the unique words in the text and then counted the number of times each word occurred (Braun and Clarke 2006). The more the same concept occurred in a text, the more likely it was that it was a theme. Eventually general themes were found and analysis continued to generate the basic theme.

Codes discovered for Enthusiasm and Attention are displayed in Table 6.1. Codes with example quotes for thinking skills are given in Table 6.2 and those relating to professional practice consideration in Table 6.3. These thematic codes were once again based on the combination of pre- and post-activity data and the quotes given in the tables are for participants in Study 3 only.

These codes were considered particularly pertinent to the research because they allowed the researcher to analyze the effectiveness of the tool in terms of engaging participants as well as developing their CT skills. Additionally, motivation to use the tool would be affected not only by a participant's engagement and ability to complete the task at hand but also by their perception of how learning these skills would impact their professional practice and prospects for enhancing their teaching careers. It was important that their intrinsic motivation was harnessed as much as possible if the tool was to be successful as Lin (2012) stated in his article. In addition, "*Intrinsic motivation has emerged as an important phenomena for educators—a natural wellspring of learning and achievement that can be systematically catalyzed or undermined by parent and teacher practices*" (Ryan & Stiller, 1991). Because intrinsic motivation results in high-quality learning and creativity, it is especially important to detail the factors and forces that engender this compared with those that undermine it (R. Ryan and Deci 2000).

6.5 RESULTS – PROCESS OF LEARNING

In this section the results of Study 3 are presented, including examples of the tasks completed by participants. The observation data is presented first followed by the thematic analysis of the interview data.

Observations

Prior to using the standalone tool some participants were fearful of having to develop their artefacts using the tool but without human intervention. However, after beginning the work, participants appeared to relax. Each participant worked through the lessons at their own pace and the lessons were tailored to their

individual needs by the system. Examples of work produced by the participants are shown in Figures 6.11 and 6.12 below.



Figure 6.10: *Different Screens Designed by Two Respondents.*

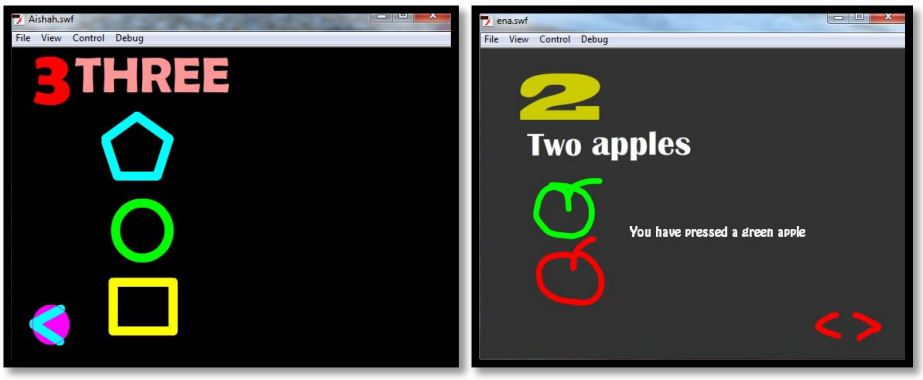


Figure 6.11: *Participants Successfully Attempt to Use Switch Case Syntax.*

Each participant could choose their own task and way of implementing the solution. There were participants who did all the tasks based on what had been assigned whilst one participant performed additional tasks. This participant wanted to explore more technical material based on her understanding of Flash Action Scripts demonstrating that she had begun to adapt her CT skills along the journey from Study 2 to this current study.

However, at this stage of the experiment three out of the eleven respondents still did not organize their layer formats correctly. They were observed to put all artefacts together in a layer (text, graphic and animation) which demonstrated a lack of

abstraction. Whilst this might not cause any problems for the immediate project because the task that was assigned was a very small task, this would not be appropriate for larger real-life projects that may be conducted in the future.

The tool did, however, appear to successfully allow participants to develop their CT and go on and produce more complex material. One example of this comes from respondent TT2 who was a Social Arts student with no background of computing or indeed any technical skills. The participant clearly found the tool very engaging and motivating. She showed great enthusiasm in learning how to program using the standalone Flash Action Script tool. Once the tutor had given the lesson, taking around 45 minutes, the participant went on and produced her own artefacts which took an additional 30 minutes. Figures 6.13, 6.14 and 6.15 display some of the screens produced by this participant independently.



Additional Text Links Created by Respondent ET2.

Figure 6.12: *Additional Navigation Links Created by Respondent.*

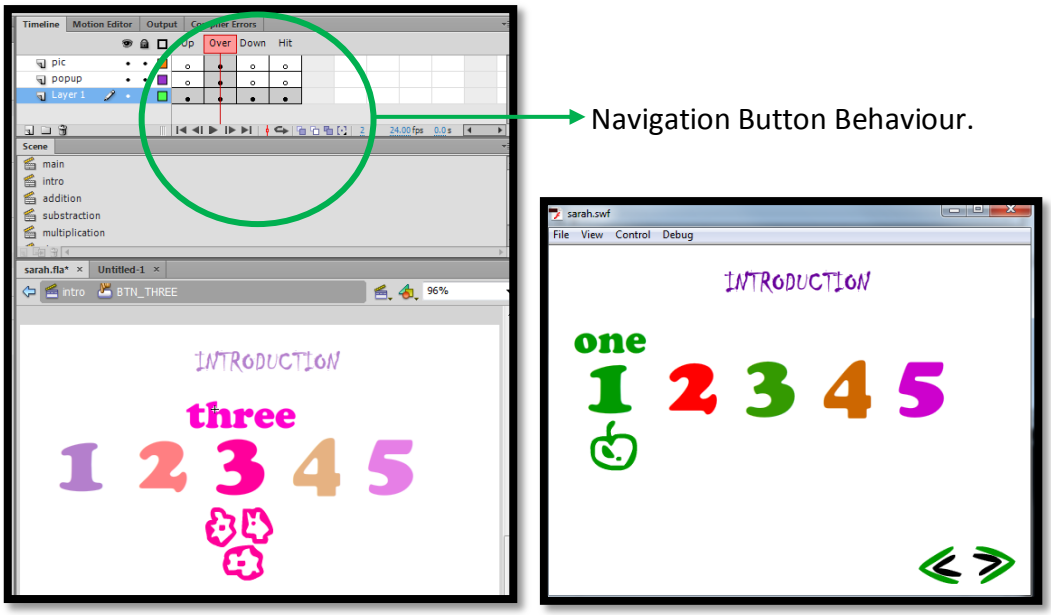
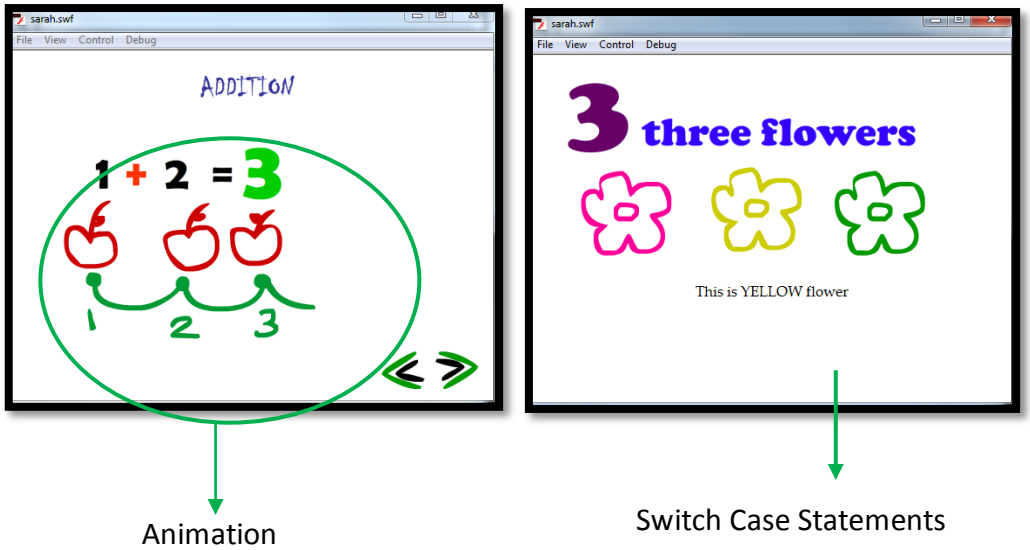


Figure 6.13: *Additional Commands Had Been Practiced.*

Figure 6.14 shows that an additional command was used to add further actions for the navigation buttons at the top of the screen. The participant added features including *'On Mouse Up', 'On Mouse Over', 'On Mouse Down'* and *'On Mouse Hit'*. These additional actions were explored by respondent TT2 because she wanted to create something different stemming from what she had learnt before. This demonstrated that the tool encouraged her to think more creatively and she was eventually successful in running these actions.



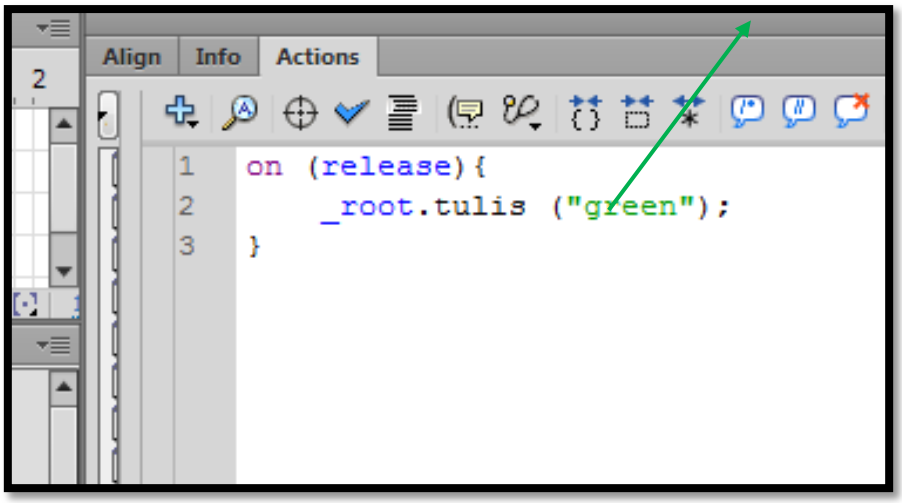


Figure 6.14: *Additional Feature and Technical Skill Developed from Respondent (TT2).*

From this respondent's feedback, it was clear that, by giving the correct methods for teaching, the motivation to learn programming can be improved (Kofoed and Reng 2010; Kalganova 2001a; Tsukamoto 2012) and CT skills can be adopted by someone without a computing background (J. M. Wing 2008; QuickStart Computing 2014; V. Barr and Stephenson 2011).

Interviews

The interviews provided a rich source of data (Turner 2010) and the coded data are shown in Tables 6.1-6.3 below. Table 6.1 focusses on codes which fall within the Enthusiasm and Attitude theme. These codes are important for representing participants' levels of engagement and motivation (Osman et al. 2012; Wong, Yatim, and Tan 2016) to use the tool. Table 6.2 shows codes related to the CT skills used by participants, and hence provides a measure of the level of development within this theme. Table 6.3 examines professional development (Boulton 2017; Duncan, Bell, and Atlas 2017; Yadav et al. 2014) which, as mentioned earlier, is important as a motivating factor for the participants since their perception of how use of the tool and learning the taught skills might impact their career is significant .

Table 6-3: *Enthusiasm and Attitude Codes from Study 3 Responses.*

ORGANIZA-TION CODE	*OPEN CODE*	*PROPERTIES*	*QUOTES*
Enthusiasm and Attentive	**Engagement**	Successfully complete the tasks given	*"I think I have improved a lot in the capacity of doing the tasks…" -TT4*
		Willing to focus on the tasks given assuredly	*"I feel that I can just concentrate my work without any help or interference from tutor anymore!" -TT5*
		Feel competent in making Flash to support the subject taught	*"Although I am not that good using latest technology, I can achieve my targets and produce many excellent works …" -TT5*
	Interesting	Learning to program is cool	*"It's really cool tool to be used later by us." TT8*
	Motivation	Pleasure to learn computing	*"It is my pleasure to learn this interesting tool…" TT1*
	Confident	Understand the concept	*"I can now understand the concept of computing and comfortable handling the tool..." -TT3*
	Enjoyment	Enjoy learning things in flash	*"I think learning programming language is very useful and can give positive output to the students too besides teacher can enjoy create their own teaching module…" -TT6*
		Love using tools	*"I just love it!" -TT6*
		Believe the importance of learning flash	*"I can apply this tool to develop my own teaching tool since finance subject…" -TT6*
		Give full concentration	*"I can concentrate now with the given kind of online manual" TT8*
		The challenge of learning Flash is exciting	*"It is challenging but too exciting using it! -TT1*

Table 6-2: *Enthusiasm and Attitude Codes from Study 3 Responses (Continued).*

ORGANIZA-TION CODE	*OPEN CODE*	*PROPERTIES*	*QUOTES*
Enthusiasm and Attentive	**Appreciative**	It would help my profession if there are good facilities at school	*"I wish that my school will provide good facilities especially computer lab with very supportive software or latest technology" TT6*
	Valuable	Lesson delivery is enhanced	*"We learn to program not only for our teaching but for our future life learning process" TT6*
		Lesson learned improved (for difficult subject)	*"We learn to program to make the boring subject more fascinating and engaging." -TT6*
		Provision of a valuable learning experience	*"This process helps students to add experience of integration, which can lead students to arranging and designing more suitable digital tools to assist..." -TT4*
		The relationship between theory and practice is strengthened (e.g. through animation to learn numeracy or literacy)	*"…teach mathematics for example. The graphics used will make it easier…" -TT6*
	Challenging	Learning and teaching programming is challenging	*"…now I realize that to learn how to program is very challenging however…"- TT3*
	Persistence	Able to work with challenging tasks	*"..before this I thought it is very difficult and very hard to learn but now I am quite confident …" -TT1*
		Ability to handle ambiguity	*"I can produce some digital products for classroom teaching" -TT3*
		Ability to deal with open-ended problems	*"I understand clearly that they are concerned about writing computer programs. However, there is not that too complicated programming scripting as for now and normally requires only one or two lines of code to be written…" -TT1*

Table 6-4: *Computational Thinking Domain Thematic Codes and Example Quotations.*

ORGANIZA-TION CODE	*OPEN CODE*	*PROPERTIES*	*QUOTES*
Computational Thinking Domains	**Algorithmic Thinking**	Effort to improve	*"Therefore, an effort must be made to promote new methods with Flash or any other software for teaching or learning." -TT3*
		This process (develop from beginning to the end) forces the pre-service teachers to think about what they are doing in Flash- teaching.	*"…not only have to focus on knowing the techniques, but we also need to operate OR practically handle the tool by ourselves."-TT6*
		Enhance the diversity and configurability of the tasks	*"..I wish I could produce more modules for my own teaching soon," -TT5*
	Decomposition	Breaking problems into sub-tasks	*"… very interesting that I can have so many sub-scenes after I created a main page…" -TT5*
	Abstraction	Organizing & manipulating simple data	*"Wow, can't believe I can create this like I did using Flash…" -TT6*
		To make systems easier to think	*"…this is where the importance of naming the layer(s) accordingly so that…" -TT2*
		Choosing a way to represent artefacts to allow them to be manipulated in useful ways.	*"I feel confident now to make my teaching varies besides using only power point slides or board to lecture" ET1*
	Generalisation	Quickly solving new problems based on previous problems that have been solved.	*"this Anime software is almost similar to flash, I don't feel any difficulties in understanding the tool so much" -TT1* *"However, the current multimedia software called ANIME has become very easy to use. There is no need to write a computer program." -TT4*
		Reuse similar codes	*"…this technique helps me a lot especially to understand more the concept of copy and paste." -TT3*
		Transferring ideas and solutions from one problem area to another	*"Through the process of practice and manipulation we learn how to apply all the skills and become more familiar with using the software"-TT2.*

Table 6.3: *Computational Thinking Domain Thematic Codes and Example Quotations (Continued).*

ORGANIZA-TION CODE	***OPEN CODE***	***PROPERTIES***	***QUOTES***
Computational Thinking Domains	**Analysis**	Debug problems	*"I can spot where the problem occur when the error appeared" -TT3*
		Solving skills	*"I learn techniques on how to make my learning module more effective way…."- ET1* *"I know why I can't navigate to the other page… wrong format "- TT6*

Table 6-5: *Professional Practice Domain Thematic Codes and Example Quotations.*

ORGANIZA-TION CODE	***OPEN CODE***	***PROPERTIES***	***QUOTES***
Professional Enhancement	**Creative Designer**	Innovation in teaching	*"Innovations in teaching graphic in technical or non-technical education through interactive medium will attract attention…" -TT5*
		Animation in learning	*"I am sure students will be more impressive with the subject…" -TT2*
		Interactivity is the best way of teaching	*"Students enjoy learning by doing." -TT4*
		Stimulate students' creative thinking	*"Student explore whatever they wish to…" TT3*
		Devise relevant assessment - sketching story board	*"This storyboard helps me to design my work more effective rather than..."- TT2*
	Professional Enhancement	Teachers will be more grateful and happy	*"…in the initial stage we have to spend a lot of energy and time on the practical aspects. Once we have mastered this stage, we will have the freedom to do other kinds of work." -TT6*
		Conduct class using computer lab	*"…they will feel more confident and comfortable using the facilities in the lab." -ET2*
		Better teacher	*"This tool really helps me to make my teaching more fascinating." -ET3*
	OPEN CODES	***PROPERTIES***	***QUOTES***
	Instructional Productivity	Flash could enhance helpful instruction	*"I believe that it will help my professional development in the future" -TT3*
		Flash can help familiarize different teaching styles	*"If teachers are not given an opportunity to develop and enhance their abilities using the latest technologies such as interactive multimedia or web- based resources, they may abandon their mission to become an EDUCATOR or fail to achieve their full potential as good teachers.'' – TT2*
		Flash helps to integrate new teaching methods	*"..no more board and pen in teaching…"- TT7*
		Pre-service teacher should include instructional applications using Flash	*"I think the lesson will be more interesting and interactive if I can apply this kind of tool in my module" ET1*

6.6 RESULTS – FINAL OUTCOMES

The research questions (RQ4 and RQ5) provided the objective of seeing if use of the tool resulted in the participants producing working code for their artefact. This objective was to see if the tool would allow the users to produce a complex piece of software that would require computational thinking to be able to successfully complete. The exercise would require them to use abstraction, algorithm development, decomposition, generalisation and analysis.

It was also implied for Study 3 that the respondents would improve their attitudes and these were coded using the attitude thematic codes described in Table 6.1 above. Such attitudes are important to maintain engagement and motivation to learn programming (Martins, Mendes, and Figueiredo 2010; T. Jenkins and Davy 2002; Kofoed and Reng 2010). The most encouraging part was that the respondents were not only become a user of technology but they were the developer of it (Lin 2012; Singh 2016)

The final objective for testing the effectiveness of the tool was to see whether the respondents would be able to take their knowledge from learning how to use the tool and apply it to developing an animation artefact using a different animation software (QuickStart Computing 2014; Dorling M. 2014). The respondents were asked to produce any tasks they like by using Anime studio which was new to them (details will be explained in 6.5.3). Anime studio had been chosen in this study because the interface looked similar to that of the Flash software and was expected to be something they should be able to use if they could generalise the skills they had learned in the previous exercise.

6.6.1 Completing a complex artefact.

All eleven participants in this phase completed the tasks. The tool led them through the learning stages and provided the correction needed to aid their design. Though some worked slowly all completed the process. As they began working through the exercise they ran the training software in parallel. When the stages of their programming were complete they were able to compare the output they had achieved (functionally) with the desired output. When this was not matching they were able to discover from the tool where their thinking had been incorrect and run the appropriate sub topic. After this they were almost

always able to produce the correct output directly but the process could be repeated until success was achieved. All were seen to increase in their skill as the project and lessons progressed. The final artefact created was far more complex than those completed in Study 2. The coding no longer contained major errors in thinking and planning. The appropriate codes had been generalised so that code was only reused when it was clearly correct to do so. For example, the more complex structures (like switch cases) had been used appropriately. All components had been broken down in a suitable way. These non-computer programming-trained participants were approaching the task using Computational Thinking.

6.6.2 Changes in Attitudes.

The results of the coding of the questionnaire responses made by the participants are collected in Table 6.4 for the attitudes before using the tool and Table 6.5 for after using the tool.

Table 6-6: *Code Frequencies* ***Before*** *Completion of the Tool Based Training (Study 3).*

Codes	Frequency *(Pre-service Teachers)* ***(TT)***	**Normalised Frequency** *(Pre-service Teachers)* ***(TT)***	Frequency *(In-service Teachers)* ***(ET)***	**Normalised Frequency** *(In-service Teachers)* ***(ET)***	Total Frequencies	**Normalised**
		Summary of Positive Codes				
Interest	59	7.4	24	8.0	**83**	**7.5**
Engagement	45	5.6	32	10.7	**77**	**7.0**
Motivation	54	6.8	27	9.0	**81**	**7.4**
Confident	52	6.5	27	9.0	**79**	**7.2**
Enjoyment	52	6.5	29	9.7	**81**	**7.4**
Affordance	47	5.9	28	9.3	**75**	**6.8**
		Summary of Negative Codes				
Boredom	0	0.0	0	0.0	**0**	**0.0**
Lack of Motivation	0	0.0	0	0.0	**0**	**0.0**
Confusion	8	1.0	4	1.3	**12**	**1.1**
Fear	0	0.0	0	0.0	**0**	**0.0**
Frustration	0	0.0	0	0.0	**0**	**0.0**
Difficult	5	0.6	2	0.7	**7**	**0.6**

Table 6-7: *Code Frequencies* ***After*** *Completion of the Tool Based Training (Study 3).*

Codes	Frequency *(Pre-service Teachers)* ***(TT)***	**Normalised Frequency** *(Pre-service Teachers)* ***(TT)***	Frequency *(In-service Teachers)* ***(ET)***	**Normalised Frequency** *(In-service Teachers)* ***(ET)***	Total Frequencies	**Normalised**
		Summary of Positive Codes				
Interest	65	8.1	32	10.7	**97**	**8.8**
Engagement	50	6.3	45	15.0	**95**	**8.6**
Motivation	58	7.3	38	12.7	**96**	**8.7**
Confident	69	8.6	36	12.0	**105**	**9.5**
Enjoyment	54	6.8	38	12.7	**92**	**8.4**
Affordance	38	4.8	25	8.3	**63**	**5.7**
		Summary of Negative Codes				
Boredom	0	0.0	0	0.0	**0**	**0.0**
Lack of Motivation	0	0.0	0	0.0	**0**	**0.0**
Confusion	0	0.0	2	0.7	**2**	**0.2**
Fear	0	0.0	0	0.0	**0**	**0.0**
Frustration	0	0.0	0	0.0	**0**	**0.0**
Difficult	0	0.0	2	0.7	**2**	**0.2**

All participants in Study 3 had already had training in using Action Scripting so their responses before using the tool were already better than they were when they had no training. Use of the tool almost eliminated all negative comments (a total of 4 negative comments after Study 3 or 0.4 comments per participant). Positive comments had almost all risen but the most significant change is possibly the more than 30% increase in confidence comments.

Thus from the subjective individual responses of the participants the tool had achieved the motivational impact looked for in the research objective. A more concrete test was set in the fourth objective. The result of this experiment was that all participants were successful in designing and building their own Action Script based artefact after the exercise.

6.6.3 Participants' ability to use another animation tool - Generalisation in CT.

Research questions RQ 4 and 5 imply that the participants could be tested using another different animation tool without receiving prior training in using this tool. The tool chosen was Anime Studio. The interface to this tool had a lot in common with the Action Script interface and it was expected that participants who were using CT to design and develop their artefacts should be able to generalise the ideas learned to using this new software. They would be able to break down a new task into a set of steps and implement pseudo code to solve this problem. They were given an example animation to reproduce and were observed using the new tool. This exercise was completed by all respondents and the output produced was successful. No help was given to them in using the new tool. Figure 6.16 shows the completion of a task using Anime Studio by one of the participants. An example of a finished artefact produced by another of the participants is given below in Figure 6.17.

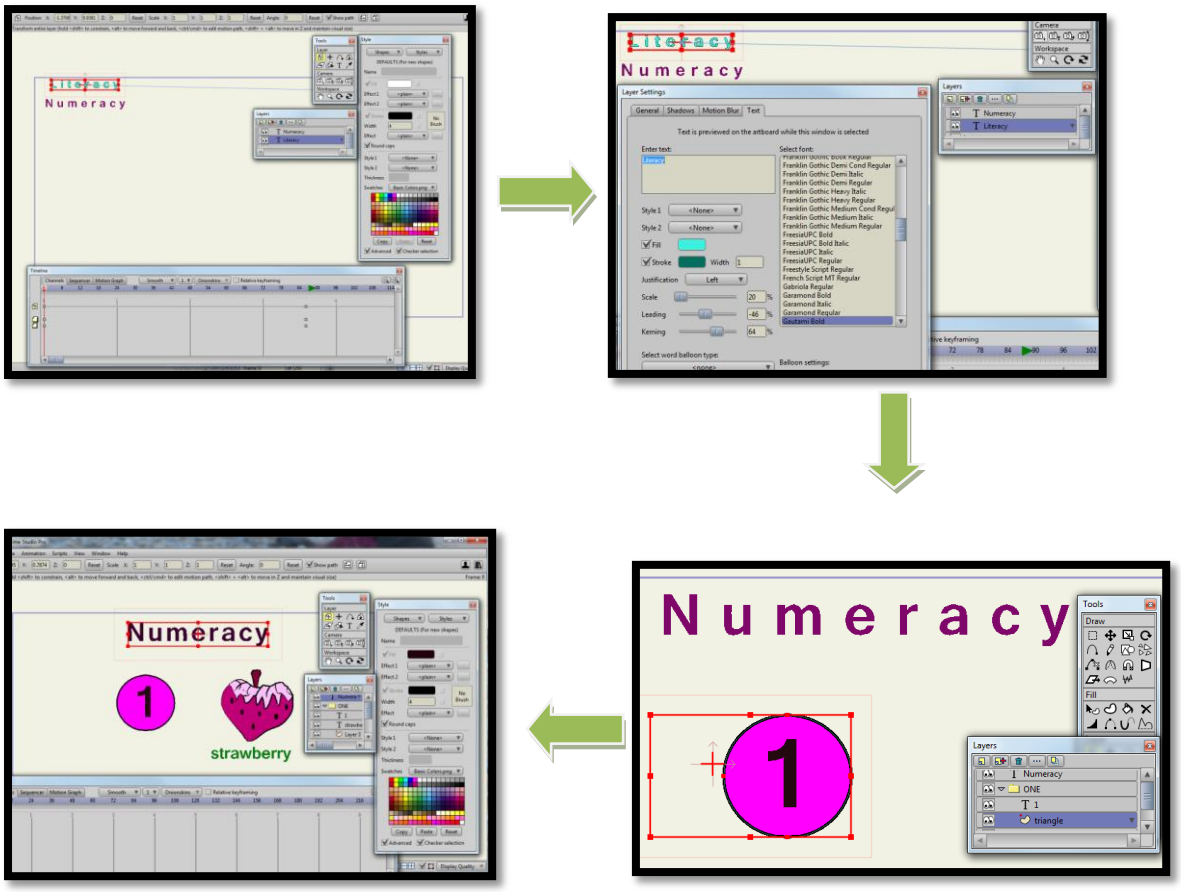


Figure 6.15: *The Same Task performed in Anime Studio (TT2)*

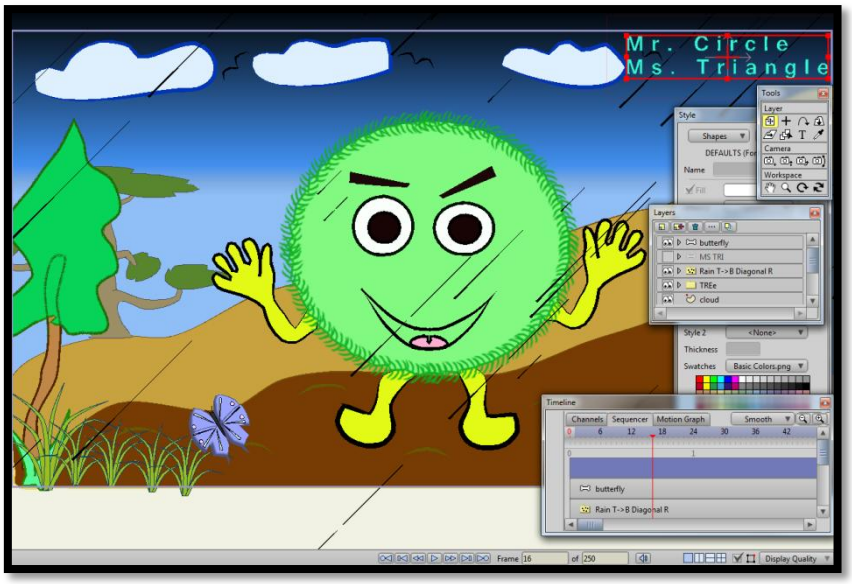


Figure 6.16: *Short Movie Clip using Anime Software (TT3).*

Thus RQ4 and 5 were fully tested and produced a positive result. Participants with no prior experience apart from the skills obtained through taking part in the Action Script lessons were able to generalise their learning to a new interface. They were observed using the basic steps of breaking the requirements down into steps (decomposition) providing a set of steps to produce these sub-components (algorithm building) and generalising their coding where appropriate.

6.7 CONCLUSION

This chapter has covered Study 3 of the experiment. All of the participants from Study 2 also took part in Study 3. Obviously their attitudes and expectations on starting Study 3 were higher than they had been when entering Study 2 but it was useful to investigate whether those attitudes could be further improved by use of the standalone tool. The design of the tool was based around the findings from Study 2. As all the students had used wrong thinking when they began Study 2 it was pleasing to note that 11 of them had already begun to improve. However, two of the respondents still started the process with the same wrong thinking as observed in Study 2. Analysis of their performance in Study 2 allowed the Study 3 software tool to be designed specifically to address these problems where they arose. The participants were able to complete the building of an artefact with a lot of individual choice

and in parallel refer to the lesson material. This provided a much more convenient way of learning. Before the exercise some of the participants had expressed specific fears over working without support apart from by the machine. All, however, were soon observed to find use of the tool easy. There was some difference in rate of progress (some had learned more in Study 2 than others so this is not at all surprising) but all participants produced valid artefacts using a good software engineering approach indicating they had begun to adopt the CT skills and apply them to programming. The result of the post exercise questioning demonstrated once again improvement in attitudes with negative attitudes to programming in Flash Action Scripts having almost completely disappeared and positive attitudes (especially those relating to self-confidence) having generally markedly increased.

RQ 4 asked whether teachers who have adopted CT can demonstrate this by applying it to undertaking 'programming' tasks. Respondents did indeed do this, thus demonstrating that they could generalise what they had learnt to using other software (Csizmadia et al. 2015). RQ 5 also asks about whether a suitable tool used in learning can motivate participants to proceed with programming. This was confirmed, as shown in Figures 6.16 and 6.17.

The final chapter moves on to discuss these findings further in light of the research literature and discusses the implications for this research and future directions.

CHAPTER 7

DISCUSSION AND CONCLUSION

7.1 INTRODUCTION

Computational Thinking skills will be integrated in the curricula for both primary and secondary schools in Malaysia starting from 2017 (http://www.themalaymailonline.com). A clear understanding on the issue and the importance of having clear perspective in Malaysia curricula since 2011 allowed this research to be conducted. However, since January 2017 the CT skill subject is still not fully implemented due to certain circumstances raised from teachers. An ambitious initiative to integrate coding into the national school curricula has worried teachers and this has halted the implementation in school curricular recently (http://www.themalaymailonline.com). Therefore, the research in the study which was conducted to look at how motivation for teachers themselves to learn CT using Flash Action Script came at a very opportune time.

The work for this thesis was inspired by the increasing requirement for the teaching of programming in schools down to the earliest years. In the UK this process has been developing over a number of years and led to the introduction of the Computers and Schools (CAS) initiative computing at school (Crick and Sentance 2011). In recent months this priority has become critical in Malaysia with the introduction of the same requirements there (A. Asohan 2016; Azlee 2016). The Malaysian government recently announced (subsequent to the work in this thesis) that in 2017 coding would be taught in all primary schools from year three (Abas 2016; Malaysia Education Blueprint 2013). As was discussed in the introduction and literature review, the fact that few primary school teachers are experienced programmers and do not have the technical background naturally to be expected to teach programming means that a problem arises (Suliman, Hawari, and Othman 2011; Umar and Hassan 2015; A. Asohan 2016). Despite the initiative of the MOE in developing Continuing Professional Development (CPD) centres specific for ICT, as seen in Chapter 2, there is still a great need for such training if Malaysia is to move forward with its ambitious programme of computer literacy fit for the 21st Century. In this thesis the concept

of Computational Thinking (CT) has been introduced and its value in solving general problems has been looked at. Its value in learning to program and then in teaching programming has been demonstrated through the literature research.

From the findings of the literature review it is expected that teachers learning computing will help produce creative designers or innovators among teachers as well as their students (Hassan et. al. 2010). However, as shown in Chapter 4, Table 4.1, respondents from among the pre-service and in-service teacher community are afraid of programming and don't think they could develop the ability to program. Therefore, the issue is one of trying to inspire and train teachers to become competent to teach programming. This resulted in a teaching artefact being developed to allow its users to adopt computational thinking (CT) skills and use these as an approach to learning to program.

The preceding chapters have presented the results of three phases of this research – Study 1, Study 2 and Study 3 to establish whether this is a real problem and, having seen that, to look for a way of motivating the teachers to overcome the challenges. The purpose of this chapter is to bring together the findings of the research and compare the outcomes of the three stages in the light of the research questions. Connections between the results of the three sections have been presented earlier but their significance can be seen clearly against the research question targets.

This research has examined the requirements for becoming competent to teach programming. Firstly, it is believed that an effective educator of computer programming needs to adopt CT themselves, which, as discussed in Chapter 1, is not just programming but the ability to solve problems in more effective ways. As such, the CT process can be broken down into, and assessed via, several components namely (Dorling M. 2014): abstraction (1), algorithmic thinking (2), decomposition (3), generalization (4) and analysis (5). The details on each domain have been explained on section 1.1.

7.2 RESEARCH QUESTIONS

The issue of how to get teachers engaged in learning computational thinking so they can then teach programming led to the development of the following research questions which have been investigated throughout the thesis:

RQ 1 Are primary school teachers ready to teach programming?

Objectives:

- To identify the challenges in learning to teach programming.
- To determine whether primary school teachers think computationally.

The literature review had identified some of the misgivings of primary school teachers as they face the prospect of having to teach programming and this was tested in Study 1 which was designed principally to start investigating this question.

Study 1 was conducted with pre-service teachers as they went through the process of learning Action Script Programming at the Sultan Idris Educational University (UPSI) in Malaysia. Pre service teachers' views were gathered on their ability to program and evidence showed that teachers viewed programming as an "impossible" subject to learn by users without a computing background. Teachers train to teach in different subject domains and therefore have their capabilities in that domain in particular. Opinions showed their reluctance to learn due to it being irrelevant to, and over-technical for, their teaching domain and they also saw no necessity to develop teaching materials as schools provide what they have to use. The majority of pre-service teachers also felt that the right teaching methods are vital but that they didn't feel ready to teach programming early on in their studies. Conventional teaching approaches such as chalk and board, or PowerPoint slides were not engaging enough (Carter and Boyle 2002) to learn to program due to a number of factors as discussed in Chapter 2, Section 2.1.

There is evidence that learning CT skills can help teachers to understand that they could learn programming as they would understand the fundamental concepts behind it (Y. Lee and Lee 2014). Study 1 indicated that both pre-service and in-service teachers did not think computationally and did not understand the concepts or fundamentals of programming (please refer to Table 4.2). This resulted in behaviors when trying to code in Action Scripts such as copying and pasting sections from existing scripts whether or not those sections were suitable. There was no motivation to learn programming as many, especially in Malaysia, felt that task should be left to ICT teachers (Singh 2016). The first objective above therefore identified some of the issues. The challenges to learning are apparent from the responses given in Study 1. Before the lessons there was a significant element of fear which

completely disappeared after the module. Boredom and lack of motivation comments were also almost all overcome after the lesson. On the other hand, comments relating to positive motivation almost doubled in frequency.

One issue that may have changed teachers' thinking since the time Study 1 was undertaken is the introduction of the Malaysian government policy that programming should be taught in all primary schools in Malaysia (A. Asohan 2016). That issue is the "Why should I learn to program?" attitude demonstrated in responses from a number of participants both in Study 1 and during Study 2.

Observations of the process in Study 1 and Study 2 showed clearly that the respondents were not thinking in a CT way. Their attempts to produce artefacts showed a lack of planning, failure to break down tasks into parts, failure to look for the correct generalisations and failure to build proper algorithms. This was also seen in Study 2 (please refer Table 5-1) of the experiment. Clearly the main challenges were negative attitudes to the idea of learning to program (both regarding confidence but also the need to learn and the appropriateness of the tools). Also the primary school teachers were clearly not aware of CT and did not understand that this would be needed to get the programs written. If they were not thinking in a computational way how would they expect to teach children to think in that way?

Study 1 therefore highlighted a number of problems related to motivating teachers to learn to program and to increasing their confidence and understanding of the relevance of programming. This lead to the question of whether those from a non-computational background could learn and be inspired to think computationally (Lin 2012; Suliman, Hawari, and Othman 2011).

RQ 2 Can those without a CT background learn to think computationally?

Objective:

- To understand how teachers might be motivated to learn to think computationally.

Clearly if primary school pre-service teachers do not think computationally but have to learn to teach programming there would be a great incentive for them to learn to think computationally. But how can this be motivated? The literature had shown a number of ways in which children were being taught the elements of programming (Brennan and Resnick 2012) but from Study 1 and Study 2 most teachers felt that the methods used on children did not inspire them, as mentioned by R. Ibrahim and Jaafar (2010) that the programming subject is found to be a difficult and boring one and (X. Ling and Wei 2014) stated that traditional teaching methods often concentrate on grammar scripts and its simple application. Although they mention algorithms this is normally limited to the algorithm itself and not the level of teaching thinking skill. Therefore, it causes the students to think of programming as boring and difficult to learn.

The basic core of the methods presented in the current research was to look for things that would attract teachers and students and thus make them intrinsically motivated. One of the ways used to engage children with programming was to get them to program games (Prensky 2003; Roslina Ibrahim and Yusoff 2010; Corneliussen and Prøitz 2015). Others appealed to the attraction of making drawings or playing with robots (Strawhacker and Bers 2015; Mariappan, Sing, and Nadarajan 2015). None of these were seen to be motivators for the majority of the teacher participants.

Study 1 identified a lack of suitable teaching instruments used to teach programming to adults and that if these were provided then motivation to learn how to program could be increased. However, some of the respondents were concerned that this would be too difficult and others felt their teaching schedule would not allow the time for this sort of development.

Those with a non-technical or non-computing background in particular felt that conventional student-centered teaching approaches were not appropriate. However one-to-one teaching was used in the study and found to be successful in identifying and correcting non-CT and enabling teachers to learn the concepts of programming. Existing teaching tools, however, such as Scratch and Storytelling Alice were not seen by the teachers as being motivating for them to learn but would rather dissuade them (E. M. Saari 2014). In the traditional classroom at UPSI teaching of Flash Action Scripts was used and were more

successful in motivating learning of programming but teachers' incorrect thinking was still identified and not corrected in that teaching. That is participants were not being challenged to think computationally.

The results of Study 1 were clearly supportive of the idea that learning Action Script programming would improve their attitudes to the concept of them learning to program (Lin 2012) and so Study 2 was designed to find out if a specific direction that focused on correcting thinking failures (Yinnan and Chaosheng 2012; Yinnan and Chaosheng 2012) would support this learning. The outcome was to be the design parameters for a standalone tool that could support this.

The experiment of Study 2 was covered in Chapter 5. Clearly the results of this phase showed a further improvement in attitudes and self-belief by the participants. The pre- and post- attitude questionnaire responses are discussed in Tables 5.4 and 5.5 respectively for clarification.

Comparisons of the normalized figures shown in Table 5.5 illustrate that there was very little difference between the pre-service teachers and their already practicing colleagues. Confidence has increased in both groups but a significant amount of confusion remained for both groups and a level of difficulty was still perceived by the participants.

Observation by the researcher in Study 2 showed once again a lack of Computational Thinking in the participants at the beginning of their experiment phase. When a failed result was achieved at any stage the participant was taken through a corrective step which pointed out the root of their problems. By the end of the sequence of sub-lessons improvements in thinking were found. This encouraged the belief that the approach taken would lead to the pre-service and in-service teachers being able to think computationally. This then laid the basis for Study 3 of the experiment where a tool would lead the participants through a set of steps to produce a more complex artifact which would have been designed to focus on developing their computational thinking.

However, in order for the tool to be designed successfully it required identification of when and where CT was missing in the teachers' thinking in order to try and promote them to

analyze and adapt their thinking. This lead on to the question of whether a lack of CT could be identified and how to change incorrect thinking.

RQ 3 Can we detect where CT is not happening and lead them to re-analyse their thinking strategies and consequently learn to think computationally?

Objective:

- To work out how to detect use of the five CT domains: abstraction, algorithmic thinking, decomposition, generalization and analysis (details about the CT domain can be referred to page 7).

Clearly the work in Study 2 had already begun to answer this question. By direct observation of the actions taken by the participants their lack of CT was seen. By repeated correction of problems so detected the researcher guiding them was able to help them to see what the correct thinking approach would be (please refer to 5.3.2.2 and 5.3.2.3). The participants would then correct their work and at least some of them were thinking in a more CT way by the end of the process.

It was found that skills in all five CT domains could be identified. Study 2 identified the presence of abstraction through observing the users' creation and organisation of the layer functions and names of scenes (see Table 4.3). Algorithmic thinking was identified in the storyline of the lessons being produced and the logical flow of content. In this initial study wrong algorithmic thinking was also detected when users were observed to copy and paste navigation buttons. To correctly perform this part of the task would have required algorithmic thinking which was not evident here.

Decomposition involves dividing a large task into smaller sub-tasks (Sentance and Csizmadia 2016; Csizmadia et al. 2015). In the experiment, this involved separating the main scene into small sub-scenes which would allow the contents to be maintained and updated more easily. The majority of participants did not apply this technique showing they lacked this skill.

The final stage of the study (Study 3, see section 6.5) involved the development of a teaching artefact to identify their thinking skills in terms of generalisation and analysis and

to see if these skills could be transferred to the use of other software (Sections 5.2 and 6.5.3).

Synthesizing these observations into a standalone tool allowed the researcher to think through this process and build the corrective steps into the process of the teaching tool. In Study 3 the participants took part without the help of the researcher. The observations (made visually and through looking through the Camtasia output afterwards) allowed the researcher to see the development of the thinking. The artifacts being designed by the students in Study 3 were more complex than those in Study 2. The participants had all taken part in the previous phase so their attitudes and abilities were already known. They all made progress through the work and produced working solutions. The complexity of the task and the simplicity of the solutions added to the belief that they had begun to think computationally.

The study lead to the conclusion that identifying CT, or lack of, is possible. The next stage of the research involved determining whether the presence of such skills could be transferred to other programming tasks.

RQ 4 Can pre-service teachers who have adopted CT demonstrate this by applying it to undertaking "programming" tasks?

Objectives:

- Provide tasks successful completion of which would demonstrate the use of CT.
- Provide tasks that would demonstrate they can generalise what they learned to using other software.

Study 3 had clearly shown that the participants could learn to program. Each made their own artefact and through the stages of the artefact they created each had to demonstrate use of the same types of programming constructs, none could be answered without going through many of the parts of computational thinking. The tasks were all broken down into sub tasks, showing use of decomposition. The algorithms for each part were worked out. Where appropriate generalisation took place – for example through the use of "switch case" statements rather than repeating the virtually identical code specifically for each task.

By adopting the CT domains (abstraction, algorithmic thinking, decomposition, generalization and analysis) in the lesson tool, they managed to explore additional scripts (abstraction and decomposition) and enhance the graphic design (algorithmic thinking) based on their own creativity, debugging errors and eventually created more screens (decomposition and analysis) with additional interactive buttons to produce their own complete lesson module.

The most interesting and fascinating part was that the application of the tool had been not only able to teach the students effectively to use the software development system they had learned but also to use other software such as Anime Studio (see Figures 6.17). Participants had adopted the skill of generalisation when the interface on both pieces of software looked similar, and respondents had had no difficulties in adapting features in both. Thus, transference of CT skills was demonstrated but it was also important to identify whether the use of the tool motivated teachers to learn further and indeed to teach programming.

RQ 5 Does the use of a suitable tool in learning create users who will be motivated to go on programming?

Objectives:

- To identify changes in attitude after use of the tool compared to before.
- To identify what barriers still remained.

To get the answer to this question it is necessary to look at the attitude responses of the participants after Study 3. However, some of the actual statements of the participants answer this directly:

'I enjoy coding because it's fun to create programs and there are no limits to your creativity.' (Respondent TT7)

'You can make anything you like and you can always improve on it' (Respondent ET3)

The attitude survey from the end of Study 3 is again repeated in Table 6.5 to aid comparison. Obviously the two groups – the large group who took part in Study 1 and the

smaller group of in-service and pre-service teachers who took part in Study 2 and 3 were very different. The first group could have been expected to have started with better than average attitudes as they were on an elective module in the subject area. They did, however, start out with a lot of negative attitudes as seen in Table 4.5. The training they received in the ordinary classroom reduced the negative attitudes and improved the positive responses as shown in Table 5.5. The group taking part in Study 2 and 3 were volunteers but during recruitment several had expressed negative attitudes to the possible effectiveness and benefit of taking part. It is useful to separate the responses into the two sub groups, the in-service teachers and pre-service teachers and this has been done in Tables 5.4, 5.5 and 6.5. In order to compare the values between the two groups the numbers were also shown normalized into number of codes per respondant.

Comparing between Table 4.5, the beginning of Study 2 and Table 6.5 at the end of Study 3 it can be seen that negative attitudes have more or less disappeared with only 0.4 negative comments per participant compared to a total of 14 negative comments per participant at the start of Study 2. Positive comments had increased from approximately 33 per participant to nearly 50 per participant at the end. The biggest individual improvement was an increase of over 5 positive comments relating to confidence in programming compared to before – corresponding to a 139% increase.

Comparing the improvement between before and after the conventional teaching approach used in Study 1 where there was a total increase in positive comments by only 2.3 comments per participant there was an increase of 6.5 total positive comments between the beginning and end of this teaching. There were only 11 participants in Study 3 compared to the 50 in Study 1 so there may be many explanations for the rate of comments. One important factor may be that the experience of using the tool is far more tailored to individual needs. In addition, very few of the 50 participants in Study 1 showed any increase in CT skills by the end of the process.

The tool did increase motivation to program, as gleaned from participants' responses, and they were no longer fearful of programming or technology (see Tables 5.5 and 6.5). The tool provided an engaging and interesting environment which was shown to help the participants with their learning of computational concepts and this is a benefit that has been

shown by other researchers (Lye & Koh 2014). The approach has to be appropriate to be successful (Lin 2012). What this tool provided, however, was the understanding of how to think before applying those computational concepts. By adopting CT skills, respondents demonstrated an ability to program not only using Flash Action Scripts but also in their applied Anime Studio software (demonstrating generalization), as shown in Figure 6.16. It was clear that pre-service teachers' thoughts had changed considerably from Study 1 and, as can be seen in their comments, they were clearly enthusiastic to use the tool and more confident.

It addition to this, the pre-service teachers could see the positive impact the tool had had on their ability to learn to think computationally and had become advocates of the use of technology in teaching. The exception to this was one in-service teacher who did not show any interest in the experiment. His lack of motivation to use the tool was evident in his comments that the school and school technicians should take on the role of providing IT, not the teachers. However, he revealed that his future career plans did not involve teaching or being in education once he had completed his postgraduate studies and he was therefore not motivated to engage with the tool for this reason.

The studies thus far have demonstrated the positive impact of the tool in its ability to motivate teachers to learn to program. They have also demonstrated the possibility of teaching CT skills which can then be successfully applied to different programming situations. This leads us to consider the possible implications of the studies for existing use of multimedia in teaching at schools.

RQ 6 What are the implications of this study for the teaching of CT and programming in teacher training courses?

It transpires that the development of the tool is particularly timely given the Malaysian government's announcement that programming will be taught in all primary schools from 2017 (Suliman, Hawari, and Othman 2011; A. Asohan 2016). A statement from the government also acknowledges the value of programming in promoting higher order thinking skills (S. Y. Tan and Halili 2015; Dewitt, Alias, and Siraj 2016). The research

presented in this thesis has significant implications for the way that programming is taught to teachers. The value in this work is in demonstrating that it is vital to determine the presence or absence of CT skills as a precursor to teaching programming and nurturing those skills through motivation and appropriate teaching methods is also key. If programming is taught in the correct way many more teachers can learn to program than would otherwise have been possible (Suliman, Hawari, and Othman 2011; A. Asohan 2016). This has significant consequences for both the Malaysian government and society as a whole. This study leads the way to future expansion of the tool in terms of content and how that content can be applied in education and further afield.

7.3 LIMITATIONS

There are a number of limitations to this research which could be addressed in the future. The standalone tool was developed as software in its own right and so was not available online. This meant that users could only access the tool when given access to it in the classroom or, in a couple of cases, when they asked to have a copy of the tool. The fact that the tool was not available online meant that there was limited scope for the number of participants who could try out the tool and give feedback.

The content within the tool was limited since it was developed to test the ideas in this research. It would be interesting to conduct further experiments with a more fully-fledged tool allowing the creation of more content whilst still guiding the user through the correct use of CT.

The tool was shown to motivate users to learn to program but the length of study was limited and it would be useful to test the tool over a much longer time period to see if it maintained users' interest and engagement. This would require more users, content and perhaps require more complex development and reasoning but this, again, would be useful to observe and analyse.

It would be useful to gain feedback from those whose job is to train teachers to teach. The tool aimed to take on some of that role in the domain of learning to program, but it would be interesting to get instructors' feedback and also to have a better understanding of whether the tool could augment existing teaching practices to produce a more optimal

outcome. The tool was only studied in its own right and not in conjunction with other teaching of the subject of programming.

7.4 FUTURE WORK

A number of future tasks have been identified. A longer term study on the use of the tool and its effect should be conducted to gauge more fully the extent to which it might be used and how good it is at teaching CT. This could be done through assessments of ability pre- and post-study and formal measurements of motivation as well as studying users' attitudes and beliefs with respect to technology and programming.

One of the more interesting directions would be to experiment with the idea of the ability to gauge and adapt other cognitive functions in users through the use of a tool. It has been shown that CT skills can be identified and, where needed, taught to users. This leads us to ask whether it is possible to identify other incorrect or unhelpful thinking in users and then correct their cognitive processes. Such a tool could have widespread uses in many domains, although would also need to be used carefully.

More elements of CT such as generalization can and should be added to the tool and this should make the resulting system even more attractive to both young learners and more experienced people from various backgrounds. Care would need to be taken in order not to make the tool overcomplicated. This could be achieved by levels in the tool through which users work.

The tool is currently a standalone artifact and one of the obvious expansions is to make it available to learners online. Making the tool more accessible would allow greater reach within the community and therefore greater impact. It could also allow users to comment on each other's development of these skills enabling scaffolding and community-based support.

Commercialization could also enhance reach and enable the tool to be more widely promoted not just in teacher training at university level but also as an additional teaching

instrument or artifact to be applied outside of academia. Consequences for industry could be significant and, as discussed previously, the development of CT skills have a wider impact than just using technology and programming. These skills are valuable throughout life.

7.5 CONTRIBUTIONS

Initially, it was anticipated that the contributions of this research would stretch to improving the experience of pre-service teachers in learning to teach programming at UPSI and in Malaysia in general. Contributions to this have indeed been made in several ways as follows.

It has been demonstrated that CT skills can be gauged and also taught where incorrect thinking is present and that a computer-based tool can be used for this. As such this study adds to the body of knowledge on CT.

This research has a wider impact on the roles of tutors who teach programming in that, if a tool such as the one developed here is used then teachers should be required less to teach higher order thinking skills and have more time to teach other material. This would contribute to both teacher training and also be useful to teachers who teach programming to pupils in schools. If their teaching is more effective because they understand more fully what they are doing, then this will enable far better learning in their pupils and much more confidence for the teachers. This contribution applies not only to Malaysia but also to countries worldwide.

A further contribution is that a mechanism has been developed which enhances motivation and helps users overcome barriers to using technology and programming. This is in addition to helping teachers understand why they should learn to program and to help them accept this role more willingly. The removal of mental blocks to engaging in an activity can reduce the battle to learn and this research makes a valuable contribution in that area.

Finally, a tool has been developed which achieves all this but, due to its effect on fundamental thinking skills, can be used to impact more widely than just learning to program computers. CT skills are important in everyday life and as such, the tool is a valuable contribution to the ability to more fully develop these skills and potentially others.

Ten years from now, much of what we have taught our students will be outdated, but the thought processes will still be applicable. The thought processes that parallel a computer's processing enables students to be lifelong learners and for example adapt to and use computing tools regardless of how tools change. As far as UPSI is concerned, the tool provides an additional medium for teaching pre-service teachers, and others skills that can be applied in different disciplines. However, this timely research can also impact on the development of other initiatives which promote progression in science, engineering and other domains in Malaysia and enable the country to keep pace with the needs of the world and become a "competitive technology hub for the region" (Wawasan2020, 2008 and Malay Mail, 2016).

Ultimately the tool will be proposed to the government of Malaysia specifically for application to the Education sector. The recent announcement from the CEO of the Multimedia Digital Economy Corporation (MDEC), stated that coding will not only be taught as one of the subjects in Computer Science but also will be included in the pedagogy of teaching (Malay Mail, 2016). This research therefore contributes widely to education as a whole in Malaysia, as well as worldwide.

REFERENCES

A. Asohan. 2016. “Malaysia to Make Coding Part of School Syllabus … 25 + Years Later.” *Http://www.themalaymailonline.com/what-You-Think-/article/malaysia-to-Make-Coding-Part-of-School-Syllabus-25-Years-Later*.

Abas, Azura. 2016. “Computational Thinking Skills to Be Introduced in School Curriculum next Year.” *News Straits Times*.

Ahn, Youngseok, Jun Lee, Hyungseok Kim, and Jee In Kim. 2014. “A Slim and Wide Multi-Touch Tabletop Interface and Its Applications.” *2014 International Conference on Big Data and Smart Computing, BIGCOMP 2014*, 276–81. doi:10.1109/BIGCOMP.2014.6741452.

Ainsworth, S E. 2006. “DeFT: A Conceptual Framework for Learning with Multiple Representations.” *Learning and Instruction*, in press. http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VFW-4JXPS4C-1&_user=10&_coverDate=06/30/2006&_rdoc=1&_fmt=full&_orig=article&_cdi=6021&_sort=v&_docanchor=&view=c&_ct=784&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=ae888bef4bbec80f6b34133f43832115.

Angeli, Charoula, Joke Voogt, Andrew Fluck, Mary Webb, Margaret Cox, J. Malyn-Smith, and Jason Zagami. 2016. “A K-6 Computational Thinking Curriculum Framework : Implications for Teacher Knowledge Teacher Knowledge.” *Educational Technology & Society* 19 (3): 47–57.

Arnold, Judith M, Floyd Csir, Jennifer Sias, and Jingping Zhang. 2008. “Does Anyone Need Help Out There ? Lessons from Designing Online Help.” *Imprint* 5301 (906718496). doi:10.1300/J136v09n03.

Azlee, B Y Aizy. 2016. “Lack of Info on Coding in Schools Prompts Concern among Educators.” *Http://www.themalaymailonline.com/malaysia/article/lack-of-Info-on-Coding-in-School-Prompt-Concern-Amon-Educators*.

Balan, Poovarasi, and V Kalavally. 2012. “Enhancing Student Motivation in Process Control via Interactive Learning Tools.” *… for Education (T4E), 2012 IEEE Fourth …*, 176–79. doi:10.1109/T4E.2012.38.

Barr, By David, John Harrison, and Leslie Conery. 2011. “Computational Thinking : A Digital Age” 5191 (April): 20–23.

Barr, V, and Chris Stephenson. 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://dl.acm.org/citation.cfm?id=1929905.

Begosso, LC. 2012. “An Approach for Teaching Algorithms and Computer Programming Using Greenfoot and Python.” *Frontiers in …*. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6462307.

Bell, Judith. 2005. *Doing Your Research Project*. *British Journal of Educational Technology*.

Vol. 3rd. http://books.google.com/books?id=aroefdSRc8AC&pgis=1.

Bernama. 2008. “Malaysia As A Fully Developed Country.” *www.wawasan2020.com*. http://www.pmo.gov.my/?menu=page&page=1898.

———. 2017. “Coding - Implementation at Schools by 2017.” *The Sun Daily*. http://www.thesundaily.my/news/1924629.

Bocconi, Stefania, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt, Panagiotis Kampylis, and Yves Punie. 2016. *Developing Computational Thinking in Compulsory Education. Implications for Policy and Practice. Proceedings EdMedia 2016*. doi:10.2791/792158.

Boeije, Hennie. 2002. “A Purposeful Approach to the Constant Comparative Method in the Analysis of Qualitative Interviews.” *Quality and Quantity* 36: 391–409. doi:10.1023/A:1020909529486.

Bond, Camilla. 2010. “Journal of Computer Assisted Learning.” *Anzmac* 3 (1): 1–9.

Boulton, Helen. 2008. “Managing E-Learning: What Are the Real Implications for Schools?” *The Electronic Journal of E-Learning* 6 (1): 11–18.

———. 2017. “Exploring the Effectiveness of New Technologies : Improving Literacy and Engaging Learners at Risk of Social Exclusion in the UK.” *Teaching and Teacher Education* 63. Elsevier Ltd: 73–81. doi:10.1016/j.tate.2016.12.008.

Boulton, Helen, and Alison Hramiak. 2013. “Cascading the Use of Web 2.0 Technology in Secondary Schools in the United Kingdom: Identifying the Barriers beyond Pre-Service Training.” *Technology, Pedagogy and Education* 23 (2): 151–65. doi:10.1080/1475939X.2013.802994.

Bradshaw, Pete, and John Woollard. 2012. “Computing at School: An Emergent Community of Practice for a Re-Emergent Subject.” *ICICTE 2012 Proceedings*, 602–12.

Braun, V, and V Clarke. 2006. “Using Thematic Analysis in Psychology.” *Zhurnal Eksperimental’noi I Teoreticheskoi Fiziki* 3: 77–101. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0.

Brennan, Karen, and Mitchel Resnick. 2012. “New Frameworks for Studying and Assessing the Development of Computational Thinking.” *… of the 2012 Annual Meeting of the …*. http://pdf.thepdfportal.com/PDFFiles/26416.pdf.

———. 2013. “Stories from the Scratch Community: Connecting with Ideas, Interests, and People.” In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education - SIGCSE ’13*, 463. doi:10.1145/2445196.2445336.

Brown, NCC, and Michael Kölling. 2013. “Bringing Computer Science Back into Schools: Lessons from the UK.” *… on Computer Science …*, 269–74. http://dl.acm.org/citation.cfm?id=2445277.

Brusilovsky, Peter, and Eduardo Calabrese. 1997. “Mini-Languages: A Way to Learn Programming Principles.” *Education and …* 83: 65–83.

http://link.springer.com/article/10.1023/A:1018636507883.

Bundy, Alan. 2007. “Computational Thinking Is Pervasive New Kinds of Question ; New Kinds of Answer New Hypotheses ; New Theories New Thinking ; New Angles,” 1–3.

Busaki, Keitarou, Yasue Iijima, and Susumu Konno. 2014. “Multi- a Gent System Learning Support Software with Fighting Games” 5 (5): 13–22.

Carter, Janet, and Roger Boyle. 2002. “Teaching Delivery Issues: Lessons from Computer Science.” *Journal of Information Technology Education* 1 (2): 78–89. http://kar.kent.ac.uk/13824/.

Catlin, Dave, and John Woollard. 2014. “Educational Robots and Computational Thinking.” *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, 144–51.

Chang, C.-Y., Y.-T. Chien, Y.-H. Chang, and C.-Y. Lin. 2012. “MAGDAIRE: A Model to Foster Pre-Service Teachers’ Ability in Integrating ICT and Teaching in Taiwan.” *Australasian Journal of Educational Technology* 28 (6): 983–99. http://ascilite.org.au/ajet/ajet28/chang-cy.html.

Charmaz, K. 1990. “‘Discovering’ Chronic Illness: Using Grounded Theory.” *Social Science & Medicine (1982)* 30 (11): 1161–72. doi:10.1016/0277-9536(90)90256-R.

Charmaz, K. 2006. “The Power of Names.” *Journal of Contemporary Ethnography* 35 (4): 396–99. doi:10.1177/0891241606286983.

Charmaz, Kathy. 2014. *Constructing Grounded Theory: Second Edition*. *Constructing Grounded Theory*. SAGE Publications.

Charnigo, Laurie. 2009. “Lights! Camera! Action! Producing Library Instruction Video Tutorials Using Camtasia Studio.” *Journal of Library and Information Services in Distance Learning* 3 (1): 23–30. doi:10.1080/15332900902794880.

Chean, Swee Ling, Sin Ban Ho, and Ian Chai. 2018. “A Conceptual Framework on Constructing Effective Learning Content for Programming Novices.” *International Journal of Engineering & Technology* 7 (2.15): 1–5.

Cockburn, a., and A. Bryant. 1998. “Cleogo: Collaborative and Multi-Metaphor Programming for Kids.” *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No.98EX110)*. IEEE Comput. Soc, 189–94. doi:10.1109/APCHI.1998.704194.

Cohen, Louis, Lawrence Manion, and Keith Morrison. 2007. *Research Methods in Education, Sixth Edition*. *Routledge, Taylor & Francis Group*.

Cooper, Stephen, Wanda Dann, and Randy Pausch. 2000. “Alice: A 3-D Tool for Introductory Programming Concepts.” *Journal of Computing Sciences in Colleges* 15 (May 2000): 107–16. doi:10.1145/1953163.1953243.

Corneliussen, Hilde G., and Lin Prøitz. 2015. “Kids Code in a Rural Village in Norway: Could Code Clubs Be a New Arena for Increasing Girls’ Digital Interest and Competence?” *Information, Communication & Society* 19 (1): 95–110.

doi:10.1080/1369118X.2015.1093529.

Crawford, Stewart, and Elizabeth Boese. 2006. “ActionScript: A Gentle Introduction to Programming.” *Journal of Computing Sciences in Colleges*, no. 2: 156–68. http://dl.acm.org/citation.cfm?id=1089203.

Crick, Tom, and Sue Sentance. 2011. “Computing at School: Stimulating Computing Education in the UK.” *… International Conference on Computing Education …*, 122–23. doi:10.1145/2094131.2094158.

Csizmadia, Andrew, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby, and John Woollard. 2015. “Computational Thinking A Guide for Teachers.”

Curzon, Paul, Joan Peckham, Harriet Taylor, Amber Settle, and Eric Roberts. 2009. “Computational Thinking (CT): On Weaving It in.” *ACM SIGCSE …*, 201–2. doi:10.1145/1562877.1562941.

Daily, SB, AE Leonard, and Sophie Jörg. 2014. “Dancing Alice: Exploring Embodied Pedagogical Strategies for Learning Computational Thinking.” *Proceedings of the 45th …*, 1–6. http://dl.acm.org/citation.cfm?id=2538917.

Davis, Richard, Engin Bumbacher, Oceane Bel, Arnan Sipitakiat, and Paulo Blikstein. 2015. “Sketching Intentions: Comparing Different Metaphors for Programming Robots,” 391–394. doi:10.1145/2771839.2771924.

Denning, Peter J. 2009. “The Profession of ITBeyond Computational Thinking.” *Communications of the ACM* 52 (6): 28. doi:10.1145/1516046.1516054.

Department for Education. 2013. “Computing Programmes of Study : Key Stages 1 and 2 National Curriculum in England.” *Department for Education*. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf.

DeWalt, Katheleen M., and Billie R. DeWalt. 2011. *Dewalt 2011.pdf*. Edited by Kathleen M. DeWalt and Billie R. DeWalt. Second Edi. USA: Rowman Altamira.

Dewitt, Dorothy, Norlidah Alias, and Saedah Siraj. 2016. “Problem Solving Strategies of Malaysian Secondary School Teachers.” *Educational Technology World Conference 2016*, 1–14.

Diwakar, Shyam, Krishnashree Achuthan, Prema Nedungadi, and Bipin Nair. 2011. “Enhanced Facilitation of Biotechnology Education in Developing Nations via Virtual Labs: Analysis, Implementation and Case-Studies.” *International Journal of …* 3 (1). http://www.ijcte.org/papers/275-D007.pdf.

Dorling M. 2014. “Developing Creativity and Computational Thinking in Your Computing Classroom.” Birmingham UK: CAS hub meeting.

Duncan, Caitlin, Tim Bell, and James Atlas. 2017. “What Do the Teachers Think ? Introducing Computational Thinking in the Primary School Curriculum.” *Proceedings of the Nineteenth Australasian Computing Education Conference*, 65–74.

doi:10.1145/3013499.3013506.

Duncan, Caitlin, Tim Bell, and Steve Tanimoto. 2014. “Should Your 8-Year-Old Learn Coding?” *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE ’14*, 60–69. doi:10.1145/2670757.2670774.

Dyne, Michele Van, and Jeffrey Braun. 2014. “Effectiveness of a Computational Thinking (CS0) Course on Student Analytical Skills.” *Proceedings of the 45th ACM Technical Symposium on Computer Science Education - SIGCSE ’14*. New York, New York, USA: ACM Press, 133–38. doi:10.1145/2538862.2538956.

Ebrahimi, Ali, Shayan Geranzeli, and Taha Shokouhi. 2013. “Programming For Children ; ‘ Alice and Scratch Analysis ,’” 106–15.

Fereday, Jennifer, and Eimear Muir-Cochrane. 2006. “Demonstrating Rigor Using Thematic Analysis : A Hybrid Approach of Inductive and Deductive Coding and Theme Development.” *International Journal of Qualitative Methods* 5: 80–92. doi:10.1063/1.2011295.

Fitriza, R. 2012. “MACROMEDIA FLASH IN IMPROVING STUDENTS’ACHIEVEMENT AND LEARNING ACTIVITIES OF GEOMETRY AT GRADE IX OF MTsN SUNGAYANG,” 267–68. http://103.10.169.96/handle/123456789/458.

Franklin, Diana, Charlotte Hill, Hilary A. Dwyer, Alexandria K. Hansen, Ashley Iveland, and Danielle B. Harlow. 2016. “Initialization in Scratch: Seeking Knowledge Transfer.” *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE ’16)*, 217–22. doi:10.1145/2839509.2844569.

Franzoni, Ana Lidia, Management Sud Paris, Laboratorie Cemantic, Charles Fourier, and France Tel. 2009. “Student Learning Styles Adaptation Method Based on Teaching Strategies and Electronic Media” 12: 15–29.

Frith, Constance. 1997. “Motivation To Learn Components of the Motivation to Learn,” 1–10.

Gallardo, Daniel, CF Julià, and S Jorda. 2008. “TurTan: A Tangible Programming Language for Creative Exploration.” *Tabletop*. http://www.researchgate.net/publication/224344485_TurTan_A_tangible_programming_language_for_creative_exploration/file/3deec515ddc2f54355.pdf.

García, Ramón Rubio, Javier Suárez Quirós, Ramón Gallego Santos, Santiago Martín González, and Samuel Morán Fernanz. 2007a. “Interactive Multimedia Animation with Macromedia Flash in Descriptive Geometry Teaching.” *Computers & Education* 49 (3): 615–39. doi:10.1016/j.compedu.2005.11.005.

———. 2007b. “Interactive Multimedia Animation with Macromedia Flash in Descriptive Geometry Teaching.” *Computers & Education* 49 (3): 615–39. doi:10.1016/j.compedu.2005.11.005.

Garofalo, Joe, and Tim Summers. 2004. “Macromedia Flash as a Tool for Mathematics Teaching and Learning.” *School Science and Mathematics* 104 (2): 89.

http://search.ebscohost.com/login.aspx?direct=true%7B&%7DAuthType=cookie,ip,uid%7B&%7Ddb=eric%7B&%7DAN=EJ701623%7B&%7Dsite=ehost-live.

Gomez, Elizabeth Avery, Dezhi Wu, and Katia Passerini. 2010. "Computer-Supported Team-Based Learning: The Impact of Motivation, Enjoyment and Team Contributions on Learning Outcomes." *Computers & Education* 55 (1): 378–90. doi:10.1016/j.compedu.2010.02.003.

Gouws, LA, Karen Bradshaw, and Peter Wentworth. 2013. "Computational Thinking in Educational Activities: An Evaluation of the Educational Game Light-Bot." *… in Computer Science Education*, 10–15. http://dl.acm.org/citation.cfm?id=2466518.

Gouws, Lindsey, Karen Bradshaw, and Peter Wentworth. 2013. "First Year Student Performance in a Test for Computational Thinking." *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference on - SAICSIT '13*. New York, New York, USA: ACM Press, 271. doi:10.1145/2513456.2513484.

Gretter, Sarah, and Aman Yadav. 2016. "Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills." *TechTrends* 60 (5). TechTrends: 510–16. doi:10.1007/s11528-016-0098-4.

Grover, S., and R. Pea. 2013. "Computational Thinking in K-12: A Review of the State of the Field." *Educational Researcher* 42 (1): 38–43. doi:10.3102/0013189X12463051.

Guzdial, Mark. 2003. "Programming Environments for Novices Specializing Environments for Novices Logo and Its Descendants : The Goal of Compu- Tational Literacy," 1–24.

———. 2008. "EducationPaving the Way for Computational Thinking." *Communications of the ACM* 51 (8): 25. doi:10.1145/1378704.1378713.

Habgood, Matthew Peter Jacob. 2010. "The Effective Integration of Digital Games and Learning Content." *Learning Sciences Research Institute*. http://etheses.nottingham.ac.uk/385/.

Hassan, Johari, and Noor Asmalinda Masrom. 2010. "Modul Pembelajaran Kendiri (MPK) Subjek Information And Communication Technology (ICT) Tingkatan 4."

Hawk, Thomas F. 2007. "To Enhance Student Learning" 5 (1): 1–19.

Heins, Tanya, and Frances Himes. 2002. "Creating Learning Objects with Macromedia Flash MX." *Macromedia White Paper*, no. April. http://www.mystery-productions.com/hyper/flash_mxlo.pdf.

Henderson, Peter B., Thomas J. Cortina, and Jeannette M. Wing. 2007. "Computational Thinking." *ACM SIGCSE Bulletin* 39 (1): 195. doi:10.1145/1227504.1227378.

Holzinger, Andreas, and Martin Ebner. 2003a. "Interaction and Usability of Simulations & Animations : A Case Study of the Flash Technology" 780: 777–80.

———. 2003b. "Interaction and Usability of Simulations & Animations: A Case Study of the Flash Technology." *Human-Computer Interaction INTERACT* 780: 777–80. http://lamp.tu-graz.ac.at/~i203/ebner/publication/03_zuerich.pdf.

Horn, Michael S., Erin Treacy Solovey, R. Jordan Crouser, and Robert J.K. Jacob. 2009. "Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education." *Proceedings of the 27th International Conference on Human Factors in Computing Systems - CHI 09*. New York, New York, USA: ACM Press, 975. doi:10.1145/1518701.1518851.

Hu, Chenglie. 2011. "Computational Thinking – What It Might Mean and What We Might Do about It." *ITiCSE '11: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 223–27. doi:10.1145/1999747.1999811.

Huang, Wenming, Zhenrong Deng, and Dong Rongsheng. 2009. "Programming Courses Teaching Method for Ability Enhancement of Computational Thinking." *2009 International Association of Computer Science and Information Technology - Spring Conference*. Ieee, 182–85. doi:10.1109/IACSIT-SC.2009.52.

Husain, Hafizah, Noorfazila Kamal, and Mohd Faisal Ibrahim. 2017. "ENGENDERING PROBLEM SOLVING SKILLS AND MATHEMATICAL KNOWLEDGE VIA PROGRAMMING 2 . Malaysian Curriculum for Mathematics," 1–11.

Ibrahim, R., and A Jaafar. 2010. "Using Educational Games in Learning Introductory Programming: A Pilot Study on Students' Perceptions" 1 (form 4): 1–5. doi:10.1109/ITSIM.2010.5561414.

Ibrahim, Roslina, and RCM Yusoff. 2010. "Students Perceptions of Using Educational Games to Learn Introductory Programming." *… and Information Science* 4 (1): 205–16. http://www.ccsenet.org/journal/index.php/cis/article/view/8246.

Jenkins, Craig. 2015. "A Work in Progress Paper : Evaluating a Microworlds- Based Learning Approach for Developing Literacy and Computational Thinking in Cross-Curricular Contexts." *ACM WipSCE'15*, 1–5.

Jenkins, Tony. 2001a. "T EACHING P ROGRAMMING – A J OURNEY FROM T EACHER TO M OTIVATOR." *Anuual LTSN-ICS Conference in London*.

———. 2001b. "The Motivation of Students of Programming." *ACM SIGCSE Bulletin* 33 (3): 53–56. doi:10.1145/507758.377472.

———. 2002. "On the Difficulty of Learning to Program." *LTSN-ICS Conference*, 53–58.

Jenkins, Tony, and J Davy. 2002. "Diversity and Motivation in Introductory Programming." *Innovation in Teaching and Learning …*. http://journals.heacademy.ac.uk/doi/abs/10.11120/ital.2002.01010003.

Kahn, K. 1996. "ToonTalk TM—An Animated Programming Environment for Children." *Journal of Visual Languages & Computing* 7 (2): 197–217. doi:10.1006/jvlc.1996.0011.

Kalelioğlu, Filiz, Yasemin Gülbahar, and Dilek Doğan. 2017. "Teaching How to Think Like a Programmer," no. October: 18–35. doi:10.4018/978-1-5225-3200-2.ch002.

Kalganova, T. 2001a. "Improving Student Motivation by Means of Multimedia."

———. 2001b. “Improving Students Motivation by Means of Multimedia.” http://dspace.brunel.ac.uk/handle/2438/2520.

Kawulich, Barbara B. 2005. “Participant Observation as a Data Collection Method.” *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 6 (2): 1–19. doi:10.1177/14687941030032003.

Kazimoglu, Cagin, Mary Kiernan, Liz Bacon, and Lachlan Mackinnon. 2012. “A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming.” *Procedia - Social and Behavioral Sciences* 47 (January): 1991–99. doi:10.1016/j.sbspro.2012.06.938.

Kelleher, Caitlin, Dennis Cosgrove, and David Culyba. 2002. “Alice2: Programming without Syntax Errors.” *User Interface Software …*, 3–4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4640&rep=rep1&type=pdf.

Kelleher, Caitlin, Randy Pausch, and Sara Kiesler. 2007. “Storytelling Alice Motivates Middle School Girls to Learn Computer Programming.” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI ’07*. New York, New York, USA: ACM Press, 1455. doi:10.1145/1240624.1240844.

Khambari, Mas Nida, Su Luan Wong, Ahmad Fauzi, Mohd Ayub, and Priscilla Moses. 2010. “Teachers ’ Technology Professional Development : A Malaysia Perspective,” 758–62.

Kofoed, LB, and Lars Reng. 2010. “Enhance Students’ Motivation to Learn Programming by Using Direct Visual Feed-Back.” *Engineering Sustainability for a Global …*. http://forskningsbasen.deff.dk/Share.external?sp=Sb3603bcf-6345-4745-b49e-2827d26da60a&sp=Saau.

Koh, KH, Alexander Repenning, and H Nickerson. 2013. “Will It Stick?: Exploring the Sustainability of Computational Thinking Education through Game Design.” *… Science Education*, 597–602. http://dl.acm.org/citation.cfm?id=2445372.

Koh, Kyu Han, Ashok Basawapatna, Vicki Bennett, and Alexander Repenning. 2010. “Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning.” *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, September. Ieee, 59–66. doi:10.1109/VLHCC.2010.17.

Kolb, Sharon M. 2012. “Grounded Theory and the Constant Comparative Method : Valid Research Strategies for Educators.” *Journal of Emerging Trends in Educational Research and Policy Studies* 3 (1): 83–86. http://jeteraps.scholarlinkresearch.com/articles/Grounded Theory and the Constant Comparative Method.pdf.

Köse, Utku. 2010. “A Web Based System for Project-Based Learning Activities In ‘web Design and Programming’ course.” *Procedia - Social and Behavioral Sciences* 2 (2): 1174–84. doi:10.1016/j.sbspro.2010.03.168.

Kramer, Jeff. 2007. “Is Abstraction the Key to Computing?” *Communications of the ACM* 50 (4): 36–42. doi:10.1145/1232743.1232745.

Krosnick, Jon a., and Stanley Presser. 2010. “Question and Questionnaire Design.” *Handbook of Survey Research* 94305: 886. doi:10.1111/j.1432-1033.1976.tb10115.x.

L’Heureux, J, and Deborah Boisvert. 2012. “IT Problem Solving: An Implementation of Computational Thinking in Information Technology.” *Proceedings of the 13th …*, 183–88. http://dl.acm.org/citation.cfm?id=2380606.

Lamb, Annette, and Larry Johnson. 2006. “Flash: Engaging Learners through Animation, Interaction, and Multimedia,” 54–57. http://olms.noinc.com/olms/data/resource/3044/Technology and the Science of Learning - week 4 reading Flash engaging learners.pdf.

Lee, Irene, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. “Computational Thinking for Youth in Practice.” *ACM Inroads* 2 (1): 32. doi:10.1145/1929887.1929902.

Lee, Youngju, and Jihyun Lee. 2014. “Enhancing Pre-Service Teachers’ Self-Efficacy Beliefs for Technology Integration through Lesson Planning Practice.” *Computers & Education* 73 (April). Elsevier Ltd: 121–28. doi:10.1016/j.compedu.2014.01.001.

Lenzner, Timo, Cornelia Neuert, and Wanda Otto. 2016. “Cognitive Pretesting: GESIS Survey Guidelines,” 1–9. doi:10.15464/gesis-sg_en_010.

Leon E. Winslow. 1996. “Programming Pedagogy -- A Psychological Overview.” *SIGCSE Bulletin* 28 (3): 17–23.

Levin, SR, JG Buell, and JA Levin. 2000. “Journal of Computing in Teacher Education.” *Journal of Computing in Teacher Eduction* Vol. 18/ N (January 2015): 37–41. doi:10.1080/10402454.2002.10784439.

Lezam Lehat, Mohd, Rashidah Mokhtar, Nora Mohd Basir, and Yusnita Sokma. 2015. “Programming: A Study on Upper Primary Students’ Attitude towards Education Digital Games.” *International Journal of Computer Theory and Engineering* 7 (3): 231–35. doi:10.7763/IJCTE.2015.V7.962.

Li, Chen, Shengling Xiao, and ShiZhou You. 2010. “Development of the Packaging Engineering Experimental Teaching Platform Based on Interaction Design Concept.” *2010 Second International Conference on Multimedia and Information Technology*. Ieee, 162–65. doi:10.1109/MMIT.2010.79.

Lin, Chien-Heng. 2012. “An Innovative Change in Technology Integration: Training Pre-Service Kindergarten Teachers to Be Courseware Designers.” *Creative Education* 3 (7): 1177–83. doi:10.4236/ce.2012.37175.

Linda, W, Jill Denner, Michelle Bliesner, Pat Rex, Carbonero Way, and Scotts Valley. 2009. “Can Middle-Schoolers Use Storytelling Alice to Make Games ? Results of a Pilot Study,” 207–14.

Ling, U. L., T. C. Saibin, J. Labadin, and N. A. Aziz. 2017. “Preliminary Investigation: Teachers’ Perception on Computational Thinking Concepts.” *Journal of Telecommunication, Electronic and Computer Engineering* 9 (2): 2289–8131.

Ling, Xu, and He Wei. 2014. “VB . NET Programming Course Teaching Based on Computational Thinking,” no. Icibet: 58–60.

Lister, Raymond, and John Leaney. 2003. “Introductory Programming, Criterion-Referencing, and Bloom.” *ACM SIGCSE Bulletin* 35: 143. doi:10.1145/792548.611954.

Lye, Sze Yee, and Joyce Hwee Ling Koh. 2014. “Review on Teaching and Learning of Computational Thinking through Programming: What Is next for K-12?” *Computers in Human Behavior* 41. Elsevier Ltd: 51–61. doi:10.1016/j.chb.2014.09.012.

Lyons, Alendra, Stephen Reysen, and Lindsey Pierce. 2012. “Video Lecture Format, Student Technological Efficacy, and Social Presence in Online Courses.” *Computers in Human Behavior* 28 (1). Elsevier Ltd: 181–86. doi:10.1016/j.chb.2011.08.025.

Maguire, Martin. 2001. “Methods to Support Human-Centred Design.” *International Journal of Human-Computer Studies* 55 (4): 587–634. doi:10.1006/ijhc.2001.0503.

Malaysia Education Blueprint, Malaysia. 2013. “Malaysia Education Blueprint 2013 - 2025.” *Education* 27 (1): 1–268. doi:10.1016/j.tate.2010.08.007.

Maloney, John, Mitchel Resnick, and Natalie Rusk. 2010. “The Scratch Programming Language and Environment.” *ACM Transactions on Computing Education* 10 (4): 1–15. doi:10.1145/1868358.1868363.http.

Mariappan, Muralindran, Jong Chia Sing, and Manimehala Nadarajan. 2015. “A Design Methodology of Programmable Tangible Blocks for Early Childhood Educational Robotic System.” *Journal of Applied Sciences Research* 11: 17–25.

Martins, Scheila Wesley, Antonio Jose Mendes, and Antonio Dias Figueiredo. 2010. “A Strategy to Improve Student’s Motivation Levels in Programming Courses.” *2010 IEEE Frontiers in Education Conference (FIE)*, October. Ieee, F4F–1–F4F–7. doi:10.1109/FIE.2010.5673366.

Matthews, Reginamary. 2014. “Learning Objects To Improve Cognitive Understanding in Learning Introductory Programming,” no. February: 251.

Mayer, R E. 2004. “Should There Be a Three-Strike Rule against Pure Discovery Learning?” *American Psychologist* 59 (1): 14–19. doi:10.1037/0003-066X.59.1.14.

Meluso, Angela, Meixun Zheng, Hiller a. Spires, and James Lester. 2012. “Enhancing 5th Graders’ Science Content Knowledge and Self-Efficacy through Game-Based Learning.” *Computers & Education* 59 (2). Elsevier Ltd: 497–504. doi:10.1016/j.compedu.2011.12.019.

Miao, Liang, and Bin Zhong. 2011. “The Construction of Educational Software Development Platform Based on Flash.” *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, August. Ieee, 4879–82. doi:10.1109/EMEIT.2011.6024056.

Mishra, Punya, and Aman Yadav. 2013. “Of Art and Algorithm: Rethinking Technology & Creativity in the 21st Century.” *TechTrends* 57 (3): 10–14. doi:10.1007/s11528-013-

0668-7.

Morse, Janice M., Michael Barrett, Maria Mayan, Karin Olson, and Jude Spiers. 2002. “Verification Strategies for Establishing Reliability and Validity in Qualitative Research.” *International Journal of Qualitative Methods* 1 (2): 13–22. doi:10.1063/1.2011328.

Muratet, Mathieu, Elisabeth Delozanne, Patrice Torguet, and Fabienne Viallet. 2012. “Serious Game and Students’ Learning Motivation: Effect of Context Using prog&play,” 123–128. doi:10.1007/978-3-642-30950-2_16.

Muratet, Mathieu, Patrice Torguet, Jean-Pierre Jessel, and Fabienne Viallet. 2009. “Towards a Serious Game to Help Students Learn Computer Programming.” *International Journal of Computer Games Technology* 2009: 1–12. doi:10.1155/2009/470590.

Murphy, Elizabeth, Robert Dingwall, Michael Parker, M E Luczun, F Maggs-Rapport, S. Thorne, D. L. Altheide, et al. 2004. “Qualitative Data Analysis.” *Social Science and Medicine*. doi:10.1080/0141192042000237220.

Najjar, Lawrence J. 1996. “Multimedia Information and Learning” 5: 1–16.

National Research Council. 2011. *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. doi:10.17226/13170.

Ng, Wan. 2012. “Can We Teach Digital Natives Digital Literacy?” *Computers and Education* 59 (3). Elsevier Ltd: 1065–78. doi:10.1016/j.compedu.2012.04.016.

Ningyu ZHANG, Gautam BISWAS, and Institute. 2017. “Assessing Students’ Computational Thinking in a Learning by Modeling Environment.” In *Conference Proceedings of International Conference on Computational Thinking Education 2017*, 164–67. Singapore.

Osman, Mohamad Amin, Siew Phaik Loke, Mohamed Nordin Zakaria, and Alan G. Downe. 2012. “Secondary Students’ Perfectionism and Their Response to Different Programming Learning Tools.” *2012 IEEE Colloquium on Humanities, Science and Engineering (CHUSER)*, no. Chuser (December). Ieee: 584–88. doi:10.1109/CHUSER.2012.6504380.

Papastergiou, Marina. 2009. “Digital Game-Based Learning in High School Computer Science Education: Impact on Educational Effectiveness and Student Motivation.” *Computers & Education* 52 (1). Elsevier Ltd: 1–12. doi:10.1016/j.compedu.2008.06.004.

Parberry, Ian, Max B Kazemzadeh, and Timothy Roden. 2006. “The Art and Science of Game Programming [Extended Abstract],” 510–14.

Pausch, Randy, and Wanda Dann. 2000. “ALICE : A 3-D TOOL FOR INTRODUCTORY PROGRAMMING CONCEPTS.” *Journal of Computing Sciences in Colleges2*, 107–16.

Pea, Roy D. 1983. “Logo Programming and Problem Solving.” *Conference Paper* 150 (ir 014 383): 1–10.

Pedersen, Susan. 2003. “Teachers ’ Beliefs about Issues in the Implementation of a Student-Centered Learning Environment” 51 (2): 57–76.

Phuong, Dinh Dong, Fumiko Harada, and Hiromitsu Shimakawa. 2009. “Encouraging Programming Learning for Novices with Grouping and Convincing Opinions.” *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*. Ieee, 283–88. doi:10.1109/ICCIT.2009.168.

Portelance, Dylan J, and College Ave. 2015. “Code and Tell : Assessing Young Children ’ S Learning of Computational Thinking Using Peer Video Interviews with ScratchJr,” 271–74. doi:10.1145/2771839.2771894.

Poulou, Maria. 2005. “Educational Psychology within Teacher Education.” *Teachers and Teaching: Theory and Practice* 11 (6): 555–74. doi:10.1080/13450600500293241.

Prensky, Marc. 2001. “The Digital Game-Based Learning Revolution,” no. c: 1–20.

———. 2003. “Digital Game-Based Learning.” *Computers in Entertainment* 1 (1): 21. doi:10.1145/950566.950596.

QuickStart Computing. 2014. “Computational Thinking.” *Technical Report*, 12. doi:10.1145/1118178.1118215.

Raadt, M de. 2008. “Teaching Programming Strategies Explicitly to Novice Programmers.” http://eprints.usq.edu.au/4827.

Radošević, Danijel, Tihomir Orehovački, and Alen Lovrenčić. 2009. “New Approaches and Tools in Teaching Programming.” *Proceedings of 20th Central …*, 49–57. http://bib.irb.hr/datoteka/427643.CECIIS2009_Radosevic_Orehovacki_Lovrencic.pdf.

Rajaravivarma, Rathika. 2005. “A Games-Based Approach for Teaching the Introductory Programming Course.” *ACM SIGCSE Bulletin* 37 (4): 98. doi:10.1145/1113847.1113886.

Ramli, Rizauddin, Melor Md Yunus, and Noriah Mohd Ishak. 2010. “Experience of Robotic Teaching for Malaysian Gifted Enrichment Program at PERMATApintar.” *Proceedings of the 9th WSEAS International Conference on Education and Educational Technology (EDU’10)*, 163–66.

Reimers, Stian, and Neil Stewart. 2007. “Adobe Flash as a Medium for Online Experimentation: A Test of Reaction Time Measurement Capabilities.” *Behavior Research Methods* 39 (3): 365–70. doi:10.3758/BF03193004.

Resnick, B Y, Karen Brennan, and Yasmin Kafai. 2009. “Scratch : Programming for All.” *Communications of the ACM* 52: 60–67.

Resnick, Mitchel. 2007. “All I Really Need to Know (about Creative Thinking) I Learned (by Studying How Children Learn) in Kindergarten.” *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition - C&C ’07*. New York, New York, USA: ACM Press, 1–6. doi:10.1145/1254960.1254961.

Resnick, Mitchel, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, et al. 2009. “Scratch.” *Communications of the ACM*. doi:10.1145/1592761.1592779.

Ronchetti, Marco. 2010. “Using Video Lectures to Make Teaching More Interactive.”

International Journal of Emerging Technologies in Learning (iJET) 5 (2): 45–48. doi:10.3991/ijet.v5i2.1156.

Routledge, Graham. 2007. "Developing Understanding of Programming Principles Using Flash Actionscript." *Innovation in Teaching …* 6 (4). http://journals.heacademy.ac.uk/doi/abs/10.11120/ital.2007.06040051.

Routledge, Graham, Amin Aminaei, and Phillip Benachour. 2007. "Developing Understanding of Programming Principles Using Flash Actionscript" 6 (4): 51–71.

Rudder, A, Margaret Bernard, and S Mohammed. 2007. "Teaching Programming Using Visualization." *Proceedings of the Web …*. http://www.actapress.com/PaperInfo.aspx?PaperID=30003.

Ryan, Gery W., and H. Russell Bernard. 2003. "Techniques to Identify Themes." *Field Methods* 15 (1): 85–109. doi:10.1177/1525822X02239569.

Ryan, Rm, and El Deci. 2000. "Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions." *Contemporary Educational Psychology* 25 (1): 54–67. doi:10.1006/ceps.1999.1020.

Saari, EM, P.Blanchfield, and G.Hopkins. 2015. "CSEDU_2015_89-1." Nottingham.

Saari, Erni Marlina. 2014. "Motivation for Trainee Teachers : Non-Computing Background Learn Action Script" 3 (1): 133–38.

Saari, Erni Marlina, Peter Blanchfield, and Gail Hopkins. 2015. "Learning Computational Thinking through the Use of Flash Action Scripts: Preparing Trainee Elementary School Teachers for Teaching Computer Programming." *CSEDU 2015 - 7th International Conference on Computer Supported Education, Proceedings* 2: 75–84. doi:10.5220/0005442600750084.

———. 2016. *Computer Supported Education*. Edited by Susan Zvacek, Maria Teresa Restivo, James Uhomoibhi, and Markus Helfert. *Computer Supported Education*. Vol. 583. Communications in Computer and Information Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-29585-5.

Saari, Ms Erni Marlina, Peter Blanchfield, and Gail Hopkins. n.d. "Learning Computational Thinking through the Use of Flash Action Scripts."

Sáez-López, José-Manuel, Marcos Román-González, and Esteban Vázquez-Cano. 2016. "Visual Programming Languages Integrated across the Curriculum in Elementary School: A Two Year Case Study Using 'scratch' in Five Schools." *Computers & Education* 97. doi:http://dx.doi.org/10.1016/j.compedu.2016.03.003.

Sands, Mike, Nick Moukhine, and G Blank. 2008. "Widening the Pipeline of K-12 Students with Flash." *Journal of Computing Sciences in …*, 52–57. http://dl.acm.org/citation.cfm?id=1352637.

Sang, Guoyuan, Martin Valcke, Johan Van Braak, and Jo Tondeur. 2010. "Student Teachers' Thinking Processes and ICT Integration: Predictors of Prospective Teaching Behaviors

with Educational Technology." *Computers & Education* 54 (1). Elsevier Ltd: 103–12. doi:10.1016/j.compedu.2009.07.010.

Sardina, Sebastian, and Stavros Vassos. 2005. "The Wumpus World in IndiGolog: A Preliminary Report." In *Proceedings of the Workshop on Non-Monotonic Reasoning, Action and Change at IJCAI (NRAC-05)*, 90–95.

Seiter, Linda, and Brendan Foreman. 2013. "Modeling the Learning Progressions of Computational Thinking of Primary Grade Students." *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research - ICER '13*, 59. doi:10.1145/2493394.2493403.

Selby, C, and John Woollard. 2013. "Computational Thinking: The Developing Definition," 5–8. http://eprints.soton.ac.uk/356481/.

Seligman, Martin E. P., and Mihaly Csikszentmihalyi. 2000. "Positive Psychology: An Introduction." *American Psychologist* 55 (1): 5–14. doi:10.1037/0003-066X.55.1.5.

Sentance, Sue, and Andrew Csizmadia. 2016. "Computing in the Curriculum: Challenges and Strategies from a Teacher???s Perspective." *Education and Information Technologies*, 1–27. doi:10.1007/s10639-016-9482-0.

Sherrell, Jake a. Qualls and Linda B. 2010. "Why Computational Thinking Should Be." *Journal of Computing Sciences in Colleges*, 66–71. http://delivery.acm.org/10.1145/1750000/1747148/p66-qualls.pdf?ip=190.90.77.86&id=1747148&acc=PUBLIC&key=4D4702B0C3E38B35.4D4702B0C3E38B35.FA5FD23F4BC440C3.4D4702B0C3E38B35&CFID=338674104&CFTOKEN=80732900&__acm__=1400019651_06d8eea82c3b6f67a444131d9d2b49f.

Singh, By Karamjit. 2016. "Computational Thinking Comes to the Fore in Malaysian Schools." *Www.digitalnesasia.com/digital-Economy*.

Smyth, Jolene D., Don A. Dillman, Leah Melani Christian, and Mallory Mcbride. 2009. "Open-Ended Questions in Web Surveys." *Public Opinion Quarterly* 73 (2): 325–37. doi:10.1093/poq/nfp029.

Sorapure, Madeleine. 2006. "Text, Image, Code, Comment: Writing in Flash." *Computers and Composition* 23 (4): 412–29. doi:10.1016/j.compcom.2006.08.001.

Stansfield, J, Brian Carr, and Ira Coldstein. 1976. "Wumpus Advisor."

Strauss, Anselm, and Juliet Corbin. 1994. "Grounded Theory Methodology." *Handbook of Qualitative Research*. doi:10.1007/BF00988593.

Strawhacker, Amanda, and Marina U. Bers. 2015. "'I Want My Robot to Look for Food': Comparing Kindergartner's Programming Comprehension Using Tangible, Graphic, and Hybrid User Interfaces." *International Journal of Technology and Design Education* 25 (3). Springer Netherlands: 293–319. doi:10.1007/s10798-014-9287-7.

Styles, L, and Is There. 2012. "Learning and Teaching Styles." *International Perspectives on Education* 78 (June): 674–81.

http://www.researchgate.net/publication/236261199_Comparative_Education_Teacher_Training_Education_Policy_School_Leadership_and_Social_Inclusion/file/60b7d5176c814cd615.pdf#page=407.

Suliman, Azizah, Rozita Hawari, and Marini Othman. 2011. "A Preliminary Study on Teaching Programming At Malaysian School." *International Conference on Computing and Informatics, ICOCI*, no. 3: 220–25. http://repo.uum.edu.my/13649/1/220.pdf.

Takemura, Yasuhiro. 2007. "Analyzing the Motivation of the Students in the Art Faculty for Learning Programming Department of Social Welfare and Psychology," no. Icalt: 0–2.

Takemura, Yasuhiro, Hideo Nagumo, Kuo-Li Huang, and Kenichi Matsumoto. 2007. "Analysis of the Relation between the Teaching Materials and Motivation in Programming Education." *20th Conference on Software Engineering Education & Training (CSEET'07)*, July. Ieee, 317–26. doi:10.1109/CSEET.2007.10.

Tan, Jessica. 2015. "Computational Thinking an Essential Skill for next-Generation." *www.todayonline.com/Singapore/computational-Thinking-Essential-Skill-next-Generation*.

Tan, Shin Yen, and Siti Hajar Halili. 2015. "Effective Teaching of Higher-Order Thinking (HOT) in Education." *The Online Journal of Distance Education and E-Learning* 3 (2): 41–47.

Tarkan, Sureyya, Vibha Sazawal, Allison Druin, Evan Golub, Elizabeth M Bonsignore, Greg Walsh, Zeina Atrash, and College Park. 2010. "Toque : Designing a Cooking-Based Programming Language For and With Children," 2417–26.

Teijlingen, Edwin R. van, and Vanora Hundley. 2001. "The Importance of Pilot Study." *Social Research Update*, no. 35. doi:10.7748/ns2002.06.16.40.33.c3214.

Thomas, David R. 2003. "A General Inductive Approach for Qualitative Data Analysis." *Population English Edition* 27 (2): 237–46. doi:10.1177/1098214005283748.

Tondeur, Jo, Johan van Braak, Guoyuan Sang, Joke Voogt, Petra Fisser, and Anne Ottenbreit-Leftwich. 2012. "Preparing Pre-Service Teachers to Integrate Technology in Education: A Synthesis of Qualitative Evidence." *Computers & Education* 59 (1). Elsevier Ltd: 134–44. doi:10.1016/j.compedu.2011.10.009.

Tsukamoto, Hidekuni. 2012. "Change of Students ' Motivation in an Introductory Programming Course for Non- Computing Majors," 4–5. doi:10.1109/ICALT.2012.38.

Turner, Daniel W. 2010. "Qualitative Interview Design : A Practical Guide for Novice Investigators." *The Qualitative Report* 15 (3): 754–60. http://mediaclass.co.uk/x9508/Turner - Qualitative Interview Design.pdf.

Umar, Irfan Naufal, and Amat Sazali Abu Hassan. 2015. "Malaysian Teachers' Levels of ICT Integration and Its Perceived Impact on Teaching and Learning." *Procedia - Social and Behavioral Sciences* 197 (February). Elsevier B.V.: 2015–21. doi:10.1016/j.sbspro.2015.07.586.

Utting, I, S Cooper, and M Kölling. 2010. "Alice, Greenfoot, and Scratch--a Discussion." *ACM*

Transactions on … 10 (4): 1–11. doi:10.1145/1868358.1868364.http.

Vizcaíno, Aurora, Juan Contreras, Jesús Favela, and Manuel Prieto. 2000. "An Adaptive, Collaborative Environment to Develop Good Habits in Programming." *Intelligent Tutoring Systems*, 262–71. http://link.springer.com/chapter/10.1007/3-540-45108-0_30.

Vs, Videogames, Block Building, and Mohd Syarqawy Hamzah. 2011. "School of Computer Science," no. September.

Wang, Danli, Tingting Wang, and Zhen Liu. 2014. "A Tangible Programming Tool for Children to Cultivate Computational Thinking." *The Scientific World Journal* 2014: 1–30. doi:10.1155/2014/428080.

Wang, Ling, Peggy a. Ertmer, and Timothy J. Newby. 2004. "Increasing Preservice Teachers' Self-Efficacy Beliefs for Technology Integration." *Journal of Research on Technology in Education* 36 (3): 231–50. doi:10.1080/15391523.2004.10782414.

Weese, Joshua Levi. 2016. "Mixed Methods for the Assessment and Incorporation of Computational Thinking in K-12 and Higher Education." *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16*, 279–80. doi:10.1145/2960310.2960347.

Weintrop, David, Elham Beheshti, Michael S. Horn, Kai Orton, Laura Trouille, Kemi Jona, and Uri Wilensky. 2014. "Interactive Assessment Tools for Computational Thinking in High School STEM Classrooms." *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST* 136 LNICST: 22–25. doi:10.1007/978-3-319-08189-2_3.

Werner, Linda, Jill Denner, and Shannon Campe. 2012. "The Fairy Performance Assessment : Measuring Computational Thinking in Middle School." In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education - SIGCSE '12*, 7–12. doi:10.1145/2157136.2157200.

Wever, B. De, T. Schellens, M. Valcke, and H. Van Keer. 2006. "Content Analysis Schemes to Analyze Transcripts of Online Asynchronous Discussion Groups: A Review." *Computers and Education* 46 (1): 6–28. doi:10.1016/j.compedu.2005.04.005.

Whitton, Nicola. 2007. "Motivation and Computer Game Based Learning," 1063–67.

Williams, Laurie A, and Robert R Kessler. 2000. "All I Really Need to Know about Pair Programming I Learned in Kindergarten." *Communications of the ACM* 43: 108–14. doi:10.1145/332833.332848.

Willis, Anne-marie, and Anne-marie Willis. 2016. "User-Centred Design User-Centred Design" 7136 (August).

Wing, J. 2011. "Research Notebook: Computational thinking—What and Why?" *The Link Magazine*, June 23, 2015. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

Wing, Jeannette M. 2006a. “Computational Thinking CS @ CMU and Grand Vision for the Field.”

———. 2008. “Computational Thinking and Thinking about Computing.” *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 366 (1881): 3717–25. doi:10.1098/rsta.2008.0118.

Wing, Jeannette M. 2006b. “Computational Thinking.” doi:10.1145/1118178.1118215.

Wong, Yoke Seng, Maizatul Hayati binti Mohammad Yatim, and Wee Hoe Tan. 2016. “Examining Effectiveness of Learning Object-Oriented Programming Paradigm Through Propriety Game-Based Learning Games.” *European Conference on Games Based Learning*, 796–804. https://search.proquest.com/docview/1859715138?accountid=15870.

Wyeth, P., and Gordon Wyeth. 2001. “Electronic Blocks: Tangible Programming Elements for Preschoolers.” *Proceedings of the Eighth IFIP TC13 Conference on Human-Computer Interaction*, 496–503. http://archive.itee.uq.edu.au/~peta/WyethInteract.pdf.

Yadav, Aman, Hai Hong, and Chris Stephenson. 2016. “Computational Thinking for All : Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms.” *TechTrends*. TechTrends, 10–13. doi:10.1007/s11528-016-0087-7.

Yadav, Aman, Chris Mayfield, Ninger Zhou, Susanne Hambrusch, and John T. Korb. 2014. “Computational Thinking in Elementary and Secondary Teacher Education.” *ACM Transactions on Computing Education* 14 (1): 1–16. doi:10.1145/2576872.

Yeh, Kuo-Chuan (Martin), Ying Xie, and Fengfeng Ke. 2011. “Teaching Computational Thinking to Non-Computing Majors Using Spreadsheet Functions.” *2011 Frontiers in Education Conference (FIE)*, October. Ieee, F3J–1–F3J–5. doi:10.1109/FIE.2011.6142980.

Yin, Robert K. 2009. “Case Study H Researc Design and Methods Fourth Edition.” In *Applied Social Research Methods Seiries*.

Yinnan, Zhang, and Luo Chaosheng. 2012. “Training for Computational Thinking Capability on Programming Language Teaching.” *2012 7th International Conference on Computer Science & Education (ICCSE)*, no. Iccse (July). Ieee: 1804–9. doi:10.1109/ICCSE.2012.6295420.

Zhong, B., Q. Wang, J. Chen, and Y. Li. 2015. “An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking.” *Journal of Educational Computing Research*, no. 122: 735633115608444. doi:10.1177/0735633115608444.

Züll, Cornelia. 2016. “Kognitives Pretesting.” *GESIS Survey Guidelines* Version 2. (December): 1–7. doi:10.15465/gesis-sg.

Zyda, M. 2005. “From Visual Simulation to Virtual Reality to Games.” *Computer* 38 (9): 25–32. doi:10.1109/MC.2005.297.

APPENDIX A



PhD. RESEARCH

Toolbox For Adopting Computational Thinking Through Learning Flash

CONSENT FORM

This is to confirm that I have agreed to take part in a research trial for the period _____ to _____ . I have read the information sheet provided and I understand what is involved.

I understand that the trial will gather data on my personal view of points and I agree that the use of this data in an anonymised form in the related research.

I understand that I can withdraw at any time, by returning the equipment to the registration point, and my personal data will be erased from the records.

I confirm that I am over the age of 21.

I give permission for the data collected to be used in subsequent associated research.

Signed :

Name :

Address :

Contact No :

Email :

APPENDIX B



Toolbox For Adopting Computational Thinking Through Learning Flash

PhD STUDY

This Phd Research will look on how to motivate students (master in education and technology or trainee teachers) to learn programming (flash programming). The task has been chosen to use the system and there will be a short interview and you are asked to answer a few questions related to this research. The interview will take about 30-40 minutes.

The data collected will be used to analyze how effective the system is and will it increase the motivation among non-technical students to learn flash programming. You may contact me at any time for information about research or in relation to your consent, my address are:

Your data will be stored in accordance with the Data Protection Act 1998, namely on a password protected drive in a secure facility and only for duration for which it is required. It will only accessible by those directly involved in this research.

You have been chosen to participate in this study as someone who might be representative of using this type of system. You may withdraw consent form from the experiment at any time during or after the task for any reason without penalty by contacting me on the address above. In this event all the data will be erased. All the data will be anonymised before publication; unless you have given specific consent data to be published that could identify you. Publishing data will result in information becoming available through the internet to anyone who wishes to access it.

RESEARCHER:

ERNI MARLINA SAARI

B49 Learning Technology Research Group

SUPERVISORS:

DR. PETER BLANCHFIELD

DR. GAIL HOPKINS

School of Computer Science

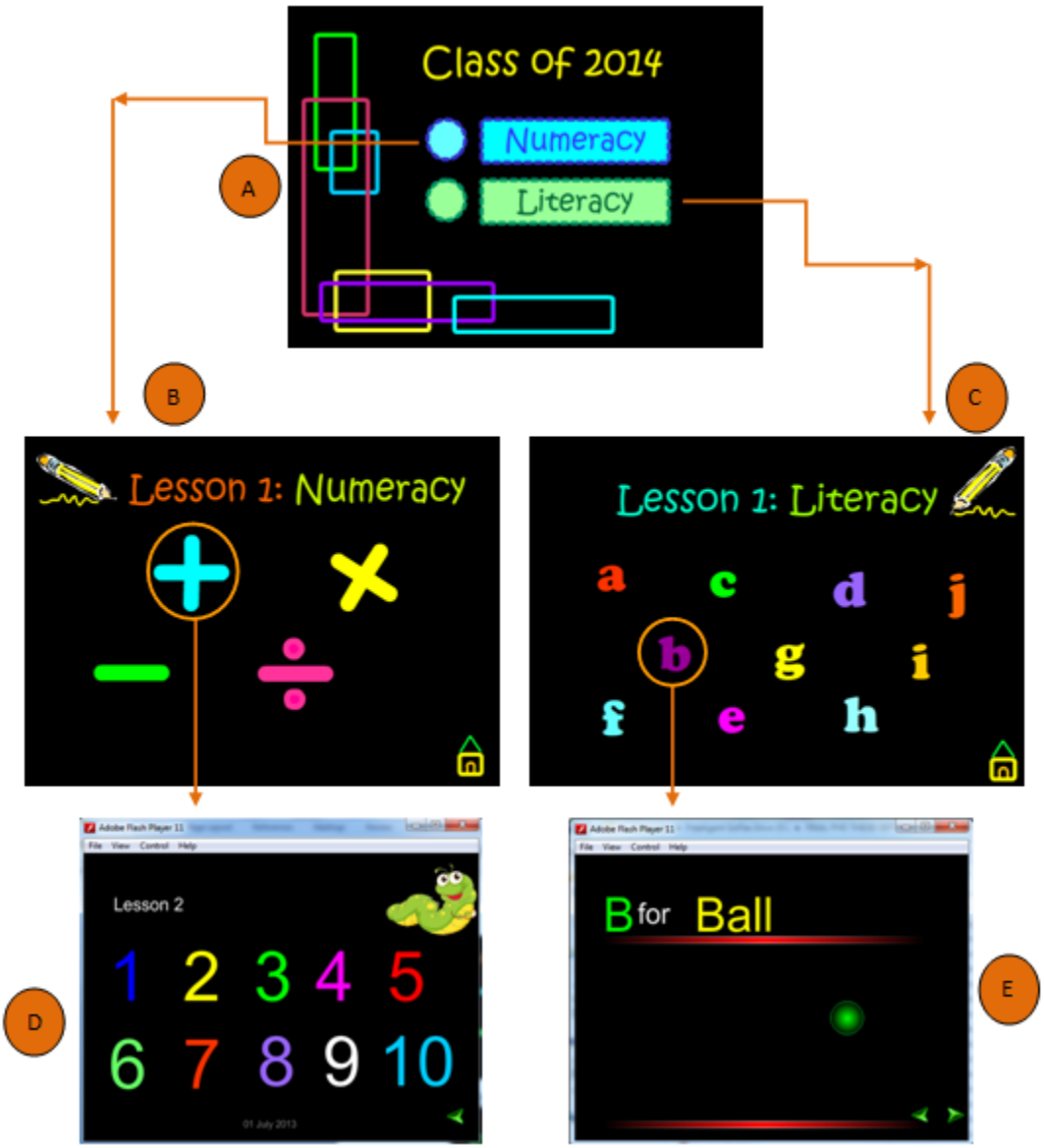
University of Nottingham, Jubilee Campus

Wollaton Road, Nottingham,

NG8 1BB

APPENDIX C

Example of the screen design: Screen Shot on the Lesson Module


Class of 2014
Numeracy
Literacy
A
B
C
Lesson 1: Numeracy
Lesson 1: Literacy
a c d j
b g i
f e h
Lesson 2
1 2 3 4 5
6 7 8 9 10
B for Ball
D
E

A

Main screen - There are two choices of subject either Numeracy or Literacy to be clicked by the user.

B

Numeracy Screen - The screen appeared after user click on a Numeracy button from the Main screen.

C

Literacy Screen - The screen appeared after user click on a Literacy button from the Main screen.

D

Sub-screen after user click on a Addition symbol. Each number clicked will navigate to the correspondent screen.

E

Sub-screen after user click on a 'b' button. The screen displays the animated bouncing ball that represent the letter 'B'.

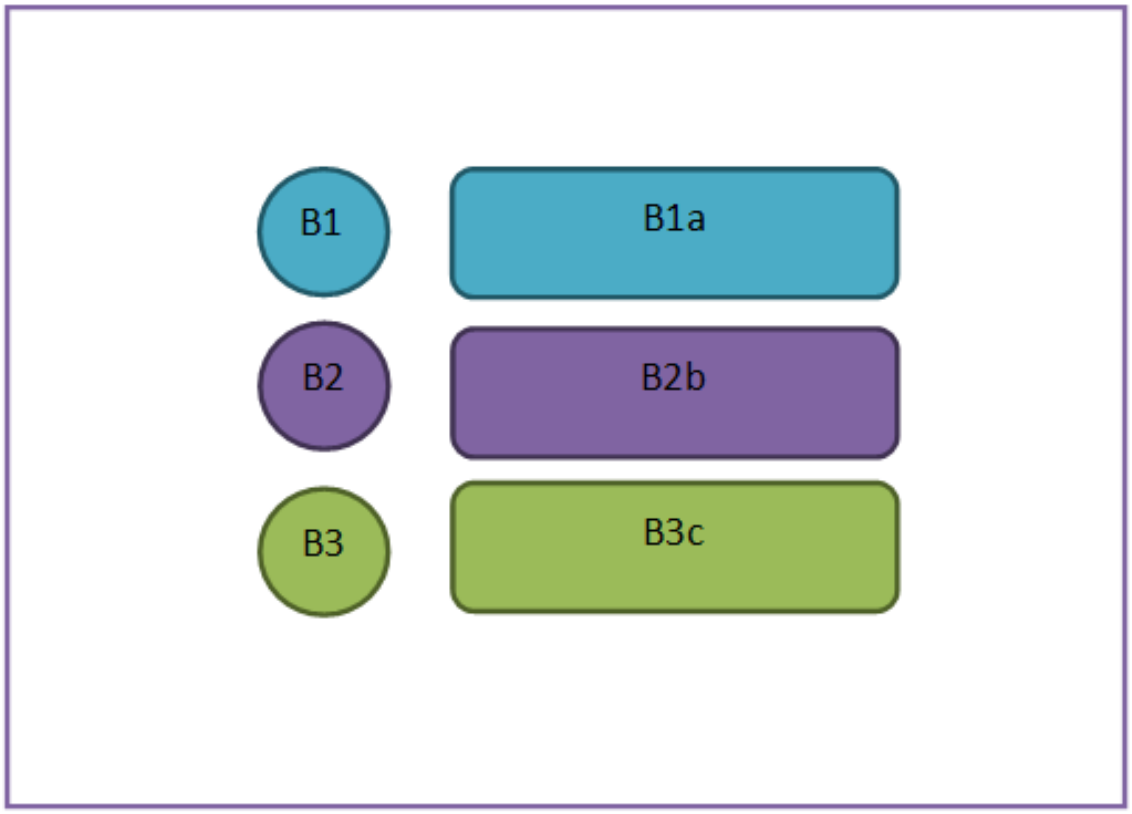
APPENDIX D

STUDY 2 LESSON TASKS

PhD. RESEARCH

Storyboard: Develop a module

Date : 01 March 2014



*Screen design on the **main** page.*



Navigation BUTTONS to the correspondent page.



Navigate to START new module on flash page.

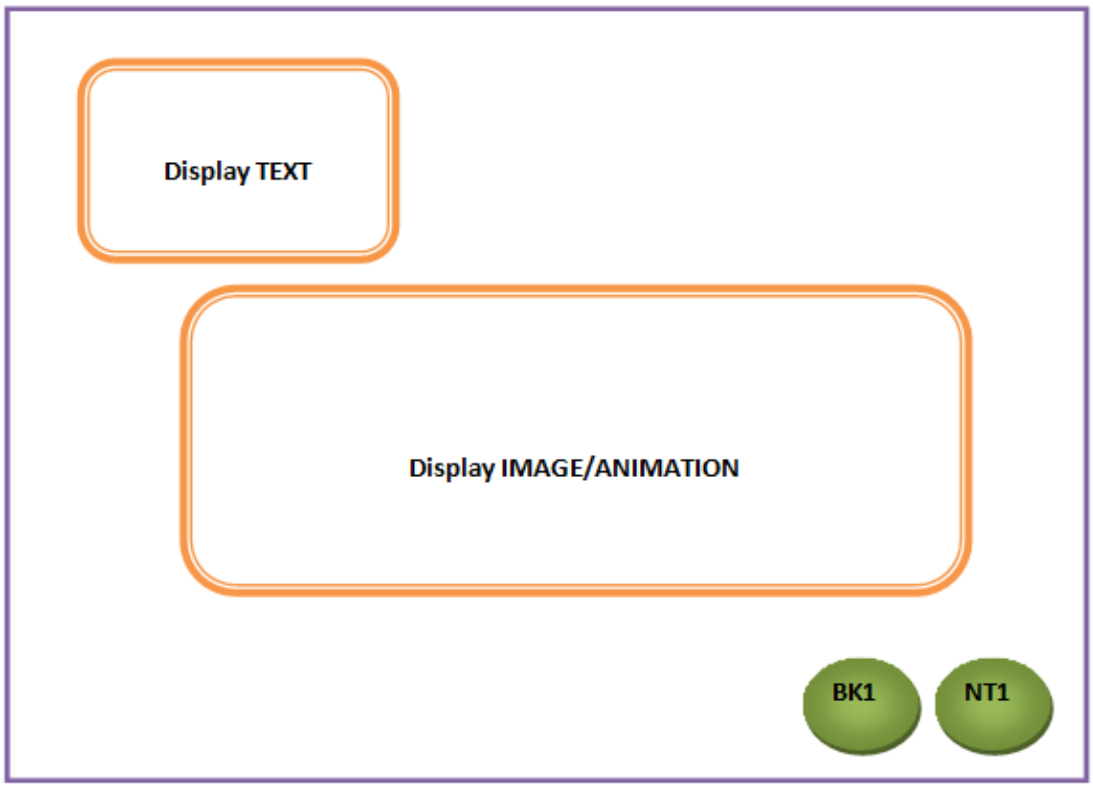


Navigate to DRAW object page.



Navigate to ANIMATION page.

If B was clicked:



BK1 Navigate to *previous* page.

NT1 Navigate to *previous* page

APPENDIX D

TUTORIAL LESSON FOR STUDY 2 EXPERIMENT

Pictionary Lesson for Beginners

You are required to Develop a Pictionary or any related module for kids. The module should comprise multimedia elements such as colorful text, graphics animation and audio (optional). You will be guided by a tutor in order to produce a task. The example as below:

Step 1: Create a NEW FILE.

Step 2: Create SCENES - at least 5 scenes.

Step 3: Rename all the scenes name into APPROPRIATE names such as Main Scene, Scene Intro etc.

Step 4: Create TEXT - create text on every scene and edit those text accordingly.

Step 5: Modify Text - learn to change color or font size.

Step 6: Draw an image. Any Image you like using a provided tool or IMPORT an image from internal or external file.

Step 6: Draw image(s) on every scenes.

Step 7: At the first scene, create a NAVIGATION button. NEXT button and BACK button these buttons will be reuse for the whole module BUT the changes will be affected only on the action Scripts).

Step 8: Start to control the scene. Understanding of concept STOP function; LAYER function etc.

Creating Motion Tween in Flash

1. Draw a vector using any of the drawing tools in Flash, say a small circle and convert it into a symbol (F8) by selecting the Graphic option and name the symbol say 'ball'.

2. Click the 20th frame in the Timeline and insert a frame (F5).

3. Now right-click the 20th frame in the Timeline and select Create Motion Tween (or Motion option from the Tween panel of the Properties inspector) and insert a Key Frame (F6).

4. Select the 10th frame and insert a Key frame (F6) and move the ball to a different position say, above the current position to create a motion sequence (automatically tweened by Flash).

5. Save your work and test the Movie (Ctrl + Enter). That's it you have created simple animation using Motion Tween.

Stop Action

Script to stop the action: Stop();

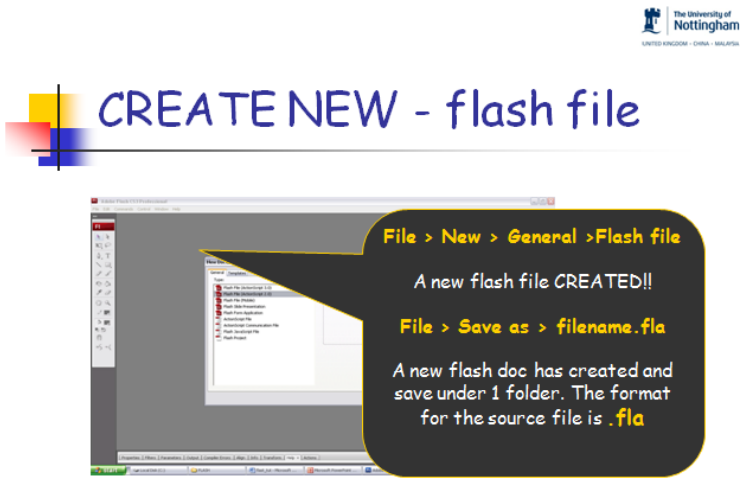
Navigation scripts

```
On (Release) {
    gotoAndPlay ("frameName", frameNumber);
}
```

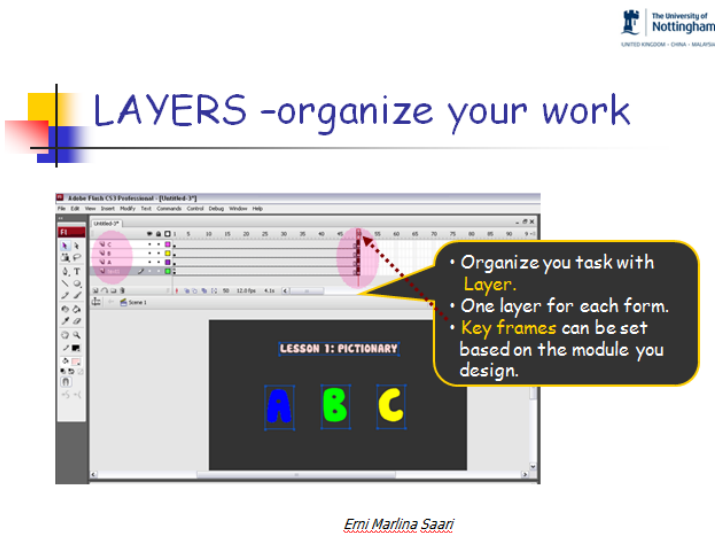
APPENDIX E

Lecture on Flash Action Scripts

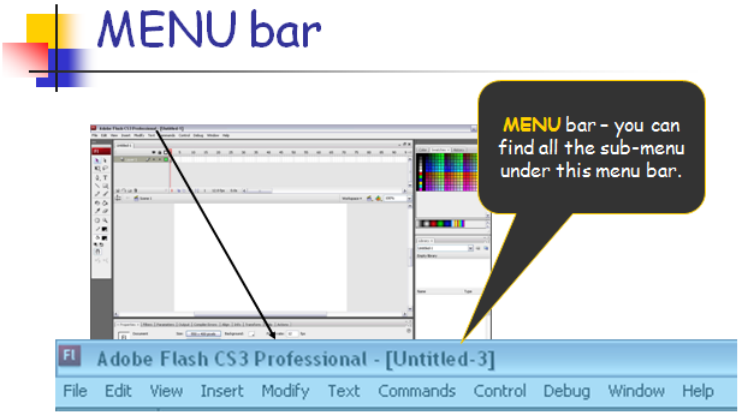
These are all steps involved during Study 2 experiment whereby tutor will start to teach respondents using conventional (traditional) approach. Appendix 1E has details about the steps involved.



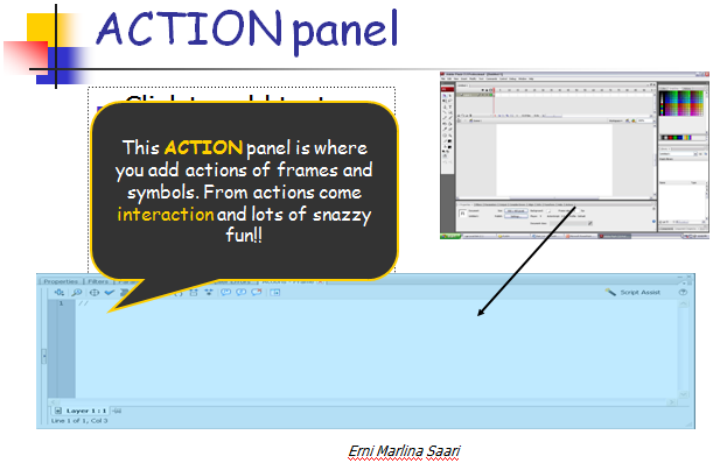
New Screen



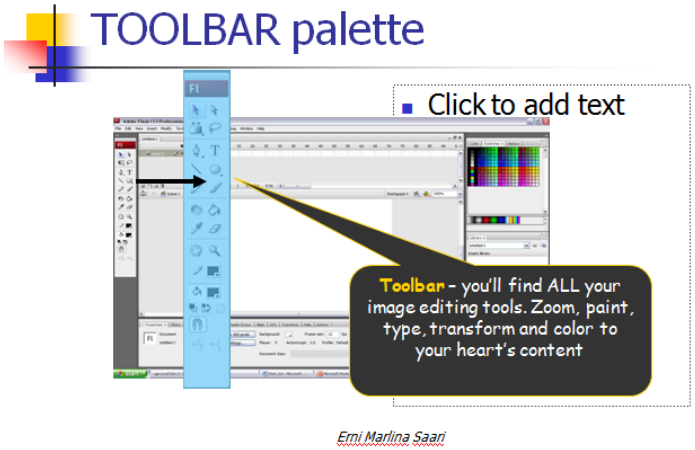
Layer Screen



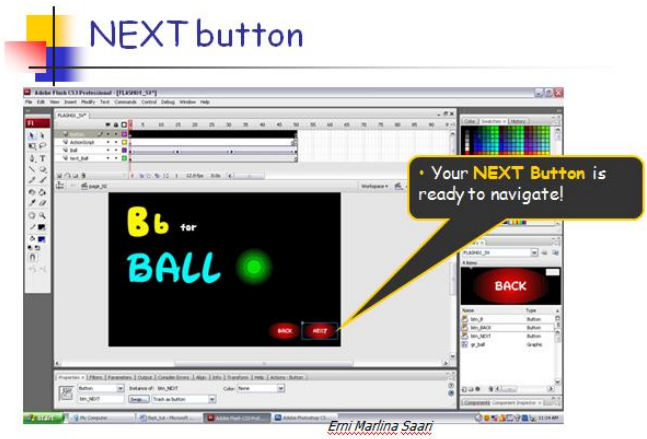
Menu bar screen



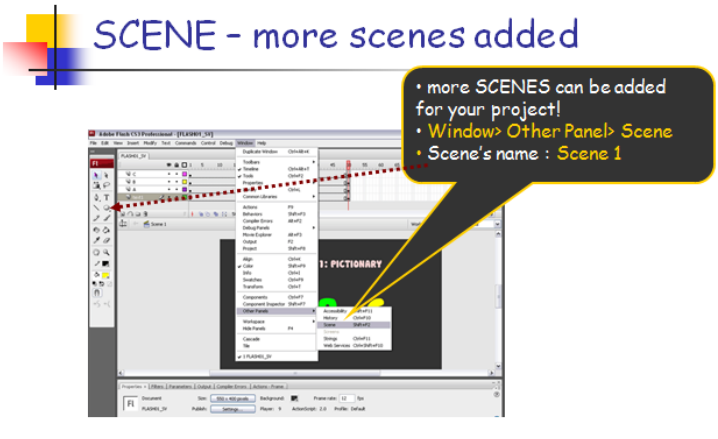
Action panel screen



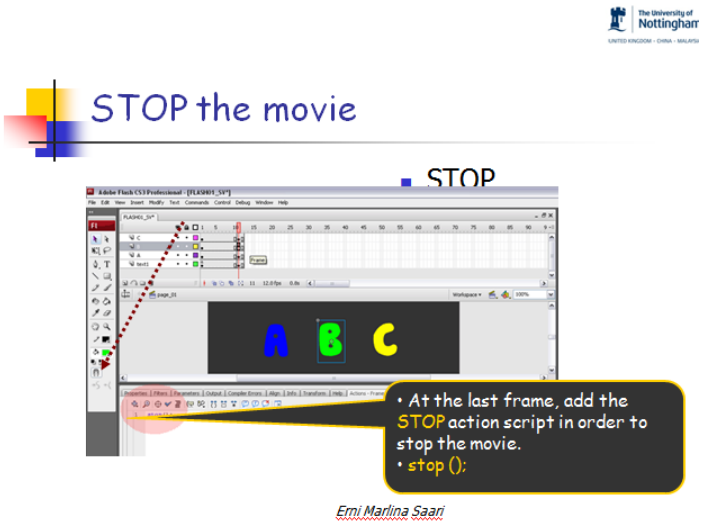
Toolbar palette screemn



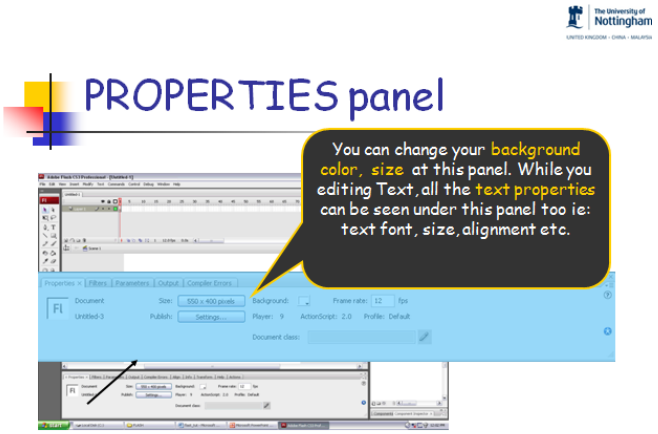
Next button screen



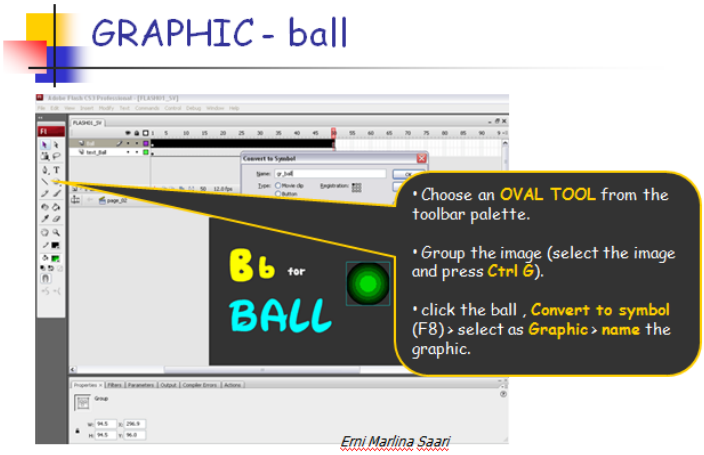
Addition scenes



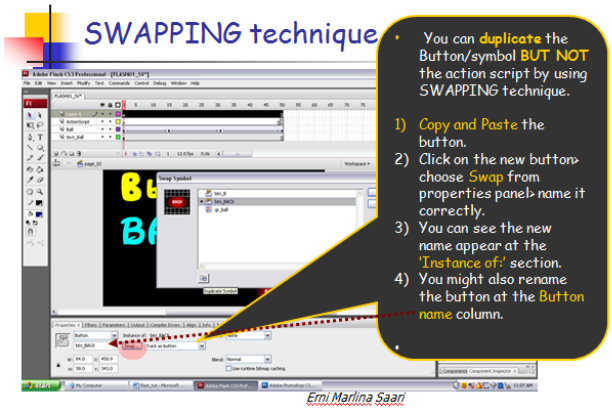
Stop movie screen



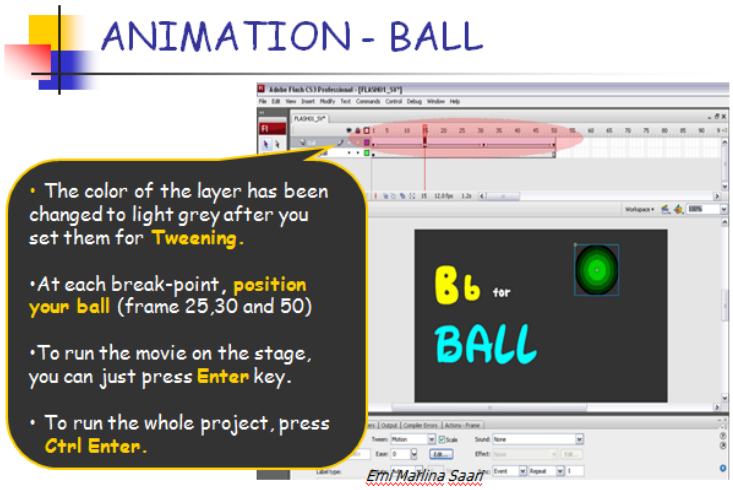
Properties panel screen



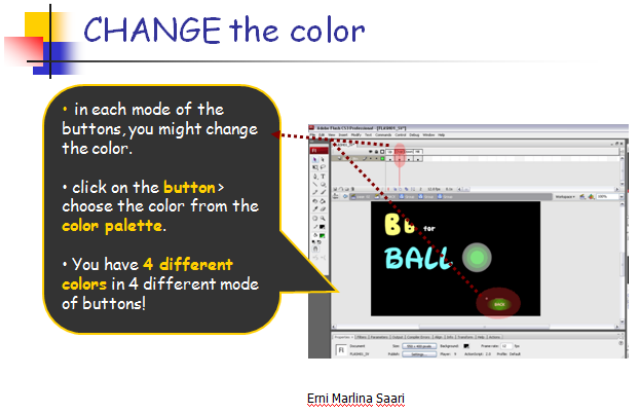
Properties screen



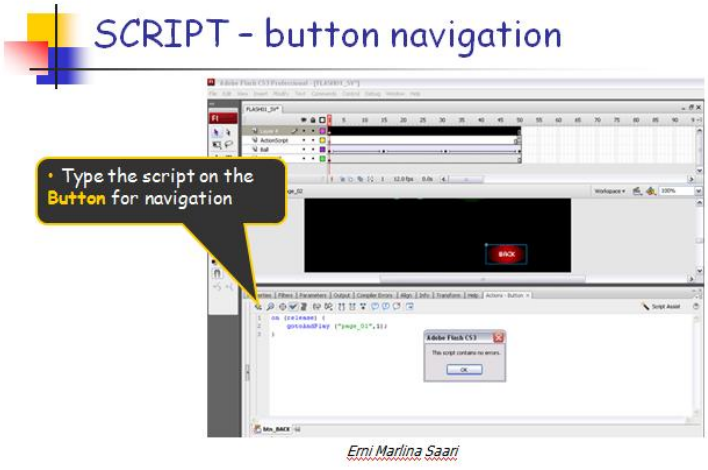
Swapping technique screen



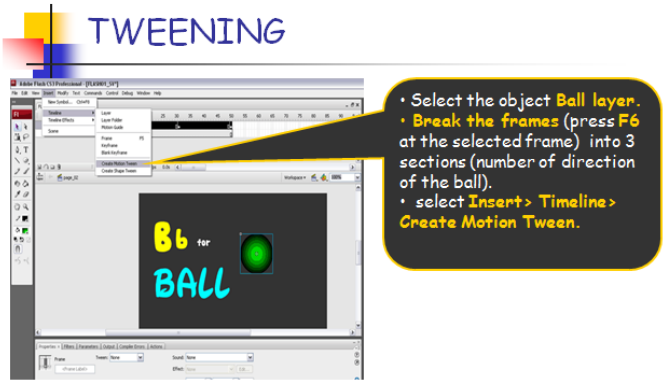
Animation screen



How to change color - screen

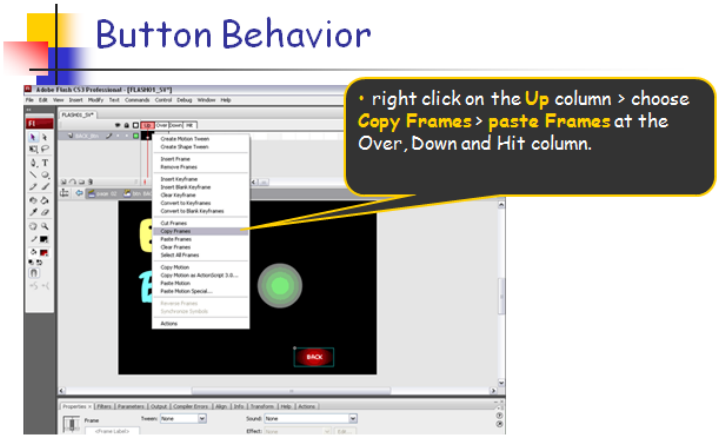


Stage to type the script - screen



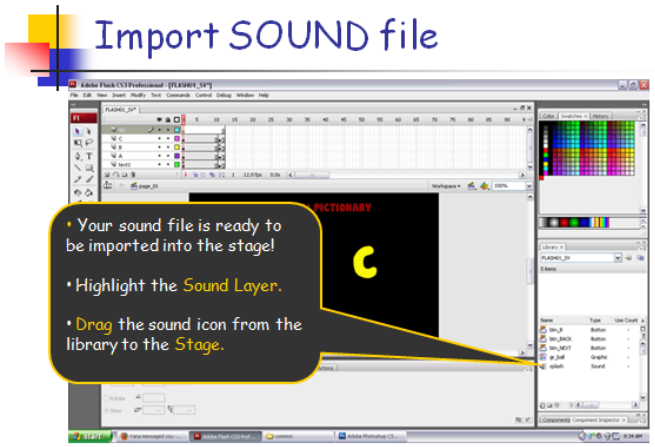
Erni Marlina Saari

Tweening tutorial screen

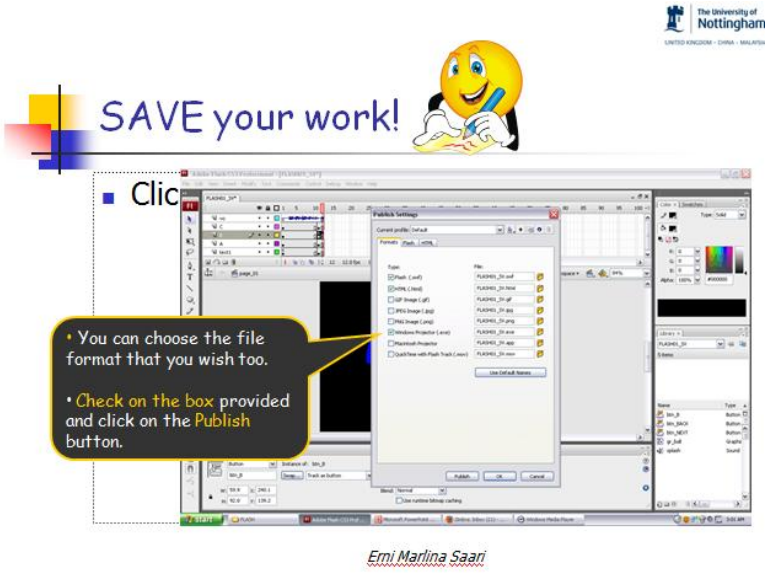


Erni Marlina Saari

How to insert behavior to each button.



How to import sound



How to save file

APPENDIX F

MINISTRY OF EDUCATION MALAYSIA

KEMENTERIAN PENDIDIKAN MALAYSIA
(MINISTRY OF EDUCATION MALAYSIA)
Bahagian Kokurikulum dan Kesenian
Division of Co-Curricular and Arts
Aras 3, Blok E9, Kompleks E
Pusat Pentadbiran Kerajaan Persekutuan
62604 PUTRAJAYA
MALAYSIA

KEMENTERIAN PENDIDIKAN MALAYSIA

Tel. : + 603-8872 1610
Faks (Fax) : + 603-8872 1608
Portal : www.moe.gov.my
(Website)

Rujukan Kami : KPM.600-14/1/20 Jld. 12 (22)
Tarikh : 14 Februari 2017
17 Jamadilawal 1438 H

Pengarah Urusan
Ardent Educational Consultants Sdn. Bhd. (791370-A)
100-1, Jalan 2/23A,
Taman Danau Kota,
53300 Kuala Lumpur

Tuan,

PENGANJURAN PERTANDINGAN *BEAVER COMPUTATIONAL THINKING COMPETITION* DI SEKOLAH-SEKOLAH RENDAH DAN MENENGAH MALAYSIA.

Dengan hormatnya perkara di atas dirujuk.

2. Sukacita dimaklumkan bahawa Bahagian Kokurikulum dan Kesenian (BKK) Kementerian Pendidikan Malaysia (KPM) telah meneliti permohonan **Penganjuran Pertandingan *Beaver Computational Thinking Competition* di Sekolah-sekolah Rendah dan Menengah Malaysia.** KPM **tiada halangan** untuk pihak tuan menganjurkan pertandingan berkenaan **secara terbuka** kepada murid-murid sekolah dan bersedia bekerjasama untuk kelancaran program tersebut.

3. Walau bagaimanapun, penganjuran pertandingan tersebut adalah tertakluk di bawah **Surat Pekeliling Ikhtisas Bil. 9/2004 – Ruj. KP(BS)8691/Jld.XVIII/(9) bertarikh 16 September 2004** – Panduan Penglibatan Murid-Murid Sekolah Dalam Program Atau Aktiviti Anjuran Agensi-Agensi Selain Daripada Agensi Di Bawah Kementerian Pelajaran Malaysia.

4. Sehubungan itu, pihak tuan hendaklah mengambil maklum dan mematuhi peraturan-peraturan berikut:

4.1 program yang dilaksanakan hendaklah tidak bertentangan dengan semua dasar kerajaan dan dasar KPM;
4.2 kebenaran daripada Pengarah JPN dan Pengetua/ Guru Besar hendaklah diperoleh terlebih dahulu sebelum mengadakan sebarang program di sekolah atau yang melibatkan penyertaan guru dan murid;
4.3 penyertaan murid dan guru hendaklah tidak mengganggu atau menjejaskan proses pengajaran dan pembelajaran mereka;

4.4 setiap murid perlu mendapatkan kebenaran ibu bapa melalui Pengetua dan Guru Besar masing-masing;
4.5 pihak penganjur diminta mengadakan langkah berjaga-jaga demi keselamatan dan kebajikan murid-murid sewaktu program berlangsung;
4.6 tempat yang dipilih untuk mengadakan program telah dikenalpasti selamat dan bersesuaian dengan program
4.7 guru-guru pengiring hendaklah dilantik oleh pihak sekolah mengiringi murid-murid ke program atau aktiviti dan
4.8 tiada implikasi kewangan kepada KPM, JPN dan PPD yang terlibat dengan pelaksanaan program yang dijalankan.

5. Pihak tuan juga adalah diminta untuk mengemukakan laporan program kepada Bahagian ini dua minggu setelah program selesai untuk tujuan rekod dan rujukan. Kelulusan ini hanya untuk tujuan yang dipohon sahaja dan luput selepas 31 Disember 2017.

Sekian.

'SEHATI SEJIWA'
'BERKHIDMAT UNTUK NEGARA'
'PERPADUAN MELALUI KOKURIKULUM'

Saya yang menurut perintah,

(ROZAINUM AHMAD)
Pengarah
Bahagian Kokurikulum dan Kesenian
b.p. Ketua Setiausaha
Kementerian Pendidikan Malaysia.

s.k. 1. Timbalan Ketua Pengarah Pelajaran Malaysia (Sektor Operasi Pendidikan)
2. Pengarah Jabatan-jabatan Pendidikan Negeri

APPENDIX G

Publication: Proceedings in CSEDU 2015

Learning Computational Thinking Through the Use of Flash Action Scripts

Preparing Trainee Elementary School Teachers for Teaching Computer Programming

Erni Marlina Saari, Peter Blanchfield and Gail Hopkins
School of Computer Science, University of Nottingham, Jubile Campus, Nottingham, U.K.

Keywords: Computational Thinking, Flash Action Scripts, Trainee Teachers.

Abstract: The teaching of Computer programming is now mandated in UK state run primary and secondary schools but few elementary teachers have any exposure to programming and are generally from backgrounds that have not given them fluency in using such technology. This paper outlines an attempt to develop a training scheme for trainee teachers that will introduce them to computational thinking through the use of the Flash Action Script Development environment. It is believed that learning how to use this tool will provide them with greater motivation to learn how to program as the scripts will be used to develop teaching instruments that they might use in their classroom. The current paper reports progress on the development of a tool that will allow the teaching of Flash Action Scripts and the testing of this approach on groups of trainee and experienced teachers. It reports on an experiment and the relative level of enthusiasm, motivation and attitude of the experienced teachers and the trainee teachers both before and after exposure to the pilot tool. As expected the trainee teachers had a more positive attitude towards the potential of the learning tool but both groups had improved attitude and enthusiasm after the experiment.

1 INTRODUCTION

The Computing and Schools (CAS) initiative in the UK was launched in 2008. It was in response to concerns that previous moves towards teaching Information & Communication Technology (ICT) in schools has led to misleading pupils as to what constitutes Computing and demotivation of pupils who already have Computing skills (Crick and Sentence, 2011). This led in the autumn term of 2013 to a program of curriculum changes to increase the degree of computer programming taught at secondary schools. This was also influenced by a response to leading industrialists asking that schools should be teaching the basics of programming to a wider audience. At the start of the 2014 autumn term this process was extended to cover elementary (primary) schools. This second process is likely to be even more difficult than the first as the nature of the elementary school teacher training programmes is not to produce teachers who are specialists; rather it necessarily attracts and develops generalists. The generalist nature of teaching in elementary schools has already meant that earlier programmes to promote maths and English teaching in elementary schools has required retraining programmes for teachers in those schools and further development of subject specialist teachers in those schools and school districts (Williams, 2008).

The CAS initiative has been built around certain facilitating universities with specialist teacher training or retraining programmes. For example Nottingham Trent University offers an MSc degree titled Computing in Education that includes modules that help teachers to learn how to teach programming skills. Regular meetings of the CAS allow experienced teachers in different areas to share their findings on how to run programming classes in schools. Different approaches have been taken but the focus must now be more on training the elementary teachers to be able to handle the material as they are seldom from a technical background and generally have less natural inclination to program. Many elementary teachers are in fact quite nervous of the whole area of technology. (See for example teacher comments in Section 4.1.)

Saari E., Blanchfield P. and Hopkins G.,
Learning Computational Thinking Through the Use of Flash Action Scripts - Preparing Trainee Elementary School Teachers for Teaching Computer Programming.
DOI: 10.5220/0005442600750084
In *Proceedings of the 7th International Conference on Computer Supported Education* (CSEDU-2015), pages 75-84
ISBN: 978-989-758-108-3
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

2 THE ROLE OF COMPUTATIONAL THINKING IN LEARNING TO TEACH PROGRAMMING

Much work is currently being undertaken to help learners overcome the inability to program. Of particular value is work being undertaken with young learners themselves. Scratch (Franklin et al, 2013) and Python (Begosso et al 2012, Bogdanchikov, Zhaparov & Suliyev, 2013) have been popular for these studies. However, based on our experience with many elementary teachers, using a visual system like Scratch is off-putting due to its child-centred look and feel, and teaching a programming language like Python can seem daunting due to its considerably increased complexity over Scratch. Either way, programming languages seem intrinsically hard to learn and most of the examples of using them are of little intrinsic value to teachers. The teachers can see that children might be drawn to programming games and animations but feel a level of fear when approaching the idea of learning programming themselves. We illustrate this further in Section 4.1. below.

Much of the work of the CAS has been to emphasise the use of "Computational Thinking" (CT) as a valuable thinking skill for all aspects of life (Wing, 2006, 2008, Grover and Pea, 2013). CT involves using the aspects of developing computer solutions to problems, algorithmic thinking, decomposition, abstraction, generalisation and evaluation (Dorling 2014). Wing (2008) has stressed that CT can be adopted in everyday life but it is doubtful that many people would consciously apply many aspects of this thinking when approaching most real life problems. In everyday life it is probably more appropriate to apply aspects of CT such as evaluating, looking for generalisations and use of decomposition. For example, a satnav will work out computationally the best route to take according to the algorithm it has been programmed to use but the user needs to evaluate the solutions and understand what is being prioritised in order to know whether to take the advice being given. Generalisation is used when we try to see how previous solutions might easily be adapted to new problems. Decomposition is used when we look at how to break down problems into parts that can be tackled more easily than the whole. For some groups the development of algorithms may be more valuable – for example those learning to program. In that regard we must now include elementary school teachers in this group as they will need to have some appreciation of how algorithms are designed and implemented in order to lead the pupils towards this understanding. To some extent this is going to be limited by how much abstraction an elementary school pupil is able to understand. Languages that are being used to teach programming in elementary school usually try to make the algorithms as concrete as possible, by using graphics and games design.

The motivation for the research presented in the current paper was our prior experience in trying to teach trainee teachers how to use ICT tools at a teacher training institute in Malaysia. Problems encountered in practice were largely due to a lack of computational thinking in the process applied. The students had a tendency to "copy and paste" examples which they felt were similar to the task they had to do rather than analyse the task, break it down into sub tasks, understand the set of steps needed to achieve the sub task and so on. This lack of CT may have been largely due to their unfamiliarity with building such processes. However, it leads us to pose the question: "how can we encourage primary school teachers to learn to think computationally and thereby be able to learn the languages they need to deal with to teach young children to program also?" Coming from a non-technical (engineering/physical science) background can make elementary school teachers fearful of using technology in general and view learning to program as a difficult task. One aspect of this, from previous observations and discussions, is their lack of intrinsic motivation to learn programming. A first step in answering the question of how to encourage them to think computationally will therefore be how can we provide intrinsic motivation to think computationally? The position taken in this paper is that using an approach where they can see that they could gain facility in using a tool that will enhance their ability to produce really attractive teaching aids will enhance their motivation to learn to think computationally if computational thinking is intrinsic to using that tool. Lin (2012) highlights that educators can become courseware designers and no longer just play the role of submissive users but instead become creative developers. For this reason this research has chosen to look at the ways in which teachers can learn to produce Flash Action Scripts to develop attractive teaching aids. It is believed that this will enhance their motivation to learn to think computationally and we move on to discuss this further in the next section.

3 TRAINING OF TEACHERS IN THE USE OF EDUCATIONAL TECHNOLOGY USING FLASH ACTION SCRIPT

One of the challenges identified for a teacher education program is engagement of pre-service teachers and teacher educators in conversations about their attitudes regarding the role technology should play in teaching and learning. Hammond (2013) mentions the significance of ICT even if engagement is not to be taken for granted and the effect of the learning is complicated. Others have sought to emphasise the need to integrate technology into education. For this to be done successfully it needs to be integrated into the training of the teachers. The arguments around this have been summarised in Tondeur et al (2012).

It is our aim that we can develop an electronic learning tool specifically aimed at helping trainee teachers in which ability to use Flash Action Scripts and the Flash development environment are the learning objectives. Utilisation of such an environment builds upon the views expressed by others of the value of using instructional video in e-learning (Zhang et al, 2006, Wieling & Hofman 2010).

The interface that is provided for the Flash Action Script developer is quite complex but has a lot in common with other IT tools that users will be familiar with – for example Photoshop and storytelling software like Microsoft Photo Story. Skills learned will also be generalizable to other animation development tools such as Anime Studio, Blender and even the Unity 3D game engine. The scripting language itself is similar in form to many other programming languages but the purpose of the scripts – such as actions performed when buttons are clicked – has a much simpler to understand connection to the purpose that is being looked at.

During interviews many of the teachers and trainees met during these studies indicated that they were familiar with using storyboarding software, for example, to produce artefacts to aid their teaching. The actual level of scripting needed to make simple animations within a Flash program is low. However as the complexity of the artefacts increases so also does the complexity of the scripting needed. This therefore potentially leads to a staged introduction to the scripting and at the same time help to develop the aspect of algorithm design within computational thinking. It can also introduce the concept of generalisation as the same basic format will be needed in many different problems. As the user progresses aspects of abstraction will also be possibly developed, for example through the use of functions to undertake often needed processes.

The ultimate aim of our learning environment is to make this a self-help tool that will analyse a user's performance in lesson exercises and that any lesson will be automatically adapted to a given user's needs. In order to investigate this a pilot tool was developed in which a set of lessons was produced aimed at teaching the use of the Flash tool to qualified school teachers and to a set of trainee teachers on a PGCE course. The lessons took the learners through the stages of building an artefact that they were expected to see as useful for their intended teaching. The progressive development of the artefact focused on providing a set of increasingly complex animations aimed at helping children in early years' education to obtain basic literacy and numeracy skills. The teachers and trainee teachers were taken individually through a set of their own lessons to build the artefact. During this process faulty or non-computational thinking approaches adopted by the teachers/trainees were identified. Any incorrect approaches were then corrected by the trainer and the success or otherwise of the strategy taken in attempting to correct them was examined.

4 EXPERIMENTAL METHOD

In order to investigate the difficulties observed in the Malaysian trainee teachers and the role of our proposed tool in teaching CT, two groups of Malaysian test subjects were recruited for a pilot study. The first was a group of experienced teachers. The second was a similar set to those in the original Malaysian trainee classes, except that they were studying on a PGCE course in the UK. Both groups had no background experience in the use of software development.

The study was subject to the ethics approval process. All participants were informed of the nature and purpose of the study before partaking and were aware of the recording methods that were to be used. They all signed consent forms for their data to be gathered and used in the study.

Prior to using our pilot tool each subject was interviewed initially about their attitude to using the Flash environment and other aspects of using technology, particularly the learning of programming tools including the use of Scratch. The purpose of this was to identify attitudes already held

by the subjects and enable us to identify in post-session interviews whether their attitudes had changed as it was expected that encounter with the actual process would improve their attitude. They then individually took part in a staged set of training lessons using the tool. The lessons were presented one-on-one but otherwise in a way similar to the original group (lecture, PowerPoint presentation and subsequent practical exercise). Each subject's comments and progress were recorded during their performance. The data captured included length of time to complete actions, evidence of enthusiasm for the task and number and types of errors made. After the lesson the subjects were once again interviewed about their attitudes towards use of Flash tools. The recordings were transcribed for analysis. A set of thematic codes was developed to use in determining the level of motivation and engagement with the activity. The codes reflected the following positive attitudes:

- I - Interest
- E - Engagement
- M - Motivation
- C - Confidence
- En - Enjoyment
- A - Affordance

And the following negatives:

- B - Boredom
- L - Lack of Motivation
- Co - Confusion
- F - Fear
- D - Difficulty

The transcriptions were then coded to identify statements and actions that indicated successful and failed understanding of the task and enthusiasm for the process (such as their attitude towards the Flash tool). Examples of positive and negative statements for each of the codes are given in Table 3 in Section 5.4 below. The time taken to develop a level of engagement was also recorded as a separate measure of participant engagement/motivation.

These lessons used in the pilot study are described in more detail in the next section.

3.1 Format of the Lessons

In the pilot study each participant worked through the sequence of lessons which aimed to start the development of the teaching/learning asset. As the lessons progressed the participants attempted to produce the asset. Their performance was monitored in order to determine causes of any failure to produce the expected outcome. These were then classified to determine if they were due to a lack of application of computational thinking. An example of non-computational thinking was the previously described use of a copying and pasting approach rather than understanding the development algorithm and following the algorithm for a new step in the process. As the stages of the lesson developed so did the complexity of the scripting needed to produce the correct performance.

In the lesson being followed an artefact was developed to create a teaching/learning aid for Elementary Level learning in Literacy and Numeracy. The lesson content is a subset of foundation level lessons in reading and number concepts. The artefact being produced was designed to meet expectations of early years learners for an attractive interface with colourful illustration and animations designed to gain attention. In the various stages the test subjects were given lessons in producing elements for the artefact that were successively more difficult to produce. The first step was to produce a simple menu – see Figure 1. This task was also broken down into stages, such as producing the title text, and then adding simple graphic elements, such as the button shapes. The next stage was producing the action for the buttons. The process of breaking tasks down into a set of simpler stages is a fundamental part of CT so in addition to the learning of the development tool users also got the opportunity to see the value of the CT approach in solving tasks.

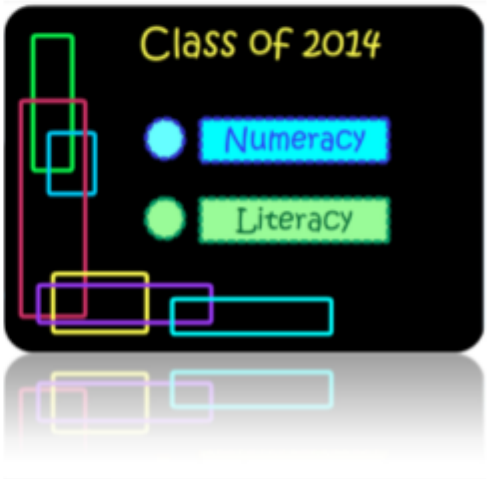


Figure 1: Original page.

This first stage of the lesson leads the learner through the process of producing the first screen to

the point of producing an action to display the next screen. So, for example, a second screen is designed to appear when the Literacy button is pressed. This will display a view as shown in Figure 2 below.



Figure 2: Literacy screen.

Each of the letters in the screen shown in Figure 2 will be turned into a button that leads to an action. The learner has to add the required components and thus practice the process from the earlier learning activity. (The letters and heading are all originally added as text.) The symbol requires the adding of a picture and the action script required is like that of the original button but the appropriate action had to be added to the letters which themselves had first to be converted into "buttons". While the next stage can be more or less derived from the previous the level of "scripting" required increases as the action now becomes more complex leading to the need to create an animated scene after an intermediate button press (see Figure 3). However, the repeated process has a lot of commonality so it should be possible for the participants to begin to understand the value of generalisation (one of the CT skills) and apply this by adding function calls to the Action Scripts to allow common code to be reused. This would not be done in early exercises as the needed instruction for using such functions is not given early in the set of tasks. Figure 3 also shows an additional feature that is taught at this point. The green path shown in the figure is added as the movement route (Tweening in Flash Action Script terminology) for the apple to follow during the animation that will be prescribed in the script.



Figure 3: Scene with added animation path.

The next section details the results that were obtained from the pilot study and we then move on to draw our conclusions and suggestions for the next stage of this work in the final section.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

5 RESULTS

The experimental subjects were all volunteers and were not incentivised to take part in the study. They fell into two groups. All the subjects were Malaysians. The subjects included a group of practicing teachers and a group of trainee teachers engaged in pursuing a PGCE qualification. There were significant differences between the qualified and trainee teachers both before and after the sessions and these differences will be explored in future publications.

5.1 Initial Interviews

As mentioned above a set of pre interviews were undertaken in order to allow for comparison between the pre and post session attitudes to programming and the use of Flash Action Scripts in particular. Certain differences also emerged between the qualified and trainee teachers in this session. Of the established teachers the responses to questions regarding the likely usefulness of learning Action Scripting were negative and all were nervous of the need to use a technical tool. A typical response was "*I am not good in using technology especially programming. I am afraid of using technology actually. I don't like to use computers ...*"

The PGCE students were less reticent to take part. They all felt it would be good to learn to use

Action Scripts as it would enable them to use the tool to make their own teaching/learning instruments. However, they generally did not feel it would be easy to learn as it appeared to involve a similar process to learning more formal programming systems like C++. A typical quote from this group was "*I want to learn this programming language and I will come out with my own teaching module soon!*" There were some negative attitudes, for example one subject asked "*Why would I need to learn Flash or have to design a learning tool as all of my lesson material will be provided by the school?*"

5.2 Performance in the Practical Sessions

During the practical sessions the volunteers were required to produce codes that performed the set of functions shown in Figures 1 to 3. The performance was monitored and the actions were recorded using screen capture so they could be analysed afterwards. The major problem was the tendency to copy and paste previous examples rather than to analyse the problem, understand the components that were needed and develop the appropriate set of steps to produce the artefact. When they found their actions had not produced the correct behaviour they were asked to analyse why this had happened. They were taken back to previous steps and the lesson was repeated until they were able to produce the correct output. Different processes in producing a good learning artefact were identified against the Computational Thinking approach. A good example is the idea of abstraction as embodied in the concept of a layer. In Flash Action Scripting the layer is a basic abstract component that gets non-abstract components – actual text, pictures, and scripts – attached to it. As a result a user thinking computationally would be expected first to engage with the abstraction – by invoking a new layer for a new action – then to populate it. Thinking non-computationally exhibited itself in various different ways. One typical example was when users wanted to remove an existing layer. Instead of selecting the layer and deleting it the users who were not thinking about the abstraction would delete the contents of the layer rather than the layer. This would lead to an error when the code was compiled and an error message that was not understood by the user. Such error statements alert the teacher with experience to the lack of correct thinking. The user is then directed to the part of the lesson on layer creation.

The PGCE students all grew in confidence and enthusiasm for the process of learning Flash Action Scripting more quickly than the experienced teachers, however, all participants were able to finish the first task correctly. Enthusiasm for and motivation to use the tool were indicated by comments made by the experimental subjects. The trainee teachers generally were more enthusiastic to start with and thus were more motivated to understand and correct their mistakes.

The average time to complete the initial task was 35 minutes. The longest time taken for the process was one hour and this was from a subject who had shown the lowest motivation for the task as he intended to return to administration within a school and not teaching.

5.3 Post-session Interviews

After the first practical session the subjects were all re-interviewed. They were asked questions about their attitude to learning Flash Action Scripting having now been exposed to it. Questions centred on their views regarding the potential usefulness of learning to use Flash Action Scripting, technology in general, their motivation and ability to learn Flash Action Scripting and their understanding of the steps used in computational thinking.

Most of the experienced teachers felt an improved attitude towards the usefulness of the process. The teacher who intended to return as an administrator had not improved his attitude at all. One of the others still did not see that it could be useful to them in their teaching or to teaching in general. The other five were enthusiastic about the possibilities. Of the PGCE students all were convinced not only of the value of the tool but also of their ability to learn to use it.

When asked about the use of technology in general to support their teaching the experienced teachers had generally gained an improved attitude.

Most of the experienced teachers did not feel intrinsically motivated to learn programming but had found the process of the lesson enjoyable and felt they had been successful in learning how to use the tool. Eight of the ten trainee teachers expressed the idea that they found the learning intrinsically motivating as they felt they would benefit in their own careers. These have all moved on to the further study of the subsequent lessons and have been willing to try to learn further by themselves!

The CT skills needed in the task to perform it efficiently were those of Algorithmic thinking and Abstraction. Eight of the trainees exhibited

Algorithmic thinking by the end of the first lesson and seven had demonstrated use of Abstraction. However, only four of the experienced teachers had developed this CT approach. For example in the use of layers – linked to the concept of abstraction – seven of the trainee teachers and one of the experienced teachers had understood how to properly use layers.

5.4 Thematic Analysis

As was explained in Section 4 above the recordings of the interviews during the practical session were all transcribed and the transcriptions were then thematically coded looking for different indicators explained in section 4 relating to positive and negative attitudes. Table 1 below shows themes that emerged prior to the experiment taking place.

Table 1: Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **before** taking part in the experiment.

Codes	TT Frequency	ET Freq.	Total
Interest (I)	23	17	50
Engagement (E)	24	15	39
Motivation (M)	27	15	42
Confidence (C)	8	0	8
Enjoyment (En)	12	11	23
Affordance (A)	2	3	5
Boredom (B)	3	11	14
Lack of Motivation (L)	6	15	21
Confusion (Co)	27	32	59
Fear (F)	15	12	27
Difficulty (D)	32	26	58

As can be seen from the results in Table 1 the trainee teachers demonstrated higher levels of positive attitudes to the potential value of the training they were about to undergo, when compared with the experienced teachers (indicated by a total of 96 coded positive statements by the trainee teacher group and only 61 by the experienced teachers). However both groups had negative attitudes in some of their statements and in particular felt the process of learning Flash Action Scripting would be difficult (roughly equal numbers of negative comments on difficulty) and that they found what they were being asked to do potentially confusing (again with similar numbers of negative comments from both groups). None of the experienced teachers expressed any confidence that they would develop the ability to use the tool but eight comments from the trainee teachers expressed such confidence.

Table 2: Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **after** taking part in the experiment.

Codes	TT Frequency	ET Freq.	Total
I	39	19	58
E	29	24	53
M	54	27	81
C	22	20	42
En	22	19	41
A	12	8	20
B	0	2	2
L	0	2	2
Co	13	10	23
F	0	0	0
D	7	3	10

Table 2 demonstrates the frequency of code words in the post-experiment interviews and it can be seen that both the trainee and experienced teachers had significant improvements in their attitudes towards learning Flash Action Scripts and programming technology in general. Of particular note is the fact that no phrases emphasising "Fear" (F) were recorded in the post-experiment responses by either group though some "Confusion" (Co) still remained in both groups (a total of 23 such comments roughly equally divided between the groups) and there was still a residual view that the process was "Difficult" (D) though this was much reduced with a total of ten comments expressing difficulty after the experiment compared to the 58 difficulty comments made before the experiment. Only one of the participants (the experienced teacher now moving into administration) used "Boredom" (B) and "Lack of Motivation" (L) phrases – two responses fitting this description for both categories. Together with the increased number of positive comments on motivation (a total of 81 such comments after the experiment compared to 42 before the experiment) this indicates that those who were expecting to be teachers after their courses all believed the use of Flash Action Scripting to be interesting and motivating. There were few phrases mentioning "Confidence" (C) before the experiment-8 from the trainee teachers and none from the experienced teachers; after the experiment only one of those involved (the administrator) failed to respond with any confidence statement. A total of 42 such comments were recorded from the rest of the subjects.

The time taken to begin to show signs of engagement and enjoyment of the use of Flash was generally lower for the trainee teachers than for the experienced teachers but only one (the

administrator) failed to complete the task and failed to become engaged in the process after 20 minutes.

Examples of quotes reflecting the codes used are listed in the Table 3 below. Some of the phrases quoted show more than one of the coded attitudes. They were coded as representing both attitudes but are only shown as illustrating one. All of the comments listed were given by trainee teachers (respondent TT5 means trainee teacher 5) though similar comments were made by the experienced teachers.

Table 3: examples of coded comments given by the Trainee Teachers (TT4 to TT10).

Code	Quote
I	"As a first time learner like me, this software is easy to learn and the instruction is quite simple..." (respondent TT5)
E	"Very good! I think students will feel more enthusiastic to learn the subject if the teaching instrument used is interesting" (respondent TT4)
M	"I think it is attractive software to learn and teach also. Besides teachers learn how to program, I think we can teach our students to learn how to program as well." (respondent TT10)
C	"I am looking forward to studying more on this and producing my own lesson.." (respondent TT9)
En	"I knew that Flash can do interesting tasks like interesting courseware to gain students' attention." (respondent TT6)
A	"Can you give me a few minutes to try on my own on this software? I would like to produce a small interactive lesson about our lesson just now. I am just so excited to do animation and navigation buttons." (respondent TT4)
B	It is too hard for me to start learn the new software and need to struggle about the technical terms...." (respondent TT5)
L	"... there is just a waste of time to learn new software, I prefer to get the teaching tools that will be provided..." (respondent TT5)
C	make me feel confuse about technical terms and interface too..." (respondent TT5)
F	" I am afraid I could not produce the module at all because I am not good handling the technical tasks.." (respondent TT5)
D	" ..sometimes it very hard to understand especially the complex syntax..." (respondent TT7)

Our results demonstrate that use of the tool increased interest in Flash Action Scripting and enhanced enthusiasm for programming with reduced levels of confusion and frustration. The next section provides some conclusions from this study and proposes the next stage of our work.

6 CONCLUSIONS

The need for elementary school teachers to engage with learning to program and to think computationally has been identified. Pressure in the UK and elsewhere has led to the development of programming as a required part of the curriculum in both elementary and secondary schools. However, elementary school teachers are for the most part ill-equipped to learn to program themselves. It is vital that they also learn to understand the Computational Thinking (CT) process. However, they find that the programming languages they encounter are difficult to learn and lack the ability to motivate them to learn. They are at the same time wary of technology and this also makes them nervous of their potential to learn to program. The current research contends that learning the use of Flash Action Scripting would be viewed as more motivating as a first approach to learning to program for elementary teachers. This is in agreement with the findings of Lin (2012) which sees the need to motivate teachers to become creators of courseware rather than just consumers. Thus the current research aims at producing a software tool for helping teachers to learn to use Flash Action Scripting. It is acknowledged that Flash is not without its issues in terms of compatibility with some systems and security issues. Ultimately the ideas expressed in this paper could be applied to another programming environment such as JavaScript, although this isn't typically used in a visual programming environment. However, Flash has provided an appropriate environment for our study and upon which we could build our idea. In order to test this idea out and to find out how such a tool might be designed a pilot study has been undertaken. In this study lessons have been developed and presented to subjects in a more or less conventional manner through one-on-one tuition. The purpose of this was to observe the learners and ascertain where they were developing "non-computational" approaches to completing their tasks. The pilot study consisted of a total of seventeen participants consisting of established teachers and PGCE students. The attitudes of the PGCE students towards the idea of learning Flash Action Scripting were generally more positive than those of the established teachers. For example after coding of the interviews that took place before use of the pilot eight comments expressing motivation were found from the trainee teachers while none were given by the established teachers. Both groups were then given one-on-one training in the use of Flash Action Scripting. While the trainee teachers

with their higher level of motivation were quicker to learn and had established good computational thinking processes by the end of the study compared with the established teachers, the latter group (of experienced teachers) were still generally found to have improved attitudes after the exercise. In fact only one of the latter group (one who was intending a future in administration rather than active teaching) failed to overcome boredom and a lack of interest in the scripting process. The results from our pilot study indicate that the use of Flash Action Scripting as a means to develop teaching lessons motivated and engaged both experienced and trainee teachers to learn to program and to develop in their Computational Thinking skills. For example both groups gave increased numbers of responses indicating motivation to learn the Flash Action Scripting after the experiment – the trainee teachers giving 54 positive responses after the study compared to 8 before and the established teachers giving 27 positive comments after the study compared to none before.

In addition valuable information was gained through the experiment into how poor CT skills interfere with learning and how good CT skills can be nurtured. Examples of this were the use of the Flash Action Script concept of layers. All users had not encountered this idea initially and did not initially understand how this concept worked. This invariably led to errors in performance. A particular example was given in section 5.2 where the contents of a layer were deleted rather than the layer itself. This led to a subsequent error statement from the compiler. This error statement was then used to help direct the learners back to the lesson component on layers. However, in the current pilot only eight of the seventeen users were able to complete such a task on their own after a second experience of this tutorial component. It is expected that the tool developed will use these lessons to provide a training tool that will work in a standalone context and provide encouragement to use CT as well as teaching the use of Flash Action Scripting. Error statements from the compiler can be parsed to get inputs to direct the user to the correct part of the lesson to tackle the error they encountered.

The next stage of the work is to develop the automated set of lessons in which the performance of the learners will be monitored by the system and the lesson adapted to their needs as perceived by the system. These lessons will be built in a learning instrument that will take the trainees through a set of steps to build a teaching/learning artefact. Currently the users are expected to be trainee elementary teachers or other teacher trainees from non-computer science/engineering backgrounds and, we surmise, therefore likely to be motivated to learn to produce a teaching/learning asset for early years' education.

REFERENCES

Begosso, L. C., Begosso, L. R., Gonçalves, E. M. & Gonçalves, J. R. 2012. An approach for teaching algorithms and computer programming using Greenfoot and Python. In *2012 Frontiers in Education Conference Proceedings (pp. 1-6). IEEE.*

Bogdanchikov B., Zhaparov M. and Suliyev R., 2013. Python to learn programming. In *Journal of Physics: Conference Series 423.*

Crick, T., & Sentance, S. 2011. Computing at school: stimulating computing education in the UK. *International Conference on Computing Education*, 122–123. Retrieved from: http://dl.acm.org/citation.cfm?id=2094158.

Dorland, M. 2014. Developing creativity and computational thinking in your computing classroom. *CAS meeting Birmingham UK Sept 2014.*

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., and Aldana, G. 2013, Assessment of computer science learning in a scratch-based outreach program. In *Proceedings of the 44th SIGCSE technical symposium on Computer science education, SIGCSE '13. ACM, 2013.*

Grover, S., and Pea, R. 2013. Computational Thinking in K-12: A Review of the State of the Field. In *Educational Researcher, 42(1).*

Hammond, M. 2013. Introducing ICT in schools in England: Rationale and consequences. *British Journal of Educational Technology*,

Lin, C. H. 2012. An Innovative Change in Technology Integration: Training Pre-Service Kindergarten Teachers to Be Courseware Designers. *Creative Education, 03(07), 1177–1183.*

Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. M. 2010 Learning computer science concepts with scratch. In *ICER '10: Workshop on Computing education research, pages 69–76.*

Tondeur, J., Van Braak, J., Sang, J., Voogt, J., Fisser, P., & Ottenbreit-Leftwich, A., 2012 Preparing pre service teachers to integrate technology in education: A synthesis of qualitative evidence. In *Computers & Education 59(1).*

Wieling, M. B., Hofman, W. H. A. 2010 The impact of on-line video lecture recordings and automated feedback on student performance. In *Computers and Education, 54(4).*

Williams, P., 2008. Independent Review Report of Mathematics Teaching in Early Years Settings and Primary Schools, *dera.ioe.ac.uk/8365/1/Williams%20Mathematics.pdf [accessed October 21, 2014]*

Wing, J. M., 2006. Computational Thinking CS @ CMU and Grand Vision for the Field.

Wing, J. M., 2008. Computational thinking and thinking about computing. In *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 366(1881), 3717–25. doi:10.1098/rsta. 2008.0118.*

Zhang, D., Zhou, L., Briggs, R. O., & Nunamaker, J.F., 2006, Instructional Video in e-learning: Assessing the impact of interactive video on learning effectiveness. In *Information and Management, 43 (1).*

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

APPENDIX H

Publication: Chapter of Book 2016

Computational Thinking - A Tool to Motivate Understanding in Elementary School Teachers

Ms Erni Marlina Saari, Dr Peter Blanchfield and Dr Gail Hopkins

School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, UK
{psxems , pszpxb, pszgtr}@exmail.nottingham.ac.uk

Abstract: The need for teachers of Elementary School children to learn to program or rather to understand the Computational Thinking behind programming has been accelerated in many countries by the mandated teaching of programming in this context. A major problem is that the majority of such teachers have no concept of how to program and naturally are not motivated to learn programming. The need to teach programming will motivate most but to learn through a tool that can be seen to have intrinsic value in their role as teachers have a greater potential. This paper describes the design and early implementation of a tool to use the learning of Flash Action Scripting as a motivational mechanism for trainee teachers. The intrinsic value to them is intended to be the utilisation of the learned Action Scripting skills to produce their own teaching material. Initial results indicate an enhanced engagement and motivation to learn to program and improved confidence in doing so.

Keywords: Computational thinking, Flash Action Scripts, Trainee Teachers.

1 Introduction

With the increasing level of use of computers in all aspects of human lives from the almost invisible use in domestic appliances to the now nearly ubiquitous use of smart phones in communication, the need to produce more programmers has never been greater. For the software industry from the smallest start-up to the major development organisations such as Microsoft and Apple the need for new staff outstrips the supply. As a result the leading software developers have been putting pressure on governments worldwide to come up with plans to increase the numbers of programmers produced by their education systems. It is felt that much of what has previously been taught in what is called Information Communications Technology (ICT) in schools only led to frustration for those who were looking to develop an understanding of computing. (Crick and Sentence, 2011) In the UK the pressure for more teaching of computing has led to a number of different initiatives. Starting with secondary schools the teaching of programming has now been mandated in UK schools. The same is happening in Malaysia; with an increased desire for more programmers to be developed there is now a need for greater levels of understanding of this process among teachers. However, those who are attracted to teaching as a career seldom come from the group who have a high level of computer programming exposure. Many (including those we have worked with in our experiments and are quoted in later sections of this paper) find the idea of learning to program "frightening" and this does not help them to be motivated to learn, even though they are aware that they will be called upon to support this learning in their classroom. For this reason there have

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

been numbers of initiatives to enable teachers at all levels to engage with learning how to program themselves and thus be more ready to teach the subject. The generalist nature of teaching in elementary schools has already meant that earlier programmes to promote maths and English teaching in elementary schools has required retraining programmes for teachers in those schools and further development of subject specialist teachers in those schools and school districts (Williams, 2008). In the UK the Computing at Schools initiative (CAS) has developed with a similar aim of enabling the teaching of computing and especially programming in schools. At first this attracted secondary school teachers who were already involved in the teaching of the existing IT courses in those schools. Increasingly these efforts are now aimed at elementary teachers and the background of these teachers requires a different level of learning.

Much of the work done to introduce programming concepts has concentrated on how to develop understanding of programming in young children. The Scratch language has been used successfully in this area (Franklin et al, 2013) as has the use of Python Python (Begosso et al 2012, Bogdanchikov, Zhaparov & Suliyev, 2013). The former uses a symbolic drag and drop interface that allows programming constructs to be easily put together in a visual way. The focus of the programs produced is often on controlling visual features such as computer sprites and producing games from these. Such things tend to be attractive to children who like playing games and thus become motivated by the idea of producing their own. However, teachers on the whole are less motivated by this and indeed the majority, like those in the experiments we have undertaken, find little motivation to learn to program from these initiatives. The concept behind the research in this paper is that teachers need to be able to see intrinsic value in learning to program in order to lose their fear of learning the process. We have proposed that by learning Flash Action Scripting through the use of the appropriate tools teachers will be motivated to learn as they will see the possibility of being able to create their own tools to enhance their teaching of the other subjects they present.

2 The role of Computational Thinking in Learning to Teach Programming

Many involved in trying to get teachers to be able to teach programming have been attracted to the concept of Computational Thinking (CT). This phrase has been made popular by Wing and others (Wing, 2006, 2008, Grover and Pea, 2013) but there seems to be a number of potential interpretations of what it is and why it is valuable to learn CT skills even for those who don't see themselves as programmers. Dorling (2014) has identified five aspects of CT that can be seen as potentially valuable in learning to program but also in tackling other aspects of daily life. He specifies these as algorithmic thinking, decomposition, abstraction, generalisation and evaluation which relate as follows:

- Algorithmic thinking will relate to the choice of approach taken in solving a problem.
- Decomposition involves breaking the problem down into manageable steps.
- Abstraction – involving the removal of the specific problem so that the more generic problem being tackled can be seen.
- Generalising the solutions – using ideas solved before for example as in using sub programs (functions etc.) in a computer program.

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

- Evaluating – the solutions and the algorithms

One question which suggests itself is how we will know if someone is using CT in their solution to a problem? One might also ask what sort of problems we would use CT for. Also we could ask, "Does CT mean thinking like a programmer or thinking like a computer?" To what extent will we expect people to apply CT in every part of their lives? If teachers for example were able to see the use of CT in their teaching lives, would that improve their motivation to learn to program?

It is useful to think of places where we might expect that CT is clearly valuable. Take the process of entering marks from exam scripts into a spreadsheet. In general the exam papers will have become disordered as the marking process took place. However, the spreadsheet will want the marks to be added in a specific order. We could approach this problem in a purely ad hoc way. We pick up the first paper locate the student ID, scan through the list of IDs in the spreadsheet and add the mark(s). If we have very few papers to grade this process might not take too long. However, if we have a large number of papers we would probably quickly conclude that it would be better to sort the papers first. First of all we are asked to think of this as a sorting problem. This is the process of abstraction. Sorting something with numbers on it that identify the order is a common problem. Thinking like a programmer we would then decide on an algorithm to implement the sorting process. We then need to evaluate the effectiveness of the available algorithms for the task at hand. The level of disorder the papers are in, for example, might influence our approach. If most of the papers are ordered we might choose a simple algorithm, if they are highly disordered we might go for one like the merge sort. Once we have decided to use merge sort we now proceed to think like a computer. We separate the large pile into small piles counting off the number into each pile sequentially. The size we need the piles to be would be chosen based on the number we feel we can easily order. We now act as a programmer again and choose to stop at (say) four as we feel we can now use our human pattern recognition skills to sort these – a programmer thinking choice not a computer thinking choice.

A second example would be choosing a route home using a map. It is probable that we would be inclined to use a satellite navigation system to do this. What we now should do is use the CT concept of evaluation to evaluate the solution given by the software. We are unlikely to get the map and start to plan the process by iterating through the Dijkstra's algorithm in our heads or on paper. This would have involved us in thinking like a computer and while it is an efficient algorithm for computers to undertake it is only one a human would be tempted to use on very small problems and in such cases a brute force solution might end up being more reliable. Thus in this case it is really the evaluation aspect of CT that is likely to be useful.

A more important concept in the current example is the idea of using a computer program to produce a teaching artefact. In order to produce more sophisticated animations using Flash it becomes necessary to use a number of aspects of CT. Breaking the problem down into manageable parts will definitely be used. Generalisation will also be valuable as many of the tasks we will want the artefact to undertake will be very similar. As the process becomes more sophisticated we begin to get value from actual algorithm design. Our solution can become more efficient as we look at the actual scripts that are produced and we can start to extend machine produced codes to increase the efficiency of what we produce. Thus though some have suggested that Flash is now coming to the end of its life we propose that learning the use of Flash Action Scripting in particular is very valuable for

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

teachers. First it is more intrinsically motivating as the teachers can see the value to them of learning to use the software. Secondly a lot of aspects needed in programming are developed and other software interfaces for producing animations and stories work in a very similar way. Thirdly the teachers can begin to interact with a scripting program that is powerful and works with many of the concepts they will need to learn to teach. Additionally others have pointed out the value of teachers being able to engage with the creation of their own teaching aids (Lin 2012).

3 Training of Teachers in The Use of Educational Technology Using flash Action Script

The initial motivation for this research was experience of one of the researchers in teaching at a teacher training institute in Malaysia. This experience showed a great tendency of learners to adopt approaches to learning the use of Flash Action Scripting in one of their classes that did not involve Computational Thinking (CT) but rather using ad hoc solutions to problems being faced. Generally the lessons were prepared in a traditional way with the staff member giving formal lectures backed up by PowerPoint presentations and live demonstration in front of the class. Learners were then given example tasks to undertake for themselves. On the whole instead of understanding the process students were inclined to try to copy and paste from the original lessons and while this might appear to be using the principle of "re-use" without first evaluating the codes they were copying the result would not be what they were aiming at. This leads to frustration in the users and feelings of fear of the whole concept of using a tool. The student teachers were surveyed after their training over their attitudes and it became clear from these initial surveys that the students could see that learning to use Flash Action Scripting would be intrinsically valuable to them. It was thus decided to develop a pilot study in which the possibility of developing an automated teaching tool would be tested. It would be necessary in the study to identify specifically where learners were adopting non CT approaches and how to correct these problems. Once these were identified the automatic tool could be designed to do the same. Utilisation of such an environment builds upon the views expressed by others of the value of using instructional video in e-learning (Zhang et al, 2006, Wieling & Hofman 2010).

Hammond (2013) mentions the significance of ICT even if engagement is not to be taken for granted and the effect of the learning is complicated. Others have sought to emphasise the need to integrate technology into education. For this to be done successfully it needs to be integrated into the training of the teachers. The arguments around this have been summarised in (Tondeur et al, 2012).

4 Experimental Method

For the pilot project two groups were recruited. Though the difference between the groups is not the major focus of this paper one group was of eight experienced teachers and one of eight trainee teachers. No member of either group had previously taken a computing or engineering course where they would have been exposed to training in programming. All of the participants were from Malaysia. The experienced teachers were all in the UK studying for advanced degrees in subjects of value to them. One of these was expecting to return to an administrative role on completion of his studies. The trainee teachers were all on a PGCE course, having

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

completed a variety of first degrees. They were expecting to return to work in schools in Malaysia. The experiment was subject to the ethics approval process and all participants were invited and partook through their own free choice. All gave permission for their responses to be reported but in an anonymised form. They are therefore quoted in an anonymous form. The initial experimental process involved:

- Pre-session interviews to identify attitudes and experience
- A session consisting of a sequence of one on one lessons in the use of Flash Action Scripting and the Flash tool
- A parallel set of example exercises to be undertaken by the participants
- Post –session interviews to identify any changes in perception

In addition to the interviews the participants' performance during the lessons and practice sessions was recorded. Activities were transcribed for analysis and coded. The frequency of the codes found was then analysed. The approach used is based on Grounded Theory and the analysis method is common in that approach.

Charmaz (2006) depicts grounded theory activities such as coding as 'grounded theory guidelines'. She indicates that these guidelines can be used in a wide range of studies by researchers who all bring with them their own background, understandings and theoretical framework.

Line by line coding is a category of coding used for the current research. Charmaz recommends that there be at least two primary phases of coding in grounded theory: the preliminary phase where data is assessed by segment or line by line; and the second focused phase where substantial amounts of data are ordered by the most frequently occurring codes (2006).

Turning a code into a classification occurs through analysing an initial code and defining its properties (Charmaz, 1990). At this stage the researcher's interests may influence what questions are asked of the data. For instance, as the current researcher has a focus on motivation to learn programming questions specific to obtaining more information about participant's experiences with computing or programming were asked.

Open coding, which is also denoted to as first-level coding, is often expressed using the participants' own words and was an early step in the coding process in this study. Charmaz discusses coding and recommends that the preliminary line by line coding that many grounded theorists engage in allows them to remain connected to the data.

Open coding was first undertaken. Hundreds of line by line codes were identified in the interview data, usually using participants' own words, some which included participant comments such as:

- Not feeling good enough
- Feeling Inferior
- Student engagement
- Everything needs to be creative
- Feeling Scared
- Feeling demotivated

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

- Difficult to learn
- Too technical
- I am social art students
- Why I should learn programming
- Module prepared by school
- Time consuming
- Low self-confidence

The formal codes reflected the following positive attitudes:

- I - Interest
- E - Engagement
- M - Motivation
- C - Confidence
- En - Enjoyment
- A – Affordance

And the following negatives:

- B - Boredom
- L - Lack of Motivation
- Co - Confusion
- F - Fear
- D - Difficulty

These codes were therefore applied to the transcriptions. Examples of the open coding are included in table 3 in Section 5.4 below. The time taken to develop a level of engagement was also recorded as a separate measure of participant engagement/motivation.

The lessons used in the pilot study are briefly described the next section.

5. Pilot Lessons

The idea of the pilot lessons was to identify where CT thinking was being applied or not being applied by the participants. The aim of this was to identify how to develop an automatic tool that itself could identify failing in thinking and therefore guide the participants in the learning process in order to encourage them to use computational thinking. The content and approach of the pilot lessons was modelled on the approach being used in training teachers in the use of Flash Action Scripting in Malaysia. The lesson first introduced the task that was to be undertaken and then led the participants through the process of developing the example of a teaching artefact for early years learning. Some of the teachers were not teaching in primary school so there was the possibility that this lesson would not motivate them. Similarly not all of the trainee teachers would end up as primary school teachers. It is possible therefore that better results would be obtained with a range of

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

example lessons but this was not practical for the experiment. The details of the lesson have been specified elsewhere (Saari, Blanchfield and Hopkins 2015). The purpose of the lesson was to show how the Action Scripting language could be used to produce an artefact that might be seen as valuable to the teaching of the participants in future. The lesson involved a sequence of stages of increasing complexity so that computational thinking ideas could be introduced steadily. The first process was to get the participants to understand the working of the interface itself and then how each component is used. To do this the production of a simple menu system formed the first part of the lesson. An example of how to get the first menu button to work was given and in the subsequent exercises the participants were required to add other buttons to the artefact being developed and to add appropriate action responses. At each stage wrong thinking by the participants was identified and corrected and correct thinking was identified and encouraged. The process of the lesson involved a lot of interaction with the researcher acting as tutor and the content of this interaction was vital to the design of the automatic tool that has subsequently been developed. Examples of the screens produced by the artefact being developed are given in the following figures.

The next section details the results that were obtained from the pilot study and we then move on to introduce how the resultant automated tool has been developed and testing of it begun.

5.1 Initial Interviews.

As mentioned above a set of pre interviews were undertaken in order to allow for comparison between the pre and post session attitudes to programming and the use of Flash Action Scripts in particular. Certain differences also emerged between the qualified and trainee teachers in this session. Of the established teachers the responses to questions regarding the likely usefulness of learning Action Scripting were negative and all were nervous of the need to use a technical tool. A typical response was "*I am not good in using technology especially programming. I am afraid of using technology actually. I don't like to use computers …*"

The PGCE (trainee) students were less reticent to take part. They all felt it would be good to learn to use Action Scripts as it would enable them to use the tool to make their own teaching/learning instruments. However, they generally did not feel it would be easy to learn as it appeared to involve a similar process to learning more formal programming systems like C++. A typical quote from this group was "*I want to learn this programming language and I will come out with my own teaching module soon!*" There were some negative attitudes, for example one subject asked "*Why would I need to learn Flash or have to design a learning tool as all of my lesson material will be provided by the school?*"

5.2 Performance in The Practical Sessions

During the practical sessions the volunteers were required to produce code that performed the set of functions shown in Figures 1 to 3. The performance was monitored and the actions were recorded using screen capture so they could be analysed afterwards. The major problem was the tendency to copy and paste previous examples rather than to analyse the problem, understand the components that were needed and develop the appropriate set of steps to produce the artefact. When they found their actions had not produced the correct behaviour they were asked to analyse why this had happened. They were taken back to previous steps and the lesson was repeated until they were able to produce the correct output. Different processes in producing a good

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

learning artefact were identified against the Computational Thinking approach. A good example is the idea of abstraction as embodied in the concept of a layer.

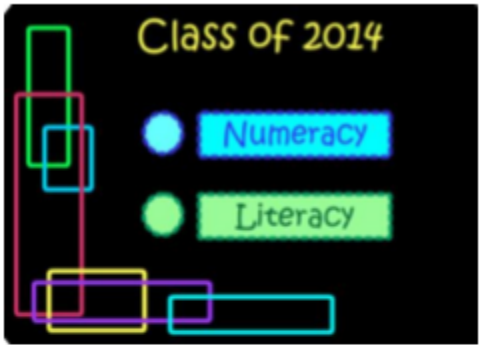


Fig.1. Original page



Fig.2. Literacy screen

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20



Fig.3. Scene with added animation path

In Flash Action Scripting the layer is a basic abstract component that gets non abstract components – actual text, pictures, and scripts – attached to it. As a result a user thinking computationally would be expected first to engage with the abstraction – by invoking a new layer for a new action – then to populate it. Thinking non-computationally exhibited itself in various different ways. One typical example was when users wanted to remove an existing layer. Instead of selecting the layer and deleting it the users who were not thinking about the abstraction would delete the contents of the layer rather than the layer. This would lead to an error when the code was compiled and an error message that was not understood by the user. Such error statements alert the teacher with experience to the lack of correct thinking. The user is then directed to the part of the lesson on layer creation.

The PGCE students all grew in confidence and enthusiasm for the process of learning Flash Action Scripting more quickly than the experienced teachers, however, all participants were able to finish the first task correctly. Enthusiasm for and motivation to use the tool were indicated by comments made by the experimental subjects. The trainee teachers generally were more enthusiastic to start with and thus were more motivated to understand and correct their mistakes.

The average time to complete the initial task was 35 minutes. The longest time taken for the process was one hour and this was from a subject who had shown the lowest motivation for the task as he intended to return to administration within a school and not teaching.

5.3 Post-Session Interviews

After the first practical session the subjects were all re-interviewed. They were asked questions about their attitude to learning Flash Action Scripting having now been exposed to it. Questions centred on their views regarding the potential usefulness of learning to use Flash Action Scripting, technology in general, their motivation and ability to learn Flash Action Scripting and their understanding of the steps used in computational thinking.

Most of the experienced teachers felt an improved attitude towards the usefulness of the process. The teacher who intended to return as an administrator had not improved his attitude at all. One of the others still did

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

not see that it could be useful to them in their teaching or to teaching in general. The other five were enthusiastic about the possibilities. Of the PGCE students all were convinced not only of the value of the tool but also of their ability to learn to use it.

When asked about the use of technology in general to support their teaching the experienced teachers had generally gained an improved attitude.

Most of the experienced teachers did not feel intrinsically motivated to learn programming but had found the process of the lesson enjoyable and felt they had been successful in learning how to use the tool. Eight of the ten trainee teachers expressed the idea that they found the learning intrinsically motivating as they felt they would benefit in their own careers. These have all moved on to the further study of the subsequent lessons and have been willing to try to learn further by themselves!

The CT skills needed in the task to perform it efficiently were those of Algorithmic thinking and Abstraction. Eight of the trainees exhibited Algorithmic thinking by the end of the first lesson and seven had demonstrated use of Abstraction. However, only four of the experienced teachers had developed this CT approach. For example in the use of layers – linked to the concept of abstraction – seven of the trainee teachers and one of the experienced teachers had understood how to properly use layers.

5.4 Thematic Analysis

As was explained in Section 4 above, the recordings of the interviews during the practical session were all transcribed and the transcriptions were then thematically coded looking for different indicators, as explained in section 4, relating to positive and negative attitudes. Table 1 below shows themes that emerged prior to the experiment taking place.

Table1. Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **before** taking part in the experiment.

Codes	TT Frequency	ET Freq.	Total
Interest (I)	23	17	50
Engagement (E)	24	15	39
Motivation (M)	27	15	42
Confidence (C)	8	0	8
Enjoyment (En)	12	11	23
Affordance (A)	2	3	5
Boredom (B)	3	11	14
Lack of Motivation (L)	6	15	21
Confusion (Co)	27	32	59
Fear (F)	15	12	27
Difficulty (D)	32	26	58

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

As can be seen from the results in Table 1 the trainee teachers demonstrated higher levels of positive attitudes to the potential value of the training they were about to undergo, when compared with the experienced teachers (indicated by a total of 96 coded positive statements by the trainee teacher group and only 61 by the experienced teachers). However both groups had negative attitudes in some of their statements and in particular felt the process of learning Flash Action Scripting would be difficult (roughly equal numbers of negative comments on difficulty) and that they found what they were being asked to do potentially confusing (again with similar numbers of negative comments from both groups). None of the experienced teachers expressed any confidence that they would develop the ability to use the tool but eight comments from the trainee teachers expressed such confidence.

Table 2. Frequency of appearance of code phrases in use by the Trainee Teachers (TT) and the Experienced Teachers (ET) **after** taking part in the experiment.

Codes	TT Frequency	ET Freq.	Total
I	39	19	58
E	29	24	53
M	54	27	81
C	22	20	42
En	22	19	41
A	12	8	20
B	0	2	2
L	0	2	2
Co	13	10	23
F	0	0	0
D	7	3	10

Table 2 demonstrates the frequency of code words in the post-experiment interviews and it can be seen that both the trainee and experienced teachers had significant improvements in their attitudes towards learning Flash Action Scripts and programming technology in general. Of particular note is the fact that no phrases emphasising "Fear" (F) were recorded in the post-experiment responses by either group though some "Confusion" (Co) still remained in both groups (a total of 23 such comments roughly equally divided between the groups) and there was still a residual view that the process was "Difficult" (D) though this was much reduced with a total of ten comments expressing difficulty after the experiment compared to the 58 difficulty comments made before the experiment. Only one of the participants (the experienced teacher now moving into administration) used "Boredom" (B) and "Lack of Motivation" (L) phrases – two responses fitting this description for both categories. Together with the increased number of positive comments on motivation (a total of 81 such comments after the experiment compared to 42 before the experiment) this indicates that those who were expecting to be teachers after their courses all believed the use of Flash Action Scripting to be interesting and motivating. There were few phrases mentioning "Confidence" (C) before the experiment- 8 from the trainee teachers and none from the experienced teachers; after the experiment only one of those involved (the

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

administrator) failed to respond with any confidence statement. A total of 42 such comments were recorded from the rest of the subjects.

The time taken to begin to show signs of engagement and enjoyment of the use of Flash was generally lower for the trainee teachers than for the experienced teachers but only one (the administrator) failed to complete the task and failed to become engaged in the process after 20 minutes.

Examples of quotes reflecting the codes used are listed in Table 3 below. Some of the phrases quoted show more than one of the coded attitudes. They were coded as representing both the trainee and experienced teachers' attitudes but are only shown as illustrating one. All of the comments listed were given by trainee teachers (respondent TT5 means trainee teacher 5) though similar comments were made by the experienced teachers.

Table 3. Examples of coded comments given by the Trainee Teachers (TT 4 to TT 10).

Code	Quote
I	"As a first time learner like me, this software is easy to learn and the instruction is quite simple..." (respondent TT5)
E	"Very good! I think students will feel more enthusiastic to learn the subject if the teaching instrument used is interesting" (respondent TT4)
M	"I think it is attractive software to learn and teach also. Besides teachers learn how to program, I think we can teach our students to learn how to program as well." (respondent TT10)
C	"I am looking forward to studying more on this and producing my own lesson.." (respondent TT9)
En	"I knew that Flash can do interesting tasks like interesting courseware to gain students' attention." (respondent TT6)
A	"Can you give me a few minutes to try on my own on this software? I would like to produce a small interactive lesson about our lesson just now. I am just so excited to do animation and navigation buttons." (respondent TT4)
B	It is too hard for me to start learn the new software and need to struggle about the technical terms...." (respondent TT5)
L	"... there is just a waste of time to learn new software, I prefer to get the teaching tools that will be provided..." (respondent TT5)
C	make me feel confuse about technical terms and

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

	interface too..." (respondent TT5)
F	" I am afraid I could not produce the module at all because I am not good handling the technical tasks.." (respondent TT5)
D	" ..sometimes it very hard to understand especially the complex syntax..." (respondent TT7)

Our results demonstrated that use of the tool would increase interest in Flash Action Scripting and enhance enthusiasm for programming with reduced levels of confusion and frustration. The next step has been the development of the tool based on the findings and approach taken in the preliminary experiment.

6 Development of The Tool

The tool itself has been developed using Flash Action Scripting. By analysing the results of the experiment above it was possible to identify points at which the participants made errors. The tool is designed around the lesson that was provided in the one-on-one training. This has been split into phases such as:

- Introducing the tool
- Identifying the function of the basic elements
- Leading the learner through some phased exercises
- Querying the user on their performance
- Using the data from the first experiment to identify if any mistakes have been made
- Leading learners through corrective lessons based on their performance
- Repeating previous stages with increased complexity

The tool also tries to identify aspects of the training in relation to basic computational thinking components. For example:

- Decomposition involved in breaking the problem down (Dorling 2012), usually in a forward direction, identifying major tasks that move the solution to the objective.
- Staged worked examples which are implemented by giving students partially completed code and asking them to complete it correctly. This is done in the development environment to provide feedback.
- Encouraging the learners to write pseudocode to solve a problem is a computational thinking skill; translating that pseudocode into Action Script is a programming skill.

Examples of screen shots from the tool are given below:

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20



Fig.4. An example of a training stage introducing the formatting tools



Fig.5. Looking at producing scripts using generalisation through switch cases

The next section provides some early results from testing the automated tool.

7 Initial Testing of The Automated Learning Tool

A similar set of participants was recruited for the initial testing of the tool. Eight trainee teachers were chosen. Each of these had a similar background to the first group. Pre-trial interviews were once more undertaken and similar responses to the thought of learning to program and to Action Scripting were again found. One issue was

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

that the tool was designed to be used by the participants in their own time as well as while being observed by a researcher. This produced an interesting response from one of the new participants:

"… I think this is an interesting programming language to be learned with the proper guidance from the tutor or instructor…"

A second participant responded:

"What if I face the problems over the weekend?"

This exchange showed their fear, perhaps, of the automated teaching tool. However, these fears proved unfounded and having had a preliminary observed use of the tool; the participants all found it easy to use. Comments by the same two participants after use of the tool included:

"… I can now do my own programming without any helps!"

"This is really a great tool! I can work with it anytime, anywhere…"

The most encouraging feedback was received from one of the participants:

"This is like online manual and at the same time module that really give a big impact for me as a trainee teacher, to develop my own teaching module soon!"

And finally from the participant who expressed fear of working at home on their own:

"Can I get a copy of this tool?"

Further more formal analysis of the results has yet to be completed but these initial responses indicate that the tool will be at least as effective as one on one lessons.

7. Conclusions

In the introduction we highlighted that the need for more programmers is now urgent. Software companies and governments around the world have become convinced that a change in the way technology is taught in schools is needed. It is felt vital that children begin to learn to program and more importantly think like programmers. The concept of Computational Thinking (CT) has recently developed and the role that this plays in developing the correct thinking to be a programmer is clear. In the introduction we also highlighted the need there will be to train elementary school teachers to be able to teach programming. It is not expected that these teachers will themselves become programmers. Also as was explained in the introduction ways of getting them to learn to program that are effective with children will not be effective with them as on the whole they will be less attracted to the idea of being able to develop their own simple games. Through our experiments we have shown that teachers who are not initially secure with the idea of learning to program can be motivated to learn by using a tool that they can see has intrinsic value to them. The tool we have begun to develop is based around teaching the use of the Flash animation tools and the production of Flash Action Scripts. Our experiments with trainee and experienced teachers have identified that this is true and those who were involved in our pilot study of one-

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

on-one lessons increased both in their confidence to use the tool and their CT skills. It is thus believed that having undergone such training they will be less apprehensive of the material they have to deliver to the children they teach. Using the results of the observations of the participants during their learning it has been possible to begin to produce an automated tool that guides learners through the process. In the pilot study we identified ways in which wrong thinking or non-CT were hindering the learning of the use of Flash Action Scripting. In the automated tool ways of identifying why wrong actions have been undertaken have been introduced. Repeating lessons about the specific subjects of need or giving special instruction when a cause of wrong thinking has been identified has allowed the unsupervised learning to continue using the tool. An initial experiment with the tool has shown it to be effective in teaching the use of Flash Action Script, improving learner confidence in their ability to use technology and improving their Computation Thinking. The tool will now be refined as a result of the testing and further improvements are expected.

8. References

Begosso, L. C., Begosso, L. R., Gonçalves, E. M. & Gonçalves, J. R. :An approach for teaching algorithms and computer programming using Greenfoot and Python. In Frontiers in Education Conference Proceedings (pp. 1-6). IEEE (2012)

Bogdanchikov B., Zhaparov M. and Suliyev R.: Python to learn programming. In Journal of Physics: Conference Series 423 (2013)

Charmaz, K.: 'Discovering' Chronic Illness: Using grounded theory. Social Science Medicine, 30(11), 1161 – 1172 (1990)

Charmaz, K.: Constructing grounded theory: a practical guide through qualitative analysis, Sage Publications, USA (2006)

Crick, T., & Sentance, S.: Computing at school: stimulating computing education in the UK. International Conference on Computing Education, 122–123. Retrieved from http://dl.acm.org/citation.cfm?id=2094158 (2011)

Dorland, M.: Developing creativity and computational thinking in your computing classroom. CAS meeting Birmingham United Kingdom September (2014)

Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., Dreschler, G., and Aldana, G.: Assessment of computer science learning in a scratch-based outreach program. In Proceedings of the 44th SIGCSE technical symposium on Computer science education, SIGCSE '13. ACM (2013)

Grover, S., and Pea, R.: Computational Thinking in K-12: A Review of the State of the Field. In Educational Researcher, 42(1) (2013)

Hammond, M.: Introducing ICT in schools in England: Rationale and consequences. British Journal of Educational Technology (2013)

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

Lin, C. H.: An Innovative Change in Technology Integration: Training Pre-Service Kindergarten Teachers to Be Courseware Designers. Creative Education, 03(07), 1177–1183 (2012)

Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. M.: Learning computer science concepts with scratch. In ICER '10: Workshop on Computing education research, pages 69–76 (2010)

Saari, E. M., Blanchfield P., and Hopkins, G.: Learning Computational Thinking Through the Use of Flash Action Scripts – Preparing Trainee Elementary School Teachers for Teaching Computer Programming. In Proceedings of the 7th International Conference Computer Supported Education (CSEDU) pages 75-84, Lisbon, Portugal (2015)

Tondeur, J., Van Braak, J., Sang, J., Voogt, J., Fisser, P., & Ottenbreit-Leftwich, A.: Preparing pre service teachers to integrate technology in education: A synthesis of qualitative evidence. In Computers & Education 59(1) (2012)

Wieling, M. B., Hofman, W. H. A.: The impact of on-line video lecture recordings and automated feedback on student performance. In Computers and Education, 54(4) (2010)

Williams, P.: Independent Review Report of Mathematics Teaching in Early Years Settings and Primary Schools, http://dera.ioe.ac.uk/8365/1/Williams%20Mathematics.pdf [accessed October 21, 2014] (2008)

Wing, J. M.: Computational Thinking CS @ CMU and Grand Vision for the Field (2006)

Wing, J. M.: Computational thinking and thinking about computing. In Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 366(1881), 3717–25. doi:10.1098/rsta.2008.0118 (2008)

Zhang, D., Zhou, L., Briggs, R. O., & Nunamaker, J.F.: Instructional Video in e-learning: Assessing the impact of interactive video on learning effectiveness. In Information and Management, 43 (1*)* (2006)

© Springer International Publishing Switzerland 2016
S. Zvacek et al. (Eds.): CSEDU 2015, CCIS 583, pp. 348–364, 2016.
DOI: 10.1007/978-3-319-29585-5_20

APPENDIX J

ERNI MARLINA SAARI (PhD. Candidate)

School of Computer Science, Jubilee Campus.UK

PILOT STUDY TASK

Name:
School/Faculty:
Programme:

1. Please produce a task as shown in the following figures.

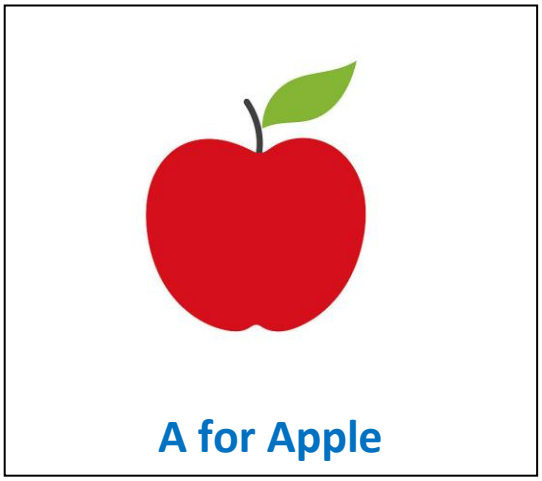


Figure 1



Figure 2

2. You should draw an image of apple and text as above for the first scene.

3. You are encourage to produce the second scene with the text 'THANK YOU' and a 'back' button.

4. Please complete a task within 60 minutes. Thank you.

APPENDIX I

ERNI MARLINA SAARI (PhD. Candidate)

School of Computer Science, Jubilee Campus.UK

OPEN-ENDED QUESTIONS (PRE TEST)

Study 1

Please answer the questions provided. Your cooperation is highly appreciated. Thank you.

Name:

School/Faculty:

Programme:

Section A

1) Do you think studying computer programming such as Flash action script is crucial for trainee teachers?
2) Do you think you have acquired enough computer knowledge and skills?
3) Do you think you would like to gain more computer knowledge and fundamental programming skill such like flash action script?
4) Is it necessarily ICT subjects being taught in Educational system in more details?
5) Do you have any experience(s) using Flash Action Script? What is your comment?
6) What subjects do you think suitable that can be designed using flash action script?
7) Why teachers might need to have knowledge in flash programming?

8) How does the multimedia tool can facilitate learning?

9) Do you think teachers can design their own lesson plan using Flash?

10) What other features do you suggest that teachers will obtain by learning flash action script?

Choose any **5** from the lists that give motivation to learn the Flash Action script.

- Because you think it will be easy
- Because you have been told you have to do it
- To learn how to program
- Because you wish to become a professional programmer
- Because you would like to write programs for your own use
- Because an understanding of programming will help in your future career
- To understand more about programs and programming
- To improve your general ability to solve problems
- To learn general study and transferable skills.
- To get the module credits

THANK YOU VERY MUCH FOR THE KIND SUPPORT ☺

APPENDIX II

OPEN-ENDED QUESTIONS (POST TEST)

STUDY 1 PhD: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 60 minutes

Venue: UPSI Malaysia (Pentagon Lab)

Class: MMD3023 2013/2014

1. Have you heard about Flash Action Script before?
2. How about the impact for the students?
3. Can you explain about the usefulness of Flash?
4. What can you describe the good think about Flash?
5. What do you think about today's lesson?
6. Can you share with me the experience using the software just now?
7. Do you feel convenient now in using Flash Action Script?
8. How Flash Action Script programming can help students?
9. How about the benefits for teachers?
10. As non-computing background user, what do you think about Flash Action Script? Do you think flash is easy to learn?
11. What do you think if teachers are highly encourage to learn programming?

12. Do you think teachers could become a creative designer if they expose themselves in producing their own teaching module? Need more comments /examples?

13. How do you organize your folders on your study table? How about in your laptop or desktop? Can you see the relationship of the questions with the function of layers in flash action script?

14. Do you think Flash Action Script can be used as one of the software in this class? Why?

15. If you are a teacher, how do you think about managing classes' modules? For particular semester?

16. Even though you are still new with Flash Action Script, do you think this helps you a lot in terms of getting the basic skill of programming before you learn other advance programming language?

17. Can you conclude about the easiness of learning to program using flash Action script?

APPENDIX III



OPEN-ENDED QUESTIONS (PRE TEST)

STUDY 2 PhD: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 60 minutes

Venue: University of Nottingham, Computer Lab

1. You have learned about Flash Action Script before right?

2. What do you think of Flash Action Script?

3. Do you know other programming language besides Flash? What it is?

4. Do you think learning programming is important nowadays and why?

5. How does Flash can help students or teachers?

6. Why Flash should be learned by students and teacher or maybe everyone?

7. Do you feel convenient now in Flash Action Script?

8. Are you excited in learning programming? Is it because Flash Action Script ? Please explain.

APPENDIX IV

OPEN-ENDED QUESTIONS (POST TEST)

STUDY 2 PhD: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 60 minutes Venue: University of Nottingham, Computer Lab

1) What do you think about today's lesson?
2) Do you feel confident now using Flash Action Script?
3) Can your share your experience using the software just now?
4) What do you think Flash Action Script can help students?
5) How about the benefits to teachers?
6) As non-computing background user, what do you think about Flash Action Script? Do you think flash is easy to learn?
7) What do you think if teachers are highly encourage to learn programming?
8) Can you give some comments about current computing or programming subjects you have learn so far?
9) Do you think teachers have possibility to become a creative designer if they expose themselves in producing their own teaching module? Need more comments /examples?
10) How do you organize your folders on your study table? How about in your laptop or desktop? Can you see the relationship of the questions with the function of LAYERS in flash action script?
11) Do you manage to create your own navigation button just now confidently?
12) Why and how do you think you able to produce the extra buttons as requested?
13) If you are a teacher of few subjects, how do you think about managing classes' modules/details?
14) Do you think that by creating the teaching module, actually you can combine everything in one tool and accessible anytime/anywhere?

15) Even though you have designed or created one module, do you think you can apply the same concepts to other subjects too?

16) How about the easiness of learning to program using flash action script?

APPENDIX V



INTERVIEW QUESTIONS (PILOT - STUDY 3 QUESTIONS)

Phd Title: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 30 minutes

Venue: University of Nottingham, Computer Lab

1. Did you have any experience learning any programming or computing subject?
2. Have you heard about Flash Action Scripts before?
3. What do you think about Flash Action Script and other programming language? Please explain.
4. Do you think by learning programming such as Flash Action Scripts could be beneficial? Why?
5. How Flash Action Script programming can help students?
6. How about the benefits for teachers and students?
7. As non-computing background user, do you think Flash Action Script is easy to learn?
8. Why do you think teachers should learn programming?
9. Give any examples of programming language that you know?
10. Could you list to me any methods that can help students or users to learn programming besides this teaching tool?

THANK YOU VERY MUCH FOR THE KIND SUPPORT ☺

APPENDIX VI



EVALUATION WITH SUBJECT MATTER EXPERT

Phd Title: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 30 minutes

Venue: University of Nottingham, Computer Lab

Name
Gender:
School/Faculty:

Please indicate the extent to which you agree with each of the following statements about the heuristics that you used in this study by using the scale below:

1. Could you have a look on the usability of this tool ?Please give some comments.
2. Please indicate any weakness of the main interface design?
3. How about the colour used of the whole design?
4. How about the interactivity section? Is it suitable to be used?
5. Please indicate any wrong doings on each scenes.
6. Does user know every steps he or she should do on any sections?
7. Does the tool helps user in understanding the functions of the scripts? Please comments.
8. Does the tool really motivate user to learn how to program? Please discuss.

9. Does the tool improve the perceptions of some people about the difficulties in learning programming?
10. Could you briefly explain about the tool?

APPENDIX VII

POST- QUESTIONNAIRES (STUDY 3)

Phd Title: TOOLBOX FOR ADOPTING COMPUTATIONAL THINKING THROUGH LEARNING FLASH

Duration : 30 minutes
Venue: University of Nottingham, Computer Lab

Please answer the questions provided. Your cooperation is highly appreciated. Thank you.

Name
Gender:
School/Faculty:
Date:

Section A

Please answer the questions.

1) Do you feel confident now about learning a programming and Why?
2) Could you share your **experience** using the tool now compare to the conventional approach used before?
3) What is your opinion about level of motivation in learning how to program so far? Compare with previous experience learning the subject.

4) As non-computing background user, **what do you think** about **learning** Flash Action Script now? You may describe your answer from different perspectives such as academic, psychology or others.
5) Do you feel that you can become a developer or **creative designer** in terms of creating your own teaching module after learn this lesson?

    a. Why?
    b. How?

6) If you encounter with any difficulties in any parts of the lesson, how you counter with the condition? For instance when you get puzzled (stuck) to certain task? (*Hint: stop it or use the tool to solve the problem)*
7) If you are to be asked to create more than one scene with interesting interactivity activities in the module using the tool as the guideline, do you think that you can work on it by yourself now?
8) Do you consider that learning to program, not only using Flash Action Script will increase your logical thinking skill? How (you can relate Flash and Anime software that you had used before)?
9) What is the most challenging part in learning or teaching programming?
10) What is the easiest to learn programming?
11) How would you describe the process of learning to program to someone who doesn't program?
12) What do you think the benefit of learning programming?
13) To sum up, what do you hope to accomplish when going to Malaysia?

Section B

This section will view on the techniques used in teaching and learning process through the concept given.

If you have **experience in teaching**, it would be highly appreciated if you can fill this section as well. Thank you.

1) As a teacher, what do you think about writing a set of rules to achieve a desired effect as an important thing that should be considered first in your class?

2) How do you think that creating set of rules can give a good, though not always be the best solutions. Fox example in teaching Mathematics for primary students, how will be the approach you think the best for them?

3) From your experience in teaching or If you are a trainee-teacher, did you or will you practise the scenario as stated below in class?

Breaking down things (whether objects, problem, processes, solution, systems or abstractions) into component or basic parts that make them easier to work with.

How do you think this approach will help your students?

4) In teaching, is it necessary to reduce complexity by removing unnecessary details? How do you think this will help both parties (teacher and student)?

5) What is easiest thing to learn programming? Is there an easy concept to learn if the users join the course (1-2 months) with no programming experience?

6) What is the difficult section in learning about programming?

7) Do you view programming as the difficult subject to be learnt today? Why?

8) Do you think that computer science – flash action script programming (to be specific such like SWITCH CASE function) influenced by logical thinking? Could you please discuss more about this?

THANK YOU FOR YOUR COOPERATION

APPENDIX IX

Prototype Design

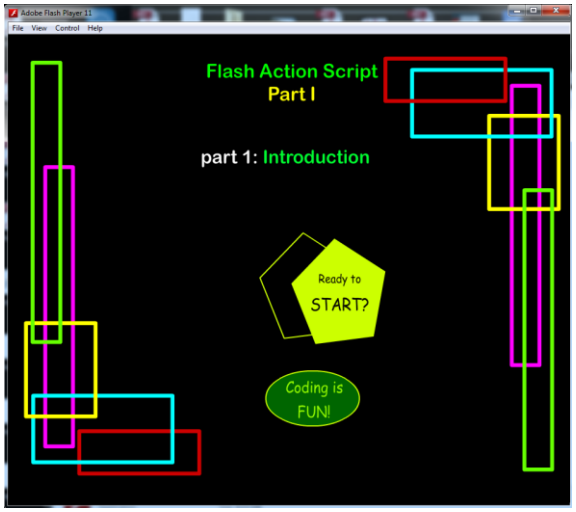


Figure 1: Main Screen

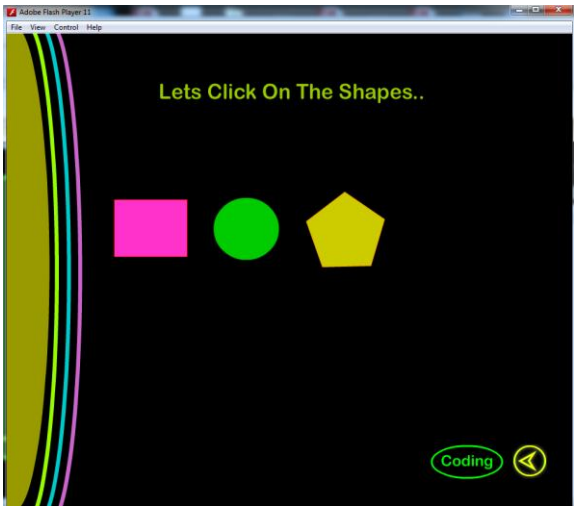


Figure 2: Switch Case Screen (After "Coding is FUN!" button is clicked

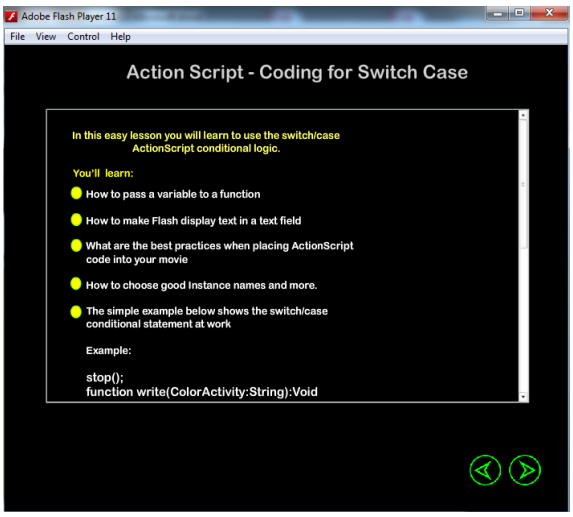


Figure 3: Scripts on Switch-Case Statement

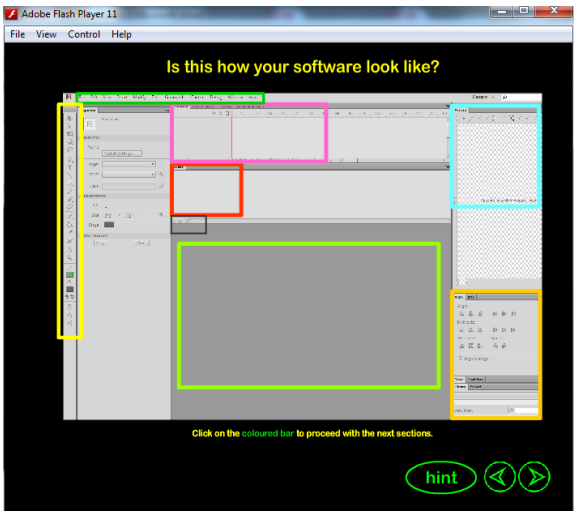


Figure 4: Interface Screen



Figure 5: Lesson Task Screen

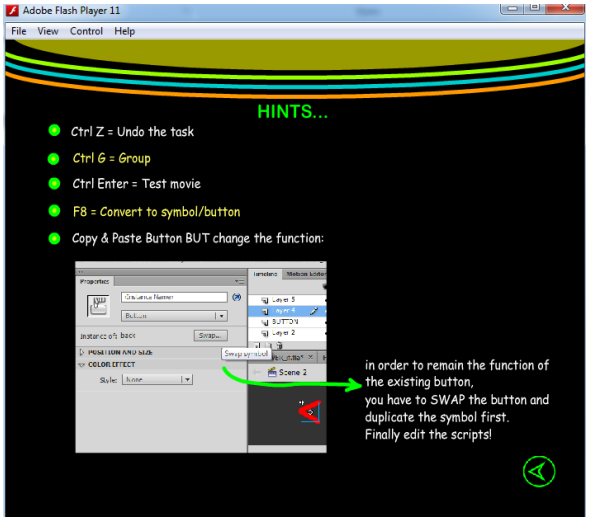


Figure 6: Hints Screen

Improved version of screens after prototype had been evaluated

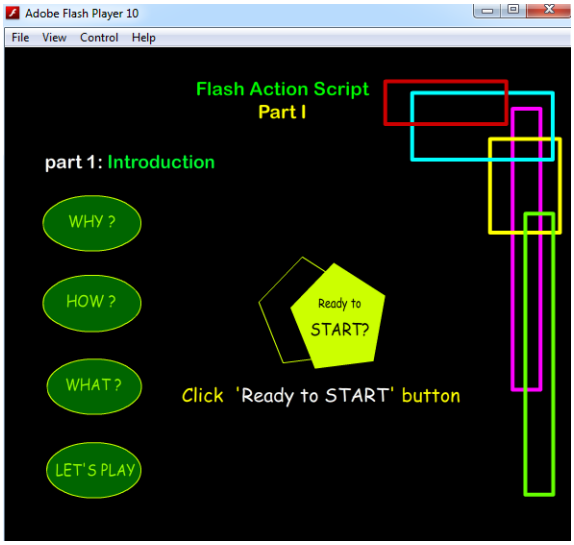


Figure 5: Main Screen

The screen had be suggested to include a brief explanation on the function of learning this tool. Four additional buttons were added as above.

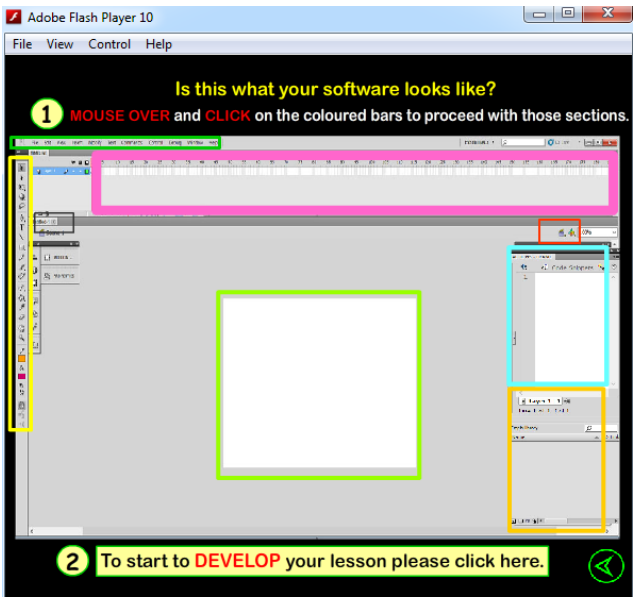


Figure 6: Interface screen

Interface Screen was asked to add few instructions so that user should not supposed to think how and where they should start with. Some important instructions added here.

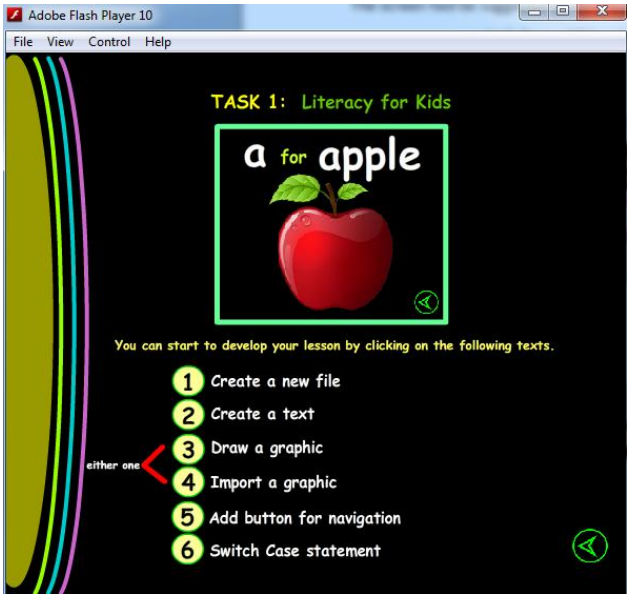


Figure 7: Lesson Task screen was advised to improve in terms of steps by steps instruction to do the task so that it motivates user to work for it. Previously, the screen did not show the steps thus, user will confuse on the screen design.

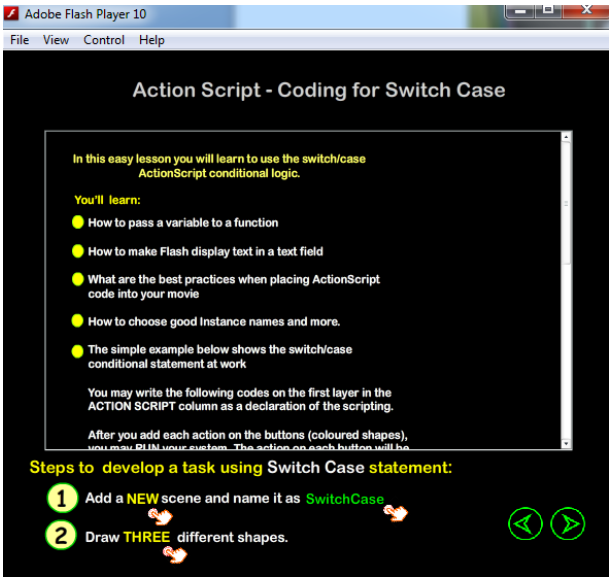


Figure 8: Switch case statement screen was designed inaccurate way for user. Novice user will not understand the complexity of coding if the instructions were not clearly defined.

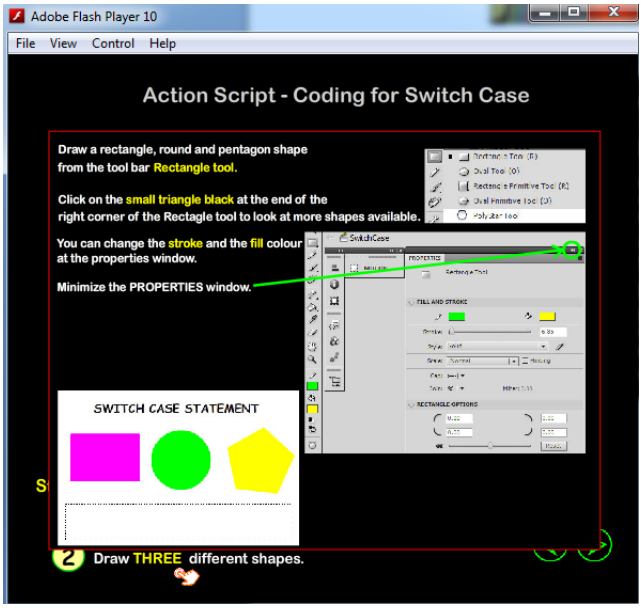


Figure 9: Additional instruction on Switch Case statements added.

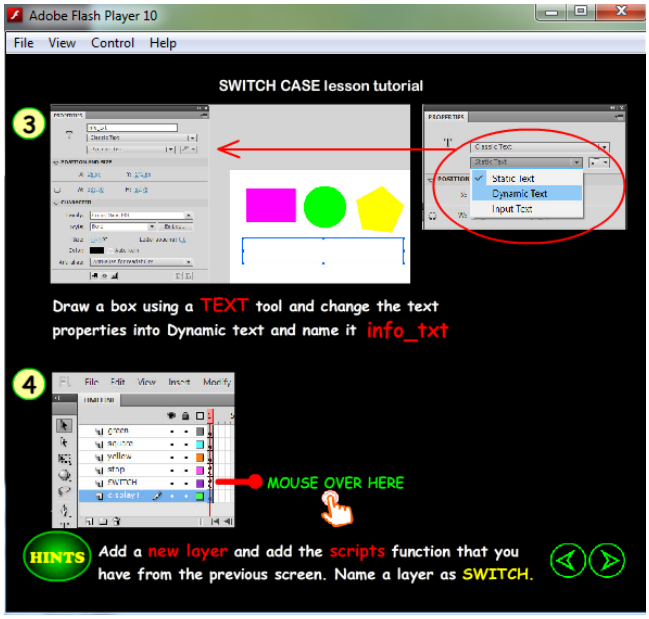


Figure 10: Additional instruction on Switch Case statements added.

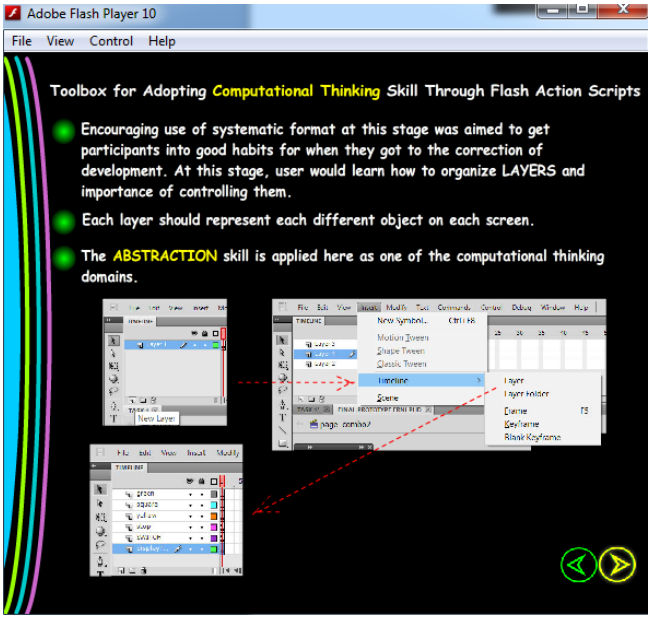


Figure 11: Additional instruction on Switch Case statements added.

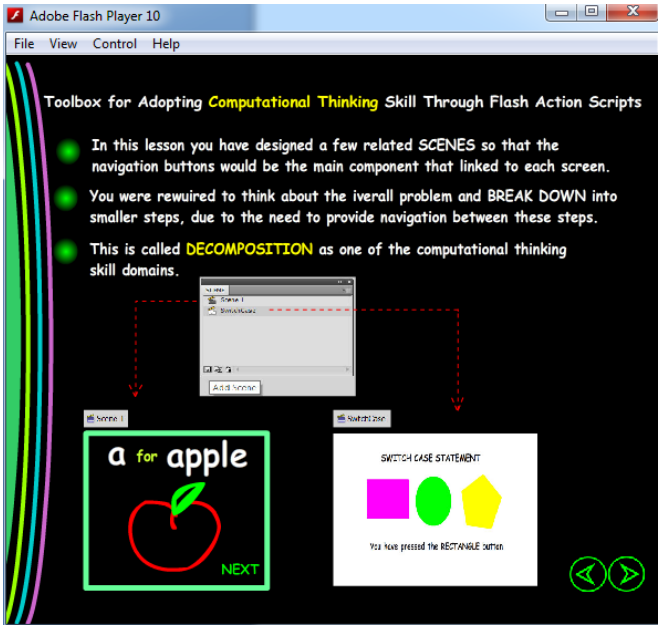


Figure 12: Additional instruction on Switch Case statements added.

APPENDIX X

Pilot Study - Pre- Study 1

ERNI MARLINA SAARI (PhD. Candidate)

School of Computer Science, Jubilee Campus.UK

PILOT STUDY - Pre Study 1

Please answer the questions provided. Your cooperation is highly appreciated. Thank you.

Name:

School/Faculty:

Course :

Section A

1) Could you please explain about your academic background?

2) Do you think learning computing is important? Why?

3) Do you know that this course is normally attended by those with computing background?

4) Why do you take this course?

5) Do you have any experience(s) using multimedia software? Please explain.

6) Why do you think, this course is suitable to be taken by any non-computing background students?

7) You are pre-service teacher, please explain why teachers need to be equipped themselves with technology?

8) How does the multimedia skill can facilitate learning?

9) Do you agree that besides multimedia, programming is an important element to be learned as well? Why?

10) Do you have experience using Flash?

11) Why do you think Flash is an important subject to be explored?

12) Please give your opinion about Flash.

Section B

Choose any **5** from the lists that give motivation to learn the Flash Action script.

- Because you think it will be easy.
- Because you have been told you have to do it.
- To learn how to program.
- Because you wish to become a professional programmer.
- Because you would like to write programs for your own use.
- Because an understanding of programming will help in your future career.
- To understand more about programs and programming.
- To improve your general ability to solve problems.
- To learn general study and transferable skills.
- To get the module credits