



This work is protected by copyright and other intellectual property rights and duplication or sale of all or part is not permitted, except that material may be duplicated by you for research, private study, criticism/review or educational purposes. Electronic or print copies are for your own personal, non-commercial use and shall not be passed to any other individual. No quotation may be published without proper acknowledgement. For any other use, or to quote extensively from the work, permission must be obtained from the copyright holder/s.

**An empirical investigation into the
effectiveness of a robot simulator as a tool
to support the learning of introductory
programming**

by

LOUIS MAJOR

A thesis submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

KEELE UNIVERSITY

March 2014

Abstract

Background: Robots have been used in the past as tools to aid the teaching of programming. There is limited evidence, however, about the effectiveness of simulated robots for this purpose.

Aim: To investigate the effectiveness of a robot simulator, as a tool to support the learning of introductory programming, by undertaking empirical research involving a range of participants.

Method: After the completion of a Systematic Literature Review, and exploratory research involving 33 participants, a multi-case case study was undertaken. A robot simulator was developed and it was subsequently used to run four 10-hour programming workshops. Participants included students aged 16 to 18 years old (*n.* 23) and trainee teachers (*n.* 23). Three in-service teachers (*n.* 3) also took part. Effectiveness was determined by considering participants' opinions, attitudes and motivation using the simulator in addition to an analysis of the students' programming performance. Pre- and post-questionnaires, in- and post-workshop programming exercises, interviews and observations were used to collect data.

Results: Participants enjoyed learning using the simulator and believed the approach to be valuable and engaging. Whilst several factors must be taken into consideration, the programming performance of students indicates that the simulator aids learning as most completed tasks to a satisfactory standard. The majority of trainee teachers, who had learned programming beforehand, believed that the simulator offered a more effective means of introducing the subject compared to their previous experience. In-service teachers were of the opinion that a simulator offers a valuable means for supporting the teaching of programming.

Conclusion: Evidence suggests that a robot simulator can offer an effective means of introducing programming concepts to novices. Recommendations and suggestions for future research are presented based on the lessons learned. It is intended that these will help to guide the development and use of robot simulators in order to teach programming.

Acknowledgements

There are many people who have helped me to reach this stage of my PhD research. I owe thanks to them all. The support of my two supervisors, Theocharis Kyriacou and Pearl Brereton, must first be acknowledged. Theo, your door was always open and I appreciate your efforts and advice. Likewise Pearl whose knowledge, experience and support I have benefited from immensely. As a supervisory team you complimented each other perfectly and I have learned so much from you both. All that you have taught me will stand me in good stead during my future career.

Thanks are also due to all of the others working in the School of Computing and Mathematics who have helped me in some way. From technical support to administrative advice (and the occasional tea bag), without your assistance everything would have been much more difficult.

I am also grateful for the financial support provided by the Research Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM). Generous grants by The Roger and Sarah Bancroft Clark Charitable Trust, Sir Richard Stapley Educational Trust and Keele Postgraduate Association must also be acknowledged. These awards enabled me to concentrate on my research, to present at several fantastic conferences (both in the UK and overseas) and to make valuable networking connections with likeminded researchers.

Those people who belong to the scholarly community, who have offered advice and pointers at various stages of my PhD, must be thanked. I am grateful to all those who reviewed protocols, early drafts and articles submitted for peer-review. Your efforts are appreciated and your feedback significantly helped.

Gratitude is also owed to the staff and students whom made this research possible. Participants from Great Wyrley High School (Cannock), New College (Telford) and Keele University were all involved. Thank you for taking the time to participate and for being brilliant to work with.

Thanks are also due to my fellow postgraduates/office-mates, past and present, at Keele University. Each of you contributed to creating a stimulating academic and social environment. Seeing Siffat

Ullah Khan, John Butcher, James Rooney and Ryad Soobhany successfully complete their PhD provided great inspiration. I also greatly enjoyed offering advice to newer members of the postgraduate office, Chris Marshall and Adam Wootton. I wish you all the best for the future but I know we will keep in touch.

I am also incredibly grateful to Sue Swaffield of the University of Cambridge. Sue, you have been incredibly understanding whilst I have been preparing this thesis for submission. I am looking forward to working with you on the exciting projects we have got planned.

Lastly, I would like to thank all my family (whom there are too many individuals to mention) for their love and encouragement. In particular, I want to thank my Mum (Giovanna) and Dad (Chris) for supporting me in everything I have ever chose to do. You have always been so selfless and only ever want for me to be happy. You are the most incredible parents – I am so, so grateful to have you. My Nan and Grandad (Pauline and Graham) and sister Nina have similarly always been there for me – thank you.

And Vikki, my fiancée, well what can I say? I cannot wait for our wedding next year and for us to spend the rest of our lives together. You have helped me more than you know and, without doubt, I appreciate how lucky I am to have found you. I am also grateful to Roz, Rod, Iris and the other members of your family for everything they have done for us.

My final and biggest thanks, however, are reserved for my Nan, Giovanna, who passed away whilst I was in the final stages of writing this thesis. I think about you every day and am so glad that the flexibility of the writing process allowed me to spend so much time with you in your final weeks. Without doubt you are the most amazing and inspirational woman I have ever met. You will never be forgotten.

Author's Declaration

During the PhD work was presented at a number of conferences. Details of several papers that were prepared for publication, along with other related activities, are provided in this section.

Publications

Major, L., Kyriacou, T. and Brereton, O.P. (2012). Teaching Novices Programming Using a Robot Simulator: Case Study Protocol. In *Proceedings of the 24th Psychology of Programming Interest Group PPIG 2012* (pp. 93-104). London Metropolitan University, UK, PPIG.

Major, L., Kyriacou, T., & Brereton, O. P. (2012). Systematic literature review: teaching novices programming using robots. *IET Software*, 6(6), 502-513.

Major, L., Kyriacou, T., & Brereton, O. P. (2011). Experiences of prospective high school teachers using a programming teaching tool. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research* (pp. 126-131). ACM.

Major, L., Kyriacou, T., & Brereton, O. P. (2011). Systematic literature review: Teaching novices programming using robots. In *Proceedings of Evaluation & Assessment in Software Engineering EASE 2011*, (pp. 21-30). IET.

Conference and Seminar Activity

External Talks

- Psychology of Programming Interest Group Annual Conference (PPIG '12), London, UK, November 2012.
- British Educational Research Association Early Career Researcher Conference (BERA-ECR '12), Manchester, UK, September 2012.

- Koli Calling International Conference on Computing Education Research (Koli Calling '11), Koli National Park, Finland, November 2011.
- Psychology of Programming Interest Group Doctorial Consortium, York, UK, September 2011.
- Evaluation and Assessment in Software Engineering Conference (EASE '11), Durham, UK, April 2011.
- International Association of Technology, Education and Development Conference (IATED '11), Valencia, Spain, March 2011.

Internal Talks

- Computing Postgraduate Research Day 2013, Keele University, January 2013.
- EPSAM Fest 2012 (Poster Presentation), Keele University, July 2012.
- Computing Postgraduate Research Day 2011, Keele University, June 2011.
- EPSAM Fest 2010 (Poster Presentation), Keele University, September 2010.

Grants Received

For the Support of Research

- £500 – The Roger and Sarah Bancroft Clark Charitable Trust (2012)
- £500 – Sir Richard Stapley Educational Trust (2011)
- £140 – Keele Postgraduate Association Bursary (2012)

Other

- £200 – British Educational Research Association (BERA):
Prize for Early Career Researcher Presentation at the BERA 2012 conference

Contents

Acknowledgements	iii
Author's Declaration	v
Contents	vii
List of Tables	xii
List of Figures	xv
Chapter One - Introduction	1
1.1 Background	2
1.2 Research Purpose	7
1.2.1 Aim and Objectives	7
1.2.2 Research Question	8
1.3 Original Contributions	10
1.4 Thesis Outline	11
Chapter Two – Literature Review	14
2.1 Introduction to the SLR	15
2.1.1 Preparation for the SLR	15
2.2 Method	16
2.2.1 Research Questions	16
2.2.2 Search Process	17
2.2.3 Inclusion and Exclusion Criteria	19
2.2.4 Quality Assessment	20
2.2.5 Data Extraction	21
2.3 Results	22
2.3.1 Search Results	22
2.3.2 Quality Assessment	23
2.3.3 Research Questions	25
2.3.4 Studies Related to Simulated Robots Identified During the SLR	30
2.3.5 Limitations of the SLR	31
2.4 Supplementary Thesis Literature Update	32
2.5 Discussion	35
2.5.1 The Findings of the SLR	35
2.5.2 Rigour of the SLR and Identification of Prominent Sources	37
2.5.3 The Potential to Investigate the Effectiveness of Simulated Robots	38
2.6 Summary	41
Chapter Three – Exploratory Studies	42
3.1 Introduction	43
3.2 Method	44
3.2.1 Overall Approach	45
3.2.2 Data Sources	52
3.2.3 Workshop Procedure and Setting	54

3.2.4	Data Collection and Analysis	56
3.2.5	Limitations of the Exploratory Studies	56
3.3	Exploratory Workshop One (EW1)	58
3.3.1	Study Execution	58
3.3.2	Results - Exploratory Questionnaire One (EQ1)	58
3.3.3	Results - Exploratory Questionnaire Two (EQ2)	62
3.4	Exploratory Workshop Two (EW2)	66
3.4.1	Study Execution	66
3.4.2	Results - Exploratory Questionnaire One (EQ1)	66
3.4.3	Results - Exploratory Questionnaire Two (EQ2)	69
3.5	Discussion	72
3.6	Summary	78

Chapter Four – The Kobot Robot Simulator 79

4.1	Introduction	80
4.2	Modifying the Pre-Existing Robot Simulator	81
4.2.1	Advantages of modifying the PERS	81
4.2.2	Modifications made to the PERS	82
4.3	Kobot: A Robot Simulator for Supporting the Learning of Introductory Programming	84
4.4	Deciding What Concepts to Teach Using Kobot	88
4.4.1	ACM/IEEE Computer Science Curricula 2008	88
4.4.2	Consideration of other curriculum guidance	92
4.4.3	How did selected programming concepts influence the design of Kobot?	94
4.5	Summary	94

Chapter Five – Case Study Methodology 95

5.1	An Introduction to the Case Study Methodology	96
5.2	Case Study Design	97
5.2.1	Aim	97
5.2.2	Propositions	98
5.2.3	Data Sources	99
5.2.4	The suitability and relevance of the case study methodology for this research	99
5.2.5	Consideration of other potentially relevant research methods	100
5.2.6	The Ethical Approval Process	103
5.3	Verification of the Case Study Approach	104
5.4	Workshop Content	106
5.4.1	The Influence of Pedagogy on the Workshop Design	108
5.4.2	Basic syntax and semantics of a higher-level language	110
5.4.3	Variables and Constants	114
5.4.4	Logical Expressions	115
5.4.5	Counting, JOptionPane and Nested Statements	119
5.4.6	Introducing the Remaining Data Types	121
5.4.7	While and Do While Loops	124
5.4.8	Method Creation	126
5.4.9	Conditional and Iterative Control Structures	127
5.4.10	For Loops	132

5.4.11 Arrays	133
5.5 Study Limitations	135
5.6 Summary	139

Chapter Six – Case One: Trainee Teachers 140

6.1 Introduction	141
6.2 Study Execution	142
6.2.1 Participants	142
6.2.2 Workshop Procedure and Setting	143
6.2.3 Data Collection Strategy	143
6.2.4 Data Analysis Strategy	149
6.3 Trainee Teacher Workshop One (TTW1)	150
6.3.1 Data Collection Instruments – Programming Exercises	150
6.3.2 Data Analysis Strategy – Programming Exercises	150
6.3.3 Results - TTW1 Trainee Teacher Questionnaire One (TTQ1)	152
6.3.4 Results - TTW1 Trainee Teacher Questionnaire Two (TTQ2)	156
6.3.5 Results - TTW1 Workshop Log (Day One Overview)	160
6.3.6 Results - TTW1 Workshop Log (Day Two Overview)	161
6.3.7 Analysis of TTW1 Participants' Programming Performance	162
6.4 Trainee Teacher Workshop One (TTW2)	164
6.4.1 Data Collection Instruments – Programming Exercises	164
6.4.2 Data Analysis Strategy – Programming Exercises	165
6.4.3 Results – TTW2 Trainee Teacher Questionnaire One (TTQ1)	166
6.4.4 Results – TTW2 Trainee Teacher Questionnaire Two (TTQ2)	169
6.4.5 Results – TTW2 Workshop Log (Day One Overview)	172
6.4.6 Results – TTW2 Workshop Log (Day Two Overview)	172
6.4.7 Analysis of TTW2 Participants' Programming Performance	173
6.5 Lessons Learned	174
6.6 Summary	176

Chapter Seven – Case Two: Students 177

7.1 Introduction	178
7.2 Study Execution	179
7.2.1 Participants	179
7.2.2 Workshop Procedure and Setting	180
7.2.3 Data Collection Strategy	180
7.2.4 Data Analysis Strategy	186
7.3 Results: Student Workshop One (SW1)	189
7.3.1 Student Questionnaire One (SQ1)	189
7.3.2 Student Questionnaire Two (SQ2)	190
7.3.3 SW1 Workshop Log – Day One Overview	194
7.3.4 SW1 Workshop Log – Day Two Overview	195
7.3.5 In-Workshop Programming Exercise One (IWE1)	196
7.3.6 In-Workshop Programming Exercise Two (IWE2)	197
7.3.7 Combined Performance on IWE1 and IWE2	198
7.3.8 Post-Workshop Exercise (PWE)	198
7.4 Results: Student Workshop Two (SW2)	199

7.4.1	Student Questionnaire One (SQ1)	199
7.4.2	Student Questionnaire Two (SQ2)	202
7.4.3	SW2 Workshop Log – Day One Overview	206
7.4.4	SW2 Workshop Log – Day Two Overview	206
7.4.5	In-Workshop Programming Exercise One (IWE1)	207
7.4.6	In-Workshop Programming Exercise Two (IWE2)	208
7.4.7	Combined Performance on IWE1 and IWE2	209
7.4.8	Post-Workshop Exercise (PWE)	209
7.5	Combined Analysis of Programming Performance	210
7.5.1	In-Workshop Programming Exercise One (IWE1)	210
7.5.2	In-Workshop Programming Exercise Two (IWE2)	211
7.5.3	Post-Workshop Programming Exercise (PWE)	212
7.6	Reliability and Validity of Programming Exercises	213
7.6.1	Reliability of the Post-Workshop Programming Exercises (PWE)	213
7.6.2	Validity of the Programming Tests	215
7.7	Additional Data Source: Teacher Interviews	216
7.7.1	Theme One – The Kobot Robot Simulator	217
7.7.2	Theme Two – Student Participants	220
7.7.3	Theme Three – Miscellaneous	224
7.8	Summary	226

Chapter Eight – Discussion 227

8.1	Introduction	228
8.2	The Case Study Propositions	230
8.2.1	Proposition One: A robot simulator is an effective tool for supporting the learning of introductory programming	230
8.2.2	Proposition Two: A robot simulator improves novices' perceptions of programming	238
8.2.3	Proposition Three: A robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience	239
8.2.4	Proposition Four: A robot simulator improves trainee ICT/Computer Science (CS) teachers' confidence in their ability to teach introductory programming	241
8.3	General Discussion of Findings	243
8.3.1	Comparison of the research findings to previously completed work	243
8.3.2	Consideration of contradictory evidence and unexpected findings	246
8.4	Reflection on the Research Undertaken	248
8.5	Guidance for Related Future Work	252
8.6	Rival Explanations and Threats to Validity	256
8.6.1	Rival Explanations	256
8.6.2	Threats to Validity	258
8.7	Summary	268

Chapter Nine – Summary and Conclusion 269

9.1	Summary and Conclusion	270
9.2	The importance, meaning and contribution of this research	273

References	275
Appendix A1 – Articles Included in the SLR	288
Appendix A2 – Electronic Search Results	292
Appendix A3 – Exploratory Questionnaires	293
Appendix A4 – Ethical Approval Documentation	295
Appendix A5 – Trainee Teacher (Case One) Questionnaires	296
Appendix A6 – Trainee Teacher (Case One) Workshop Log Transcripts	298
Appendix A7 – Programming Exercises Used During TTW1 Only	306
Appendix A8 – Programming Performance During TTW1	309
Appendix A9 – Programming Exercises	317
Appendix A10 – Programming Performance During TTW2	320
Appendix A11 – Student Questionnaires	323
Appendix A12 – Student Workshop Log Transcripts	325
Appendix A13 – Teacher Interview Transcripts	333
Appendix A14 – Kobot Control Methods	342
Appendix A15 – Case Study Design Checklist	344

List of Tables

Table 2-1. Results of the SLR Quality Assessment.	23
Table 2-2. Data extracted from studies included in the SLR.	25
Table 3-1. Programming languages EW1 participants had experience using arranged according to their self-perceived knowledge of each.	59
Table 3-2. Programming concepts arranged by the number of EW1 participants who stated they had used each concept in their past code.	60
Table 3-3. EW1 participants' experience of learning programming.	60
Table 3-4. EW1 participants' opinions on teaching high school students programming (pre-EW1).	61
Table 3-5. EW1 participants views on the difficulty of the workshop tasks and whether they would use the PERS in the future.	63
Table 3-6. EW1 participants' opinions on the effectiveness of elements of the workshop (arranged by the number of participants who selected each option).	63
Table 3-7. EW1 participants' opinions on teaching high school students programming (post-EW1).	64
Table 3-8. EW2 participants' views on their past programming experience.	67
Table 3-9. Programming concepts arranged by the number of EW2 participants who stated that they had used each concept in their past code.	68
Table 3-10. EW2 participants' opinions on teaching high school students programming (pre-EW2).	68
Table 3-11. EW2 participants' enjoyment, and difficulty, of their programming experience during the workshop.	69
Table 3-12. EW2 participants' opinions on whether such a robot simulator offers an effective method of introducing programming concepts to novices.	70
Table 3-13. EW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.	70
Table 3-14. EW2 participants' opinions on teaching high school students programming (post-EW2).	70
Table 4-1. Main modifications made to the PERS in addition to justification for them.	83
Table 4-2. Fundamental programming constructs identified by the ACM/IEEE.	89
Table 4-3. Learning objectives for the fundamental programming constructs identified by the ACM/IEEE.	89
Table 4-4. Outlining the similarities between the fundamental programming concepts identified by the ACM/IEEE and Flot et al.	93
Table 5-1. The workshop format.	107
Table 5-2. Details of previous research that influenced how basic Java syntax and semantics were introduced during the workshop.	110
Table 5-3. Details of previous research that influenced how variables and constants were initially introduced during the workshop.	114
Table 5-4. Details of previous research that influenced how the remaining data types were introduced during the workshop.	121
Table 5-5. Details of previous research that influenced how While and Do While Loops were introduced during the workshop.	124
Table 5-6. Details of previous research that influenced how the creation of Methods was introduced during the workshop.	126

Table 5-7. Robot control methods used extensively during Day Two of the workshop.	127
Table 6-1. Programming languages TTW1 participants had experience using arranged according to participants self-perceived knowledge of each.	152
Table 6-2. Programming language most recently learned by TTW1 participants.	153
Table 6-3. TTW1 participants' experience of learning their most recent programming language.	153
Table 6-4. Programming concepts arranged by the number of TTW1 participants who stated that they had used each concept in their past code.	154
Table 6-5. Issues identified by TTW1 participants in regards to the learning of programming.	154
Table 6-6. Stereotypes identified by TTW1 participants in regards to the learning of programming	155
Table 6-7. TTW1 participants' opinions on teaching high school students programming (pre-TTW1).	155
Table 6-8. TTW1 participants' opinions of their workshop experience and effectiveness of the Kobot simulator.	156
Table 6-9. Aspects of Kobot liked by TTW1 participants.	157
Table 6-10. Aspects of Kobot disliked by TTW1 participants.	157
Table 6-11. TTW1 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.	158
Table 6-12. Comparison of TTW1 participants' previous introductory programming experience to the one using Kobot.	158
Table 6-13. Comparison of TTW1 participants' previous introductory programming experience to the one using Kobot (participants who most recently learned Java only).	158
Table 6-14. TTW1 participants' opinions on teaching high school students programming (post-TTW1).	159
Table 6-15. Programming languages TTW2 participants had experience using along with their self-identified knowledge of each.	166
Table 6-16. Information relating to TTW2 participants' most recent programming learning experience.	167
Table 6-17. Information relating to TTW2 participants' most recent programming learning experience.	167
Table 6-18. TTW2 participants' enjoyment of their past programming experience.	168
Table 6-19. TTW2 participants' opinions on teaching high school students programming (pre-TTW2).	168
Table 6-20. TTW2 participants' opinions of their workshop experience and effectiveness of the Kobot simulator.	169
Table 6-21. Aspects of Kobot liked and disliked by TTW2 participants.	170
Table 6-22. TTW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.	170
Table 6-23. Comparison of TTW2 participants' previous programming experience to the one using Kobot.	171
Table 6-24. TTW2 participants' opinions on teaching high school students programming (post-TTW2).	171
Table 7-1. Programming languages SW1 participants had experience using along with their self-identified knowledge of each.	189
Table 7-2. Information relating to SW1 participants past programming learning experience.	189
Table 7-3. Opinions of SW1 participants on the learning of programming and the teaching of programming in high schools (pre-SW1).	190

Table 7-4. SW1 participants' opinions on their workshop experience and use of Kobot.	190
Table 7-5. SW1 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.	191
Table 7-6. SW1 participants' views on the effect of Kobot upon their opinions of programming.	191
Table 7-7. Aspects of Kobot liked and disliked by SW1 participants.	192
Table 7-8. Opinions of SW1 participants on the learning of programming and the teaching of programming in high schools (post-SW1).	193
Table 7-9. Breakdown of SW1 participants' performance on IWE1.	196
Table 7-10. Breakdown of SW1 participants' performance on IWE2.	197
Table 7-11. Combined means scores of SW1 participants on the in-workshop exercises.	198
Table 7-12. Breakdown of SW1 participants' performance on the PWE.	198
Table 7-13. Programming languages SW2 participants had experience using along with their self-identified knowledge of each.	199
Table 7-14. Information relating to SW2 participants past programming learning experience.	200
Table 7-15. Opinions of SW2 participants on the learning of programming and the teaching of programming in high schools (pre-SW2).	201
Table 7-16. SW2 participants' opinions on their workshop experience and use of Kobot.	202
Table 7-17. SW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.	202
Table 7-18. Aspects of Kobot liked and disliked by SW2 participants.	203
Table 7-19. SW2 participants' views on the effect of Kobot upon their opinions of programming.	204
Table 7-20. Opinions of SW2 participants on the learning of programming and the teaching of programming in high schools (post-SW2).	204
Table 7-21. Breakdown of SW2 participants' performance on IWE1.	207
Table 7-22. Breakdown of SW2 participants' performance on IWE2.	208
Table 7-23. Combined means scores of SW2 participants on the in-workshop exercises.	209
Table 7-24. Breakdown of SW2 participants' performance on the PWE.	209
Table 7-25. Breakdown of SW1 and SW2 performance on the PWE.	212
Table 7-26. Breakdown of the relationship between the in-workshop exercises and the programming fundamentals taught.	215
Table 7-27. Breakdown of the relationship between the post-workshop exercise and the programming fundamentals taught.	215
Table 8-1. Relevant learning principles identified by Gee (Gee, 2003) which can be used to further demonstrate the effectiveness of a robot simulator to support the learning of programming.	245
Table 8-2. Recommendations to support the future development and use of robot simulators in an introductory programming context.	253

List of Figures

Figure 1-1. Summary of Thesis Content.	13
Figure 2-1. Publication year of studies included in the SLR.	22
Figure 2-2. Programming languages used during studies included in the SLR.	26
Figure 2-3. Effectiveness of robots as tools to teach programming.	28
Figure 2-4. Breakdown of effectiveness of robots as tools to teach programming (by nature of robot reported on).	29
Figure 2-5. The effectiveness of robots as tools to teach programming after excluding studies awarded a quality score of five or less.	29
Figure 3-1. A photograph of a Mark III robot after which the Pre-Existing Robot Simulator is modelled.	46
Figure 3-2. Mark III Robot Topology.	46
Figure 3-3. The Pre-Existing Robot Simulator class diagram when loaded in the BlueJ IDE.	47
Figure 3-4. Annotated Screenshot of the Pre-Existing Robot Simulator GUI.	48
Figure 3-5. Screenshot of the Pre-Existing Robot Simulator with a simulated agent, and examples of 2D and 3D objects, visible in the arena.	50
Figure 3-6. Example of some of the challenges completed by participants.	55
Figure 3-7. One solution to the 'avoiding behaviour' task set during the workshop (with simple explanation).	55
Figure 3-8. The views of EW1 participants on whether such a robot simulator offers an effective method of introducing programming concepts to novices.	62
Figure 3-9. EW1 participants' enjoyment of the programming experience using the PERS.	62
Figure 4-1. The Kobot Class Diagram when loaded in the BlueJ IDE.	84
Figure 4-2. Annotated screenshot of the Kobot robot simulator.	85
Figure 4-3. A screenshot of the code editor.	85
Figure 5-1. The multiple-case case study design.	98
Figure 5-2. Screenshot of a robotic agent completing the 'Square Drawer' task.	112
Figure 5-3. Sample code completed during the 'Square Drawer' task before and after the introduction of parameters.	113
Figure 5-4. The solution to the 'Line Tracer' task.	113
Figure 5-5. A screenshot of the 'Line Tracer' task.	114
Figure 5-6. Screenshot of the 'Information Panel' available as an option within the Kobot simulator.	115
Figure 5-7. The programming code which comes preloaded in the Berners robot class.	116
Figure 5-8. Screenshot of a robotic agent completing one of several 'Pauser Robot' tasks.	117
Figure 5-9. Example code produced during the 'Pauser' workshop tasks (with explanation).	118
Figure 5-10. Screenshot of the 'Shapes Arena' task.	118
Figure 5-11. Screenshot of a robotic agent completing the 'Line Counter' task.	120
Figure 5-12. Example code produced during tasks related to Boolean variables and printing to the Terminal Window (with explanation)	121
Figure 5-13. Annotated Screenshot of the Kobot Helper Program GUI.	122
Figure 5-14. Example code produced during one of the Do While Loop tasks.	125
Figure 5-15. Example screenshots of robotic agents completing the 'Robot Dance' task.	127

Figure 5-16. A robotic agent performing a 'Figure of 8' - one of the first challenges undertaken during Day Two of the workshop (along with one potential solution).	128
Figure 5-17. Screenshot showing a robotic agent with a 3D object drawn in the arena. The values of the agent's proximity sensors are shown printed to the Terminal Window.	129
Figure 5-18. Example code that demonstrates how the proximity sensors are used in the Kobot simulator.	130
Figure 5-19. Example code that demonstrates one solution to the 'Curious' Behaviour robot programming task.	131
Figure 5-20. Screenshots of robotic agents exhibiting 'seeking' (Left) and 'avoiding' (Right) behaviour.	131
Figure 5-21. Example code that demonstrates a solution to the 'Black Hole' task.	132
Figure 5-22. Screenshot of a robotic agent completing the 'Black Hole' task.	133
Figure 5-23. Screenshot of a robotic agent completing the 'Barcode Reader' task.	135
Figure 6-1. Data collection instruments used during Case One: Trainee Teachers	143
Figure 6-2. Screenshot of the Page robotic agent used during the post-workshop programming exercises.	148
Figure 7-1. Data collection instruments used during Case Two: Students	180
Figure 7-2. Combined scores of all SW1 participants on IWE1.	196
Figure 7-3. Combined scores of all SW1 participants on IWE2.	197
Figure 7-4. Combined scores of all SW2 participants on IWE1.	207
Figure 7-5. Combined scores of all SW2 participants on IWE2.	208
Figure 7-6. SW1 and SW2 participant performance on IWE1.	210
Figure 7-7. SW1 and SW2 participant performance on IWE2.	211
Figure 7-8. Combined performance of SW1 and SW2 participants on the PWE.	212
Figure 7-9. Themes identified during the teacher interviews.	216

Chapter One

Introduction

This chapter introduces the main focus of the thesis, specifically an investigation into the effectiveness of simulated robots as introductory programming teaching tools. The primary research question that is examined throughout – is a robot simulator an effective tool for supporting the learning of introductory programming – is outlined and the motivation and objectives for the work explained. The novelty of the thesis, and how it contributes to knowledge, is also identified. Finally, the structure of the thesis is presented.

1.1 Background

We live in an information age and computers have become indispensable in our daily lives (Song & Yu, 2011). This dependence has resulted in an increase in demand for computer scientists, which is expected to increase irrespective of the poor current state of the economy (Wellman *et al.*, 2009). In particular, there is a need to produce competent computer programmers (Robins *et al.*, 2003). After all, it is not just PCs and PDAs that need to be programmed, but also washing machines, microwaves and a range of other ‘essential’ items. There is currently a problem, however, attracting students to (and maintaining their interest in) the study of programming (Bergin, 2006).

Traditional methods of teaching programming fail to excite a large proportion of learners and the subject is often perceived to be difficult and laborious (Esteves *et al.*, 2008). Negative opinions which novice programmers may hold are often realised when their first programming challenge is to complete an unimaginative task such as the “Hello World!” program (Talbot, 2000) and when subsequent assignments continue in the same style. Student interest may be lost due to programming being considered as dull (Kelleher & Pausch, 2007). Lacklustre initial interactions with programming also lead to high course withdrawal rates and poor academic performance (Kinnunen & Malmi, 2006). Indeed, it takes few negative experiences in the early stages to disillusion a novice programmer (Dunican, 2002).

Other issues are associated with the teaching of programming. Learning to program can be hard and novices may suffer from a range of difficulties and deficits (Robins *et al.*, 2003). Programming also requires a strong understanding of abstract concepts (Lahtinen *et al.*, 2005) and a variety of misconceptions, which many newcomers to the subject hold, must be overcome (Lahtinen & Ahoniemi, 2009). In regards to today’s “MTV or Nintendo generation” of students, whose perception of technology and media has been profoundly influenced by these sources, the outdated views held by a large proportion of computer science educators does little to engage or foster deep learning (Guzdial & Soloway, 2002). The abstract and text based world that many computer science teachers grew up with themselves does not appeal to the learners of today.

Various types of interventions have been used to help students develop programming skills and to overcome problems associated with the learning and teaching of the subject (Miliszewska & Tan, 2007). These include (Powers *et al.*, 2006): *narrative tools* which support programming to tell a story (e.g. Alice), *visual programming tools* which support the construction of programs through a drag-and-drop interface (e.g. JPie), *flow-model tools* (e.g. VisualLogic) which construct programs through the connection of program elements that represent order of computation, *specialised output realisations* (e.g. Lego Mindstorms) that provide execution feedback in non-textual ways and *tiered language tools* (e.g. ProfessorJ) in which novices can use more sophisticated versions of a language as their expertise develops. These interventions aim to enhance the learning experience and to dispel any negative perceptions of individuals being introduced to programming for the first time. Such tools generally fall into two categories (Kelleher & Pausch, 2005):

1. *Teaching systems* – tools that provide novices with exposure to fundamental aspects of programming. The majority of tools in this category simplify the mechanics of programming by focusing on ways to express (e.g. simplifying typing code), structure (e.g. making new models of programming that are accessible) and understand programs (e.g. making programming concrete). Teaching systems may also attempt to place programming in a context that is accessible, either by providing reasons for programming or by supporting novices working together.
2. *Empowering systems* – systems that aim at teaching programming as an intermediate tool to solve problems. Empowering systems demonstrate different possibilities, and allow problems to be solved, with little regard as to how these skills could be applied in industrial programming tasks. This category includes tools where programming actions are demonstrated in the interface or which aim to improve learners' interaction with the language.

Visual learning, which involves learning based on analogy and abstraction (Miliszewska & Tan, 2007), has successfully supported the teaching of programming (McGrath & Brown, 2005; Pears *et al.*, 2007). Visual learning tools, which often use animation and interactivity to support the learning process, can be found in both of the categories defined by Kelleher and Pausch. Visual learning

engages students more fully in the ideas presented, makes the learning experience stronger and can kick start interest in programming (Lahtinen & Ahoniemi, 2009). Visual learning can also increase the motivational aspects of a programming course while enabling an easier transition to more advanced tasks (Kasurinen *et al.*, 2008; Nevalainen & Sajaniemi, 2006). Programming teaching-systems related to visual learning (or visualisation) tools are typically *microworlds* (which focus on teaching constructs and methods via a high level of abstraction and interactivity) or *direct interaction environments* (which offer support tools such as syntax highlighting) (Henriksen & Kölling, 2004). The work reported in thesis focuses on the use of one visualisation tool, a robot simulator, which shares similarities with these systems. The results of a Systematic Literature Review (SLR) significantly influenced this decision. The SLR, which is presented in Chapter Two along with details of a supplementary literature review, offers a thorough investigation into the use of robots as tools to support the teaching of programming. The SLR also established the need to investigate the use of simulated robots as programming teaching tools.

The use of robots to support the teaching of programming

The use of robots to teach programming principles has long captured the attention of computer science educators (Fagin *et al.*, 2001). This is because robotic agents are exciting to work with (Goldweber *et al.*, 2001; Sklar *et al.*, 2003), appeal to a variety of people of different ages and backgrounds (Hirst *et al.*, 2003; Price *et al.*, 2003) and are capable of evoking complex emotions in humans (Braitenberg, 1984). The use of robot technology can also lead to unintended learning ‘via the back-door’ (Petre *et al.*, 2004). Since the invention of the Logo programming language in 1967 the potential of robot type devices, to support the learning of programming, has been recognised. Logo involved the introduction of, “...the idea of programming through the metaphor of teaching a ‘Turtle’ a new world” (Papert, 1993). In the early-1980s the Karel paradigm, which was influenced by Logo and named after Karel Capek who invented the word “robot” in 1921, was created (Solin, 2013). Developed by Pattis (Pattis, 1981; Pattis *et al.*, 1997), Karel is a teaching tool that models a rudimentary virtual robot, which inhabits a simple grid-based world, and supports limited instructions such as move forward and turn left (Edwards, 2003). Karel continues to influence

approaches to the teaching of programming (Anderson & McLoughlin, 2006) and similar software such as GvR (Yadin, 2011) and RUR-PLE (desJardins *et al.*, 2011). This is despite the limitation that Karel is neither a full-featured, nor sophisticated, machine (Oliveria *et al.*, 2009).

Recent technological advances have allowed for the greater use of physical robots to support the teaching of programming (Soule & Heckendorn, 2011). Most research that has investigated the use of physical robotic tools has focused on Lego Mindstorm robots (Major *et al.*, 2012a). Lego Mindstorms, so-named after Papert's 'Mindstorms: Children, Computers, and Powerful Ideas' book (Papert, 1980), are a mass produced technology based around a programmable brick called the RCX (Fagin *et al.*, 2003b) and later the NXT (Kelly, 2010). Mindstorms support a variety of sensors (such as light, sound and colour) and a range of actuators such as motors. Other physical robots have also been used to support the learning of programming. These include the Scribbler (Cowden *et al.*, 2012), Arduino (Martin & Hughes, 2011) and Koala (Čermák & Kelemen, 2011) robots. Whilst generally considered to be useful in a programming learning context (as is outlined in the following chapter), problems have been associated with the use of physical robots. These include the high financial cost associated with buying and maintaining them, the extensive preparation/set-up time, issues with space and storage, unavailability of robots to learners outside of class, support staff problems and issues with mechanical failure (Goldweber *et al.*, 2001; McWhorter & O'Connor, 2009). Moreover, by using physical robots, students may be inclined to spend their time concentrating on the physical aspect of the technology itself when the link to programming should be being reinforced (Sklar *et al.*, 2006). Partly due to these issues with physical robots the use of simulated robot tools has been investigated (Flot *et al.*, 2012). This is because robot simulators offer an opportunity to overcome the problems with physical robots (Kammer *et al.*, 2011). Indeed, it has been suggested that simulators may replace the need for physical robots in an introductory programming context (Alemany & Cervera, 2012). Preliminary results of on-going research demonstrate the advantages of using simulated robots over physical robots because they allow for faster and more efficient learning (Liu *et al.*, 2013). Substantially less research has, however, considered the use of simulated robots and potential for work in this area

remains (Major *et al.*, 2012a). The aim of this thesis is to address this gap in knowledge by describing an investigation into the use of robots as tools to support the teaching of programming. A review of the literature related to the teaching of programming using robots (both physical and simulated) is provided in Chapter Two. This provides the justification and motivation for the research activities reported. The work also has additional relevance given that the teaching of CS and ICT in UK high schools has been undergoing a period of reform (McAuley, 2012). Traditionally, CS has not been considered as a core curriculum subject in the UK (Drummond, 2009). This has resulted in most high school students not being introduced to programming (Carter, 2006). From 2014, however, computing will be regarded as a science in high schools and teaching approaches will have to be transformed¹. This is despite potential problems that may occur due to such an overhaul (Wells, 2012).

¹ Coughlan, S. (4th February 2013). Computer science part of English Baccalaureate. *BBC News*. <http://www.bbc.co.uk/news/education-21261442> (Accessed 9th October 2013)

1.2 Research Purpose

1.2.1 Aim and Objectives

The overall aim of this thesis is to investigate whether simulated robots are effective tools to support the learning of introductory programming concepts. A robot simulator, and associated 10 hour programming workshop, has been developed and participants (including student programmers and pre- and in-service high school teachers) have taken part in empirical research to achieve this aim. The aim of the thesis incorporates five objectives:

1. Examine the potential and viability of using simulated robots as programming teaching tools
 - a. through a secondary analysis of existing research (through a SLR)
 - b. through the completion of preliminary primary research
2. Further develop an existing robot simulator and associated introductory programming workshop, based on the findings of the secondary analysis, other relevant research and lessons learned when conducting the preliminary research
3. Utilise the robot simulator, and associated teaching material, during a case study involving workshops for novice programmers and trainee and in-service high school teachers
4. Evaluate the robot simulator and workshop approach through an analysis of collected data
5. Produce recommendations, based on the results of the research and lessons learned, to support the future development and use of robot simulators in an introductory programming context

1.2.2 Research Question

The research question that this thesis addresses is:

Is a robot simulator an effective tool for supporting the learning of introductory programming?

The Oxford dictionary² defines effectiveness as:

Effectiveness [noun]

the degree to which something is successful in producing a desired result; success:

e.g. the effectiveness of the treatment

In regards to this thesis, effectiveness has been determined through an analysis of data collected about participants' programming performance and consideration of their opinions, attitudes and motivation while using a robot simulator. In reference to the dictionary definition of effectiveness, the “desired result” for the purpose of this thesis is that a robot simulator supports the learning of introductory programming concepts. For this work a robot simulator will be deemed effective if:

- 1) Participants enjoy learning programming in such a manner,
- 2) Participants believe the approach to be valuable and useful,
- 3) Participants successfully learn introductory programming concepts.

In the context of educating novice computer scientists, previous research has considered the effectiveness of various educational interventions. Moskal (Moskal *et al.*, 2004) partly determined effectiveness by asking, “does exposure to this innovative approach improve student performance?” and, “does exposure to this innovative approach improve students' attitudes and confidence in their ability?”. In other work (Cliburn, 2006b), effectiveness of a game assignment during an introductory programming course was determined by assessing learners' preferences (do they prefer a particular approach to an alternative one), performance (does a particular method result in improved test scores) and motivation (do students demonstrate an appetite to learn further programming skills as a result of the intervention). Finally, a SLR that investigated the

² <http://oxforddictionaries.com> (Accessed 9th October 2013)

effectiveness of pair programming on novice programming courses, identified how researchers determined effectiveness by considering one (or more) of five factors: exam/assignment marks; assignment quality; pass/success rate; retention; confidence, enjoyment and attitude (Brereton *et al.*, 2009).

An underlying assumption of this research is that if a robot simulator can be shown to motivate participants then this would offer some evidence of effectiveness. This is because there is a link between learning and enjoyment (Lumby, 2011) and as it has been shown that increased motivation can lead to improved levels of learner effort, persistence, performance and cognitive processing (Jerez *et al.*, 2012; Ring *et al.*, 2008). Also, work that evaluated physical robots as programming teaching tools suggests that “fun” approaches are likely to be effective (Fagin *et al.*, 2001). As suggested by Marsh (Marsh, 1984), however, effective teaching is a hypothetical construct for which there is no single indicator. When considering the value of an introductory programming teaching tool, therefore, an intervention could also be effective if it excites, provokes interest, dispels stereotypes or misconceptions, promotes a positive early programming experience, helps learners understand concepts or enhances the learning experience. Analysis of the previous work described above demonstrates how effectiveness has broadly been established by:

- assessing performance (e.g. through an analysis of test scores, code quality)
- considering factors related to attitudes and motivation (e.g. enjoyment of an approach, improved confidence, motivation to continue learning, comparison to similar approaches)

For this work, since effectiveness can be determined and established in a number of ways, the case study methodology has been used. This is because case studies allow for an investigation where the boundaries are not clear and they can provide a fuller understanding of how, or why, an intervention works compared to alternative research methods (Pacuilla *et al.*, 2011; Yin, 2009). Further rationale for, and full details of, the case study approach are outlined in Chapter Five.

1.3 Original Contributions

As discussed in Chapter Two, the SLR that was performed in the early stages of the research established the potential to investigate simulated robots as programming teaching tools. This is because only limited high quality research was found to consider such an approach. The SLR, and supplementary literature search that followed, provide the foundation for the activities reported in this thesis. The empirical research undertaken aims to address the gap in knowledge identified by the SLR. The research that has been performed, therefore, is considered necessary and valuable. The SLR itself also contributes to knowledge as it is the first time the methodology has been applied to this research area. The SLR was disseminated in conference and journal papers which have subsequently been cited by several independent studies.

The SLR established the need to explore the teaching of introductory programming using simulated robots and the work that followed investigated the area for the first time in a novel manner. This thesis reports on empirical research that was undertaken using the case study methodology. This is the first time this technique has underpinned work in this research area. The robot simulator developed during the project (named Kobot and discussed further in later chapters) has not been used previously in an education research context. A supporting 10-hour programming workshop and associated data collection instruments were newly created for the purposes of this work. The project itself involved several groups of participants including students (aged 16 to 18 years old) and high school teachers (both pre- and in-service). This combination of participants has enabled an investigation into the effectiveness of a robot simulator in a new way. This has resulted in a substantial amount of information collected for the first time and demonstrates how the work reported is novel and contributes to knowledge. A set of recommendations, based on the results of the research and the lessons learned, have also been produced to support the future development and use of robot simulators in an introductory programming context.

The launch of a new high school curriculum, in 2014, will bring changes to how computing is taught in British high schools (Copping, 2012; Wells, 2012). The fact that recent high school

leavers, in addition to pre- and in-service high school teachers, were involved has relevance in the context of these on-going reforms. This is because a number of issues have been identified that could impact similar approaches which may be used to educate students and high school teachers about programming. The influence and contribution of this work, therefore, extends beyond the original scope of project. However, despite the project being based in the UK, and potentially having implications for the high school computing reforms that are on-going, the contribution of the work is still considered to generalise across international boundaries.

1.4 Thesis Outline

A description of each chapter is now given. See Figure 1-1 for this information in a diagram.

In *Chapter Two* an analysis of existing literature, undertaken to investigate the effectiveness of robots as tools to support the learning of introductory programming, is reported. The Systematic Literature Review (SLR) methodology has been used and a supplementary search of the literature performed. Work reported establishes the potential to investigate robots as programming teaching tools. This is particularly true in regards to simulated robots, as limited high quality research has been found to consider such an approach.

Chapter Three reports on the use of a pre-existing robot simulator, which replicates the movement and behaviour of robots virtually, during two exploratory empirical studies. Details of these activities are outlined and the outcomes of two day-long workshops, involving 23 trainee high school teachers and 10 in-service high school staff, discussed. Lessons learned influenced research that followed.

In *Chapter Four* details of a robot simulator, that has been used to achieve the objectives of this thesis, are presented. Influenced by the findings of the SLR and exploratory studies, the pre-existing robot simulator was enhanced. This modified simulator, named Kobot, allowed for an extensive investigation into the effectiveness of simulated robots as teaching tools. The programming concepts that were taught using Kobot are also provided.

In *Chapter Five* a study design using the case study methodology is presented. The rationale for, and a discussion of the appropriateness of, this method are given. An introduction to the sources of data used and participants involved is presented. This is in addition to details of an associated 10-hour workshop, which has been influenced by education and other research, created to support the teaching of introductory programming concepts. Limitations of the study are also discussed.

In *Chapter Six* details of the execution and results of one component of the case study, Case One – ‘Trainee Teachers’, are presented. Two workshops were completed. In total, 22 trainee Information Communication Technology (ICT) / Computer Science (CS) teachers took part. All had learned programming concepts in some capacity previously. This enabled a comparison of participants’ prior learning experiences with their experience using Kebot. In addition, their views on the effectiveness of the method could be established. Details of other data collection activities involving workshop observations and programming exercises are provided.

In *Chapter Seven* details of the second case study component, Case Two – ‘Students’, are given. Two workshops were again completed. In total, 23 students aged 16 to 18 years old took some part. All were enrolled on a Further Education (FE) course. Data has been collected using questionnaires and programming exercises. Statistical analysis has been undertaken to compare the programming performance of the two cohorts. In-workshop observations were made. An additional data source has been used in the form of semi-structured interviews with three in-service teachers.

In *Chapter Eight* the findings generated from the multiple sources of evidence used during the case study have been brought together. Data have been used to address four propositions and the thesis research question. Recommendations, based on the results and the lessons learned, are presented and suggestions for future research are provided.

In *Chapter Nine* a summary and conclusion is given. The original contributions of this work are again outlined.

Chapter Two

Literature Review

Details the Systematic Literature Review (SLR) and supplementary literature review

Chapter Three

Exploratory Studies

Describes exploratory empirical research involving 23 pre-service high school teachers and 10 in-service high school staff using an existing robot simulator

Chapter Four

Kobot Robot Simulator

Details the main features of a robot simulator that has been developed to achieve the aims of this research. Programming concepts that are taught using the simulator are also outlined

Chapter Five

Case Study Methodology

A case study design is presented. An introduction to the sources of data and participants involved is provided. Details of a 10-hour programming workshop are also given

Chapter Six

Case One: Trainee Teachers

Describes empirical research involving two groups of pre-service teachers (17 in group one, five in group two) during two workshop sessions

Chapter Seven

Case Two: Students

Describes empirical research involving two groups of students aged 16 to 18 (12 in group one, 11 in group two) during two workshop sessions. The results of three in-service teacher interviews are also provided

Chapter Eight

Discussion

Findings generated from multiple sources of evidence have been brought together to address the thesis aims and research question. Recommendations for future work are also outlined

Chapter Nine

Summary and Conclusion

Figure 1-1. Summary of Thesis Content.

Chapter Two

Literature Review

In this chapter an analysis of existing literature, undertaken to investigate the effectiveness of robots as tools to support the learning of introductory programming, is reported. The Systematic Literature Review methodology has been used and nine electronic databases, the proceedings from six conferences and two journals searched for relevant work. After applying exclusion criteria and performing validation exercises, 36 studies were identified. In total, 75% of studies report that robots are an effective teaching tool and can help novices to learn programming. Two and a half years after completion of the SLR, a supplementary search of the literature was performed. This was done to ensure all relevant information, published following the SLR, has been located. Work reported in this chapter establishes the potential to further investigate the use of robots as programming teaching tools. This is particularly true in regards to simulated robots, as limited high quality research has been found to consider such an approach.

2.1 Introduction to the SLR

The Systematic Literature Review (SLR) methodology (Kitchenham & Charters, 2007) is a trustworthy, rigorous and auditable approach that involves the collection of evidence on a topic (Kitchenham, 2004). This evidence is then summarised and gaps in current research may be identified (Kitchenham & Charters, 2007). SLRs can play an important role in supporting education research and can inform practice on the impact of a technology (da Silva *et al.*, 2011). SLRs are referred to as secondary studies while the studies they analyse are examples of primary studies (Kitchenham *et al.*, 2010). The SLR was undertaken as no previous secondary study was found to examine the use of robots as tools to support the learning of programming. This SLR is, therefore, the first of its kind. After initially being disseminated in a conference paper (Major *et al.*, 2011a), the SLR was later extended for publication in a journal (Major *et al.*, 2012a). It should be noted that literature of relevance to other aspects of this research, such as that related to education and research design, is located in alternative areas of this thesis as this chapter considers the use of robots as tools to teach programming only.

2.1.1 Preparation for the SLR

To prepare for the SLR, and to gain an overview of research relating to the teaching of programming, a broad search of existing literature was undertaken. The abstracts of retrieved studies were analysed. Relevant references were stored using the JabRef¹ bibliography manager. In regards to the teaching of programming in general, this preparatory search enabled:

- Relevant journals and conferences to be identified
- An overview of the most investigated topics to be determined
- The scale of the existing body of knowledge to be established
- The identification of relevant academic resources (such as digital libraries)

¹ <http://jabref.sourceforge.net/>

The preparatory search allowed valuable knowledge to be acquired and experience gained influenced the design of the SLR. Whilst such a preparatory search is not a formal part of the SLR process, it allowed an opportunity to become acquainted with literature related to the teaching of programming. Important studies, related to learning programming using robots, were also identified. These would be used to validate the SLR search strategy. Use of different search strings, on a range of electronic resources, influenced the approach that was later adopted. This is discussed further in Section 2.2.2.

2.2 Method

2.2.1 Research Questions

This SLR is based upon the guidelines proposed by Kitchenham and Charters (Kitchenham & Charters, 2007). A protocol was developed as part of the SLR (Major, 2010). This protocol was reviewed by an expert (Barbara Kitchenham of Keele University). The SLR had several objectives:

- Undertake a SLR related to the teaching of introductory programming using robots
- Select a sub-set of studies to review in greater detail
- Collect and analyse the evidence from these studies
- Offer a clear picture of current research in this field
- To provide background so that future research activities can be appropriately positioned

The aim of the SLR was to determine how effective the use of robots has been in the teaching of introductory programming. Six research questions were created to achieve this:

[RQ1] What computer languages are being taught in introductory programming courses that make use of robots as teaching tools?

[RQ2] Are the robots that are being used simulated or physical?

[RQ3] What are the characteristics (i.e. what is the age, level of education etc.) of the novices being taught?

[RQ4] How have researchers investigating the teaching of introductory programming concepts using robots evaluated the approach?

[RQ5] What is the scale (e.g. number of participants) of studies that are being performed?

[RQ6] Do studies suggest that using robots to teach introductory programming is effective?

2.2.2 Search Process

The search process comprised manual and automatic searches of electronic resources. This strategy was deemed suitable after performing trial searches when devising the SLR protocol. Nine electronic databases were searched. Five of these were identified as potentially useful by a previous study (Brereton *et al.*, 2007). The databases searched were:

ACM Digital Library	(http://dl.acm.org)
CiteSeerX	(http://citeseerx.ist.psu.edu)
EBSCOhost	(http://search.ebscohost.com)
ERIC	(http://eric.ed.gov)
IEEEExplore	(http://ieeexplore.ieee.org)
Web of Science	(http://wok.mimas.ac.uk)
Australian Education Index	(http://www.acer.edu.au/aei)
British Education Index	(http://www.bei.ac.uk)
Keele University Library	(http://opac.keele.ac.uk)

Manual searches of conference proceedings took place. Six conferences were identified as relevant after the preparatory literature search (discussed in Section 2.1.1):

ECSS	(European Computer Science Summit)
ESOP	(European Symposium on Programming)
ICSE	(International Conference on Software Engineering)
ITiCSE	(Conference on Innovation and Technology in Computer Science Education)
SIGCSE	(Technical Symposium on Computer Science Education)
SIGITE	(Conference on Information Technology Education)

Two journals were also examined to see if they contained relevant literature:

The Journal of Information Technology Education

Oxford Computer Journal

Search strings were created to retrieve information from the electronic resources outlined:

(robots *OR* robotics) *AND* (“amateur programming” *OR* “amateur programmer”)

(robots *OR* robotics) *AND* (“beginner programming” *OR* “beginner programmer”)

(robots *OR* robotics) *AND* (“first time programming” *OR* “first time programmer”)

(robots *OR* robotics) *AND* (“introductory programming” *OR* “introductory programmer”)

(robots *OR* robotics) *AND* (“novice programming” *OR* “novice programmer”)

(robots *OR* robotics) *AND* “teaching programming”

(robots *OR* robotics) *AND* “learning programming”

The following search string was then used to search on the title and abstracts of papers alone:

(robots *OR* robotics) *AND* programming *AND* (novice *OR* beginner *OR* introductory *OR* teaching *OR* learning *OR* CS1² *OR* “first time”)

These search strings were formed after analysing the keywords of literature that was found during the preparatory search. Use of a two-stage search method was chosen to ensure the collection of all relevant material and to make searches manageable. A trial search was used to validate the search strategy and three studies previously identified as relevant were returned during this (Fagin & Merkle, 2003; Flowers & Gossett, 2002; Petre & Price, 2004).

² CS1 or ‘Computer Science 1’ is a term often used in the USA to describe an introductory computing course.

2.2.3 Inclusion and Exclusion Criteria

Criteria were used to ensure only relevant literature was accepted into the SLR:

Inclusion Criteria

- a. Papers were included that reported the use of robotics in teaching introductory programming to students who were studying a specific Computing or IT-related course.
- b. Papers that report an empirical study, or have a “lessons learned” (experience report) element, were included.
- c. Where several papers reported the same study only the most recent paper was included.
- d. Date of publication did not act as a barrier for inclusion.

Exclusion Criteria

- a. Papers were excluded if their main focus was not on the use of robotics in teaching Computing or IT students introductory programming but on the use of robots in general education courses, as part of a non-IT or Computing related course syllabus or to teach rudimentary programming concepts to very young children.
- b. Papers that just propose an approach or describe the use of robots to teach introductory programming (with no “lessons learnt” component) were excluded.
- c. Papers were excluded if they are not written in English.

The potential impact of some of these criteria is considered shortly.

2.2.4 Quality Assessment

Each study in the final set was assessed for quality. This quality assessment procedure was performed during the data extraction phase and ensured that included studies made a valuable contribution to the SLR. Dybå and Dingsøyr (Dybå & Dingsøyr, 2008) discuss 11 criteria for quality assessment. These were used in an SLR when there were a number of different study types. Use of the same criteria was deemed appropriate during this SLR as it was envisaged that it would also include several types of studies.

The criteria used to assess quality were:

1. Is the paper based on research or is it a “lessons learned” report based on expert opinion?
2. Is there a clear statement of the aims of the research?
3. Is there an adequate description of the context in which the research was carried out?
4. Was the research design appropriate to address the aims of the research?
5. Was the recruitment strategy appropriate to the aims of the research?
6. Was there a control group with which to compare treatments?
7. Was the data collected in a way that addressed the research issue?
8. Was the data analysis sufficiently rigorous?
9. Has the relationship between researcher and participants been considered adequately?
10. Is there a clear statement of findings?
11. Is the study of value for research or practice?

The first two of these criteria were used to exclude non-research papers and those that did not clearly state the aims of the research. This represents the minimum quality threshold that was observed during the SLR. The remaining nine criteria aimed to determine the rigour of the research methods employed and the relevance of each study to the SLR. The answers to each question (in regard to each included study) were tabulated and assigned a value of 1 (‘Yes’) or 0 (‘No’). To assess the validity of the quality assessment procedure one of the PhD supervisors, Theocharis Kyriacou of Keele University (hereafter referred to as TK), assessed a random sample of seven

studies based on the same quality criteria. After a discussion between the author (LM) and TK the quality assessment undertaken was considered valid as the same scores were awarded by both assessors in regards to this random sample of papers.

2.2.5 Data Extraction

To answer the research questions discussed in Section 2.2.1, the following data was extracted from each study included in the SLR:

- Abstract and bibliographic reference
- Publication type (e.g. journal paper, conference paper)
- Study aims and objectives
- Setting
- Information about baseline used (if applicable)
- Number of participants (e.g. number of students in an experiment)
- How authors evaluated their approach
- Characteristics of the novices being taught (e.g. age, level of education)
- Type of computer language being taught using robots (e.g. Java, C++)
- Type of robot used to teach programming (e.g. simulated or physical)
- Findings and conclusions
- Relevance of the study (to the topic under consideration)
- Effectiveness of robots used to teach introductory programming

All data was extracted by a sole reviewer (LM) while the second reviewer (TK) independently extracted information from a random sample of seven publications. These results were then compared and discussed. As no anomalies were evident the data extraction strategy was considered appropriate. Extracted data was stored in a spread sheet.

2.3 Results

2.3.1 Search Results

After reading the full text of 60 studies that were selected using the search process 34 relevant studies were initially identified. The remaining 26 were judged to be either irrelevant or incompatible with the inclusion criteria. Following this the ‘snowball’ technique was used. This involved reviewing the ‘Background’ or ‘Introduction’ sections of the 34 identified papers for other potentially relevant information. The references section of each study was also analysed. After consulting the second PhD supervisor, Pearl Brereton of Keele University (referred to as PB), it was decided that if four or more additional studies were found that met the inclusion criteria then this would indicate that the SLR strategy was flawed. Two additional studies were identified (Wong, 2001; Imberman & Klibaner, 2005) and this brought the number of total studies considered to 36. Reasons why these studies were not found originally are discussed later in this chapter. Appendix A1 lists the studies included in the review while Appendix A2 displays the results of the automatic search process. Figure 2-1 shows the year of publication for included studies. Only one study was published before 2000. The SLR identified studies published up to (and including) 2010.

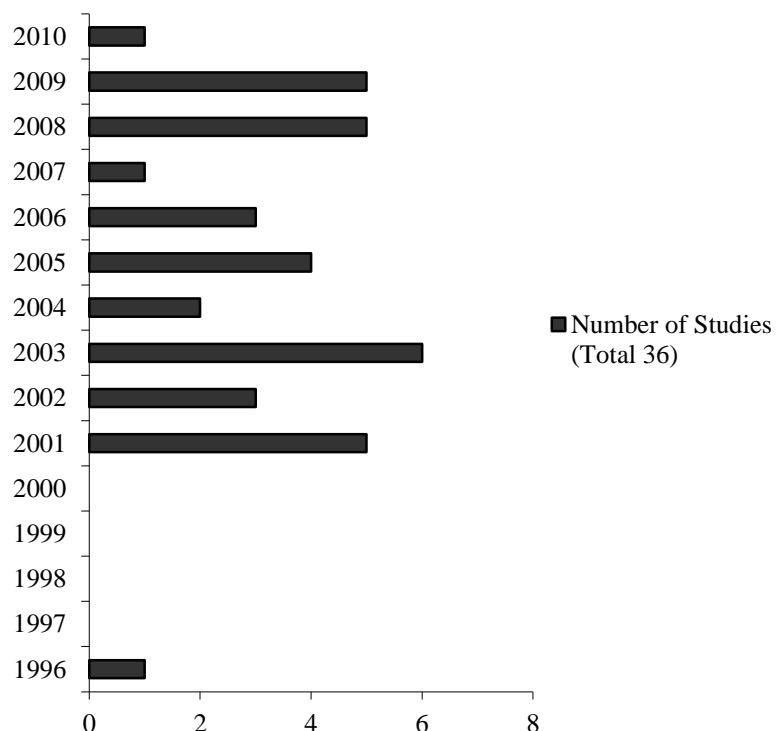


Figure 2-1. Publication year of studies included in the SLR.

2.3.2 Quality Assessment

In Section 2.2.4 the quality assessment strategy used during the SLR is discussed. The results of this evaluation are presented in Table 2-1. Each study has been assigned a quality score out of 11.

<i>Study</i>	<i>Research</i>	<i>Aim</i>	<i>Context</i>	<i>Research Design</i>	<i>Recruitment Strategy</i>	<i>Control Group</i>	<i>Data Collection</i>	<i>Data Analysis</i>	<i>Relationship</i>	<i>Findings</i>	<i>Value</i>	<i>Quality Score</i>
[Barnes02]	1	1	0	0	0	0	0	0	0	0	1	3
[Becker01]	1	1	1	1	0	0	0	0	0	1	1	6
[Bell08]	1	1	1	1	0	0	0	0	0	1	1	6
[Borge04]	1	1	1	1	1	0	1	1	0	1	1	9
[Brauner10]	1	1	1	1	1	0	1	1	1	1	1	10
[Buck01]	1	1	0	0	0	0	0	0	0	1	1	4
[Cliburn06]	1	1	1	1	0	0	0	0	0	1	1	6
[Czejdo09]	1	1	0	0	0	0	0	0	0	0	1	3
[Enderle08]	1	1	1	1	0	0	0	0	0	1	1	6
[Fagin01]	1	1	0	0	0	0	0	0	0	1	1	4
[Fagin03a]	1	1	1	1	1	1	1	1	1	1	1	11
[Fagin03b]	1	1	1	0	0	0	0	0	0	1	1	5
[Flowers02]	1	1	1	1	0	0	0	0	0	1	1	6
[Garrett05]	1	1	1	1	0	0	0	0	0	1	1	6
[Goldweber01]	1	1	0	0	0	0	0	0	0	0	1	3
[Hirst03]	1	1	1	1	0	0	0	0	0	1	1	6
[Imberman05]	1	1	1	1	0	0	0	0	0	1	1	6
[Imberman07]	1	1	1	1	1	0	1	1	0	1	1	9
[Jadud03]	1	1	1	0	0	0	0	0	0	0	1	4
[Kurebay06]	1	1	1	1	1	0	1	1	0	1	1	9
[Ladd05]	1	1	1	1	0	0	0	0	0	1	1	6
[Lauwers09]	1	1	1	1	1	1	1	1	0	1	1	10
[Lawhead02]	1	1	1	0	0	0	0	0	0	0	1	4
[Lemone96]	1	1	1	0	0	0	1	0	0	1	1	6
[Mcwhorter09]	1	1	1	1	1	1	1	1	1	1	1	11
[Petre04]	1	1	1	1	1	0	1	1	0	1	1	9
[Sartatzemi03]	1	1	0	1	1	0	1	0	0	1	1	7
[Sartatzemi05]	1	1	1	1	0	0	0	0	0	1	1	6
[Sklar06]	1	1	1	1	1	0	1	1	0	1	1	9
[Summet09]	1	1	1	1	1	1	1	1	1	1	1	11
[Vandelden08]	1	1	1	1	0	0	0	0	0	1	1	6
[Weiss08]	1	1	0	1	0	0	0	0	0	1	1	5
[Wellman09]	1	1	1	1	1	1	1	1	0	1	1	10
[Williams03]	1	1	1	1	1	0	1	0	1	1	1	9
[Wong01]	1	1	1	1	1	0	1	1	0	1	1	9
[Wu08]	1	1	1	1	1	1	1	1	0	1	1	10
Totals	36	36	29	27	15	6	16	13	5	31	36	/

Table 2-1. Results of the SLR Quality Assessment.

Table 2-1 displays how all included studies are based on research (or presented a lessons learned account) and clearly stated their aims. Of the 36 included studies, 29 offered a description of the context in which the research was carried out while 27 had an appropriate research design. A large proportion of included studies were found to have an inadequate recruitment strategy, failed to use a control group, did not collect (or sufficiently analyse) data in a way that addressed the research issue and did not consider the relationship between participants and researcher(s). The majority of studies that scored 0 in respect to these criteria offered a “lessons learned” account and did not report any empirical data. Three of the studies included in the review scored 11. The lowest score was three. The average quality score of studies was 6.9. The mode score of studies was six (with 12 studies achieving this score).

Feedback received from a reviewer after submitting the SLR for presentation at the EASE 2011 conference (the final article is available in Major *et al.*, 2011a) suggested removing low scoring studies from the aggregation (i.e. those with a score of five or below) and to analyse the effect of such a change. It was believed this might be useful given the average quality score of included literature varied widely and as low scoring studies often report larger treatment effects. Nine studies were awarded a quality score of five or less during the first round of quality assessment. These were removed from the aggregation during the second round of quality checking. When the nine low scoring studies are omitted the average quality score of included studies rises from 6.9 to 8 (out of 11). The implications of removing low scoring studies, in regards to the effectiveness of using robots to teach introductory programming, is discussed further in Section 2.3.3 [RQ6].

2.3.3 Research Questions

A summary of the information extracted from all included studies is shown in Table 2-2.

<i>Study</i>	<i>Language Taught</i>	<i>Robot Type (Phy or Sim)³</i>	<i>Type of Novices</i>	<i>Evaluation Method</i>	<i>Number of participants</i>	<i>Are Robots Effective?</i>
[Barnes02]	Java	Phy - LegoMind	University	-	-	<i>Unclass.</i>
[Becker01]	Java	Sim	University	-	-	Yes
[Bell08]	Robolab	Phy – LegoMind	High School	-	-	Yes
[Borge04]	Java	Sim	University	Interviews and Discussions	<i>Unknown</i>	Yes
[Brauner10]	Scratch	Phy and Sim	High School	Questionnaire	31	Yes
[Buck01]	Java	Sim	University	-	-	<i>Unclass.</i>
[Cliburn06]	<i>Various</i>	Phy – LegoMind	University	-	-	Mixed
[Czejdo09]	Python	Phy – Scribbler	University	-	-	Yes
[Enderle08]	C++	Sim	Various	-	<i>Unknown</i>	Yes
[Fagin01]	Ada	Phy – LegoMind	University	-	-	Yes
[Fagin03a]	Ada	Phy – LegoMind	University	Comparative – Learning with and without robots	938	No
[Fagin03b]	Ada	Phy and Sim	University	-	-	Mixed
[Flowers02]	Java	Phy – LegoMind	University	-	-	Yes
[Garrett05]	NQC	Phy and Sim	Various	-	-	Yes
[Goldweber01]	<i>Various</i>	Phy – Several	University	-	-	Mixed
[Hirst03]	<i>Various</i>	Phy – LegoMind	Various	-	-	Yes
[Imberman05]	C	Phy – Handyboard	University	-	<i>Unknown</i>	Yes
[Imberman07]	C++	Phy – LegoMind	University	Questionnaire	121	Yes
[Jadud03]	Scheme	Phy – LegoMind	High School	-	-	Yes
[Kurebay06]	Dolittle	Phy – Custom	High School	Questionnaire	40	Yes
[Ladd05]	Java	Sim	University	-	-	Yes
[Lauwers09]	Java	Phy – iCreate	University	Questionnaire + Analysis of Grades + Impact upon retention	72	Yes
[Lawhead02]	Java	Phy – LegoMind	University	-	-	Yes
[Lemone96]	C++	Sim	High School	<i>Unknown</i>	15	Yes
[Mcwhorter09]	C++	Phy – LegoMind	University	Questionnaire	78	Mixed
[Petre04]	<i>Various</i>	Phy – Several	Various	Observations & Interviews	<i>Unknown</i>	Yes
[Sartatzemi03]	<i>Various</i>	Sim	University	Pilot Lessons	20	Yes
[Sartatzemi05]	Robolab	Phy – LegoMind	High School	Pilot Lessons	<i>Unknown</i>	Mixed
[Sklar06]	Various	Phy – Several	University	Questionnaire	<i>Unknown</i>	Yes
[Summet09]	Python	Phy – Scribbler	University	Analysis of Grades	259	Yes
[Vandelden08]	Java	Phy – LegoMind	University	Questionnaire	<i>Unknown</i>	Yes
[Weiss08]	<i>Various</i>	Phy – Several	University	-	-	<i>Unclass.</i>
[Wellman09]	Custom	Phy – iCreate	University	Questionnaire	40	Yes
[Williams03]	C	Phy – LegoMind	University	Questionnaire	45	Yes
[Wong01]	<i>Various</i>	Phy – LegoMind	University	Questionnaire + Comparative – Learning with against learning without robots	<i>Unknown</i>	Yes
[Wu08]	Java	Phy and Sim	High School	Comparative – Physical against Simulated Robots	151	Yes

Table 2-2. Data extracted from studies included in the SLR.

³ Phy – Physical Robot(s) / Sim – Simulated Robot(s)

[RQ1] What computer languages are being taught in introductory programming courses that make use of robots as teaching tools?

When analysing studies included in the SLR, 10 categories were established in regard to the programming languages used. Java was the largest contributor having been used in 10 studies. This was followed by eight studies that reported the use of a combination of programming languages. C++ was implemented during four studies while the use of C was reported in two. Not Quite C (NQC) was used during one study. Ada (three studies), Python (two studies) and Scheme (one study) have also been used. Attempts have been made to use specially designed programming languages in conjunction with robots. This includes use of the Robolab software (two studies), the Scratch (one study) and Dolittle (one study) educational languages. A customised language that integrated use of the Alice animation software was also reported by one study. Figure 2-2 presents this information.

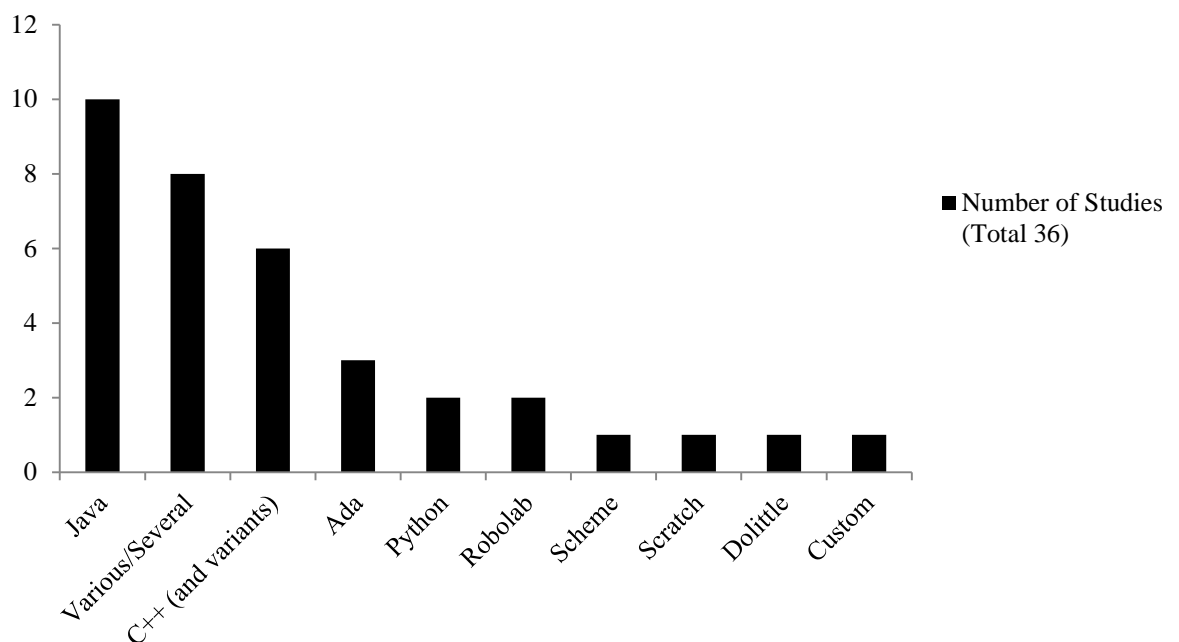


Figure 2-2. Programming languages used during studies included in the SLR.

[RQ2] Are the robots that are being used simulated or physical?

Of the studies included in the SLR, 25 report the use of physical robots, seven studies discuss the use of some sort of simulated robot technology while four report the use of physical and simulated

robotic tools. The 25 studies that report on physical robots can be divided to reflect the type used. It was found that 15 studies describe the implementation of Lego Mindstorms while the iCreate and Scribbler robots were each discussed in two studies. One study details the use of a custom robot while four made use of several types of physical robot. One study reports use of the HandyBoard robotics controller.

[RQ3] What are the characteristics of the novices being taught?

The context of each study was scrutinised to determine the characteristics of the novices involved. Three groupings were established: university, high school or ‘various’. For this SLR University students were considered to be learners enrolled in Higher Education (HE - aged 18 or older) while high school students identified learners still in compulsory education (aged around 11 years or older). Of the 36 studies 25 reported on the use of robots in a university setting, seven were based in a high school and four discussed the use of robots in different environments.

[RQ4] How have researchers investigating the teaching of introductory programming concepts using robots evaluated the approach?

Use of questionnaires (or course critique surveys) was the most common method by which included studies evaluated their findings and proposals (10 studies). Interviews and focus groups have also been described (two studies) while robotic interventions have been implemented in pilot lessons (two studies). In addition, comparative experiments have taken place. These involved contrasting the effect of learning with robots to learning without (two studies) as well comparing physical and simulated technologies (one study). Analysis of student grades has also been reported in two studies while one study examined the impact that robots had upon retention rates. The remaining 16 studies offered a “lessons learned” report or did not explicitly state the results of any empirical research.

[RQ5] What is the scale of studies that are being performed by researchers?

16 included studies are examples of “lessons learned” or experience style reports while 20 offer evidence that empirical research took place. The scale of studies included in the SLR varied widely and ranged from studies involving 15, 20 and 31 participants through to larger studies with 121 and 151 people involved. One study compares the test results of hundreds of participants who took part in robotics and non-robotics based classes (Fagin & Merkle, 2003) and the potential implications of including such a large study in the review are discussed in Section 2.5.1. 12 studies were found to report the exact number of students that took part while eight discuss conducting empirical research, or collecting information from participants, but do not state the number involved.

[RQ6] Do studies suggest that using robots to teach introductory programming is effective?

Of the 36 studies identified, 27 report that the use of robots is effective when teaching introductory programming concepts, five offer mixed results while one study reports robots to be ineffective. Three studies were unclassifiable as they do not provide a measure of effectiveness. Figure 2-3 displays this information in graphical form.

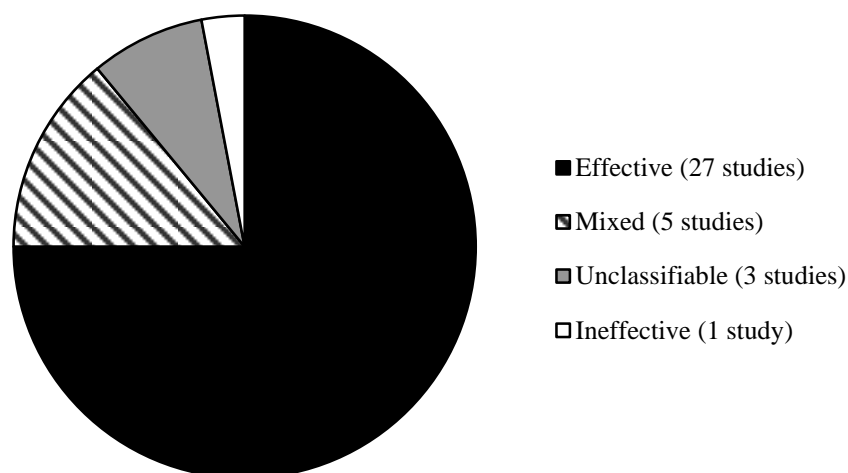


Figure 2-3. Effectiveness of robots as tools to teach programming.

In total, 36 studies were included. Figure 2-4 shows this information.

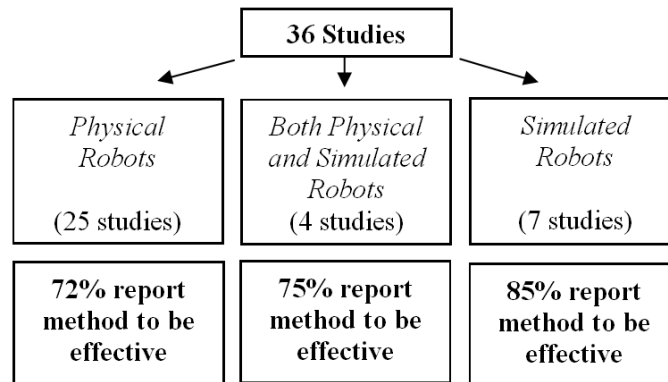


Figure 2-4. Breakdown of effectiveness of robots as tools to teach programming (by nature of robot reported on).

Comments received after the peer-review of the SLR (after submission to the EASE 2011 conference) suggested it would be interesting to examine the effect of removing the nine studies with a quality score of five or less. This is because past research in medicine has found that low quality studies reported significantly larger effects (i.e. treatment impact) than higher quality studies (Moher *et al.*, 1998) and that this can alter the interpretation of benefit in regards to an intervention. Other work similarly found how low-quality studies show more beneficial treatment effects (Shang *et al.*, 2005). Nine studies were excluded that had been awarded a quality score of five or less. This left a subset of 27. The effectiveness of robots as tools to teach programming, arranged by robot type and with low scoring studies excluded, is displayed in Figure 2-5. After removing low scoring studies all that discuss simulated robots report this method to be effective.

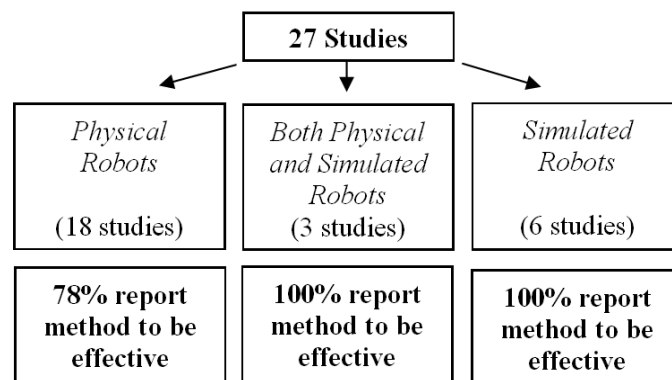


Figure 2-5. The effectiveness of robots as tools to teach programming after excluding studies awarded a quality score of five or less.

2.3.4 Studies Related to Simulated Robots Identified During the SLR

When considering all 36 included studies, five of the seven related to simulated robots discuss the use of Karel the Robot (Becker, 2001; Buck & Stucki, 2001; Borge *et al.*, 2004; Lemone & Ching, 1996; Sartatzemi, 2005). The two other studies included in the SLR consider the use of simulated tools. Enderle (Enderle, 2008) discusses a tool for the teaching of object-oriented (OO) concepts. Influenced by a flow chart approach, the Graphical Robot Programming for Beginners software (GRAPE) allows students to produce code from user input. This permits mapping between flow charts and programming syntax. As such GRAPE does not replicate the movement of a robot, but instead provides a simulated means for the production of code. Ladd and Harcourt (Ladd & Harcourt, 2005) discuss a more conventional robot simulator environment. In this study students engaged in competitions using a custom built simulator that replicates several winter sporting events. The approach is reported to be effective. The four papers that consider the use of physical and simulated technology together have also been examined to determine the nature of the simulated tools used. One study (Wu *et al.*, 2008) describes a simulator environment that is used in conjunction with Lego Mindstorms robots. Fagin and Merkle (Fagin & Merkle, 2003) describe the use of an Ada/Mindstorms programming tool that allows the execution of code to be traced although actual robot movement is not replicated. Garrett and Thornton (Garrett & Thornton, 2005) discuss how a web-based interface was used to control a robot. Brauner (Brauner *et al.*, 2010) details a rudimentary robot simulator using the Scratch programming environment.

After analysis of the seven studies included in the SLR that consider simulated robots alone, the quality and rigor of these was judged to be poor as: four offer a ‘lessons learned’ account, or description of an approach, but provide no empirical data (Becker, 2001; Buck & Stucki, 2001; Enderle, 2008; Ladd & Harcourt, 2005); one describes the results derived from interviews/discussions as being non-generalisable as only four novice programmers were involved (Borge *et al.*, 2004); one specifies the involvement of 15 participants, and the use of a questionnaire, but presents no quantitative data (Lemone & Ching, 1996); one describes the implementation of pilot lessons, involving 20 students, but does not undertake any detailed analysis

(Sartatzemi, 2005). Moreover, none of the four studies which considered the use of physical and simulated technology together (Brauner *et al.*, 2010; Garrett & Thornton, 2005; Fagin & Merkle, 2003; Wu *et al.*, 2008) present a rigorous evaluation of the effectiveness of a robot simulator. This is because these four studies focused on a comparison of physical and simulated robot technologies, rather than considering the effectiveness of just simulated robots.

2.3.5 Limitations of the SLR

The main threats to the validity of the SLR relate to bias when selecting publications and inaccurate data extracted. Search strings were developed after conducting preliminary searches, consulting experts and using a thesaurus. A range of resources, relating to education and computing research, were searched to ensure thoroughness. These were selected after referring to previous work and after undertaking an initial analysis of the literature. Despite this it is not possible to guarantee that all relevant studies were returned and there is a risk that some may have been omitted. Publication bias (the phenomena where ‘negative’ results are less likely to be published) may also have had some impact on the findings of the SLR, although it is difficult to ascertain whether this was the case. The data extraction process may also have been biased. This is because the data extraction procedure was performed by one reviewer. The development of a SLR protocol and use of validation strategies (i.e. data being extracted from random a sample of papers by a PhD supervisor) helps to reduce this bias. Finally, it is possible that the inclusion criteria may have resulted in the inadvertent exclusion of some relevant papers. This is because the criteria excluded studies that contained no “lessons learned” component or were related to the teaching of very young children. It is recommended that when researchers perform future SLRs, similar in scope to the one presented, that they consider the potential effects of adopting similar criteria as these could have a detrimental impact upon a SLRs findings.

2.4 Supplementary Thesis Literature Update

The SLR includes studies published up to, and including, 2010. To ensure all relevant work (published between January 2011 and June 2013) has been considered in this thesis, an additional search of the literature has taken place. Whilst the search strategy outlined in Section 2.2 was not re-implemented, previous experience gained when conducting the SLR influenced this supplementary search. This is because the same search terms and electronic resources were used as during the SLR to locate relevant papers. A manual search of the Journal of Computing in Small Colleges, and the SIGCSE conference proceedings, also took place. This is in addition to the ‘snowball’ technique being adopted when analysing retrieved literature.

As was the case during the SLR, most studies identified during this supplementary search discussed the use of physical robots. Lego Mindstorms were, again, the most commonly used physical robotic tool (Butterworth, 2012; Gavan & Anderson, 2013; Howard *et al.*, 2012; Szweda *et al.*, 2012a; Yu, 2012). One of these papers considered the use of Lego Mindstorms during a three-day workshop to introduce high school teachers to programming concepts (Kay & Moss, 2012). Similar teacher education workshops have also been reported (Saad *et al.*, 2012). Use of other physical robots was noted. This includes the Scribbler (Cowden *et al.*, 2012; Salcedo & Idrobo, 2011), Arduino (Martin & Hughes, 2011) and Koala (Čermák & Kelemen, 2011) robots. Other studies were found to consider issues associated with the use of physical robots, in particular their high cost, and Soule and Heckendorn (Soule & Heckendorn, 2011) discuss how they believe it is possible to put computationally powerful robots in Computer Science (CS) classrooms. The Finch robot, a product of Carnegie Mellon’s CREATE lab, is one example and costs less than \$100 (Lauwers, 2013).

During the supplementary search, a limited amount of research was found to consider the use of simulated robots. One study reports how it is not always feasible to distribute physical robot kits to students and this can impact independent learning outside of lessons (Kammer *et al.*, 2011). The authors of this study discuss the development of a simulated environment that can be used with LEGO Mindstorm robots. Such an approach was found to improve the outreach of CS courses.

Other work discusses a rudimentary robot simulator to facilitate the teaching of Java (Szweda *et al.*, 2012b). This study reports how simulated tools can have a positive impact but that there is a need for an evaluation of a more sophisticated simulator (e.g. one in which virtual objects can be introduced). This is because the tool implemented did not replicate the movement of a robot on-screen. The RoboBuilder project is described as a blocks-based programming game that draws on constructionist design principles to create a computational learning environment (Weintrop & Wilensky, 2013). As the RoboBuilder project is a low-threshold programming environment (due to 'blocks' being arranged on screen rather than code typed) the software is only considered to be a simulator for the teaching of core computational thinking skills (such as algorithmic thinking and debugging). Alemany and Cervera (Alemany & Cervera, 2012) present an experience report that summarises their teaching experiences using robots. It is discussed how realistic robot simulators may replace the need for physical robots, and that they may be implemented on distance learning courses. Alemany and Cervera also establish a connection between the appeal of simulated robots and videogames. An experience report describing use of the Robomind software was also identified during the supplementary search (CAS, 2011). Robomind is a learning support program based on the Logo programming language (Kochakornjarupong, 2010) and similar software has been used to support widening participation activities that aim to inspire young people to think imaginatively about computing (Posey, 2012). Finally, one other relevant study was found to have been published since completion of the SLR. This involved an investigation into the effect of robots upon student motivation and found how robots had a positive influence on participants' attitudes towards programming (McGill, 2012).

None of the additional studies found during the supplementary search of the literature reported negative findings in regards to the use of robots as programming teaching tools. All studies reported robots as being effective or offered no specific verdict. The number of studies identified during the supplementary search suggests that research in this area is still on-going. Literature retrieved during the supplementary offered similar conclusions to that collected during the SLR. This is because:

1. The use of physical robots remains more greatly reported than the use of simulated robots
2. Lego Mindstorm robots were, again, the most commonly reported physical robot used
3. There continues to be a need to investigate the use of simulated robots, as tools to teach programming, due to a lack of high quality research considering such an approach

As a final point it was noted that the SLR papers (Major *et al.*, 2011a; Major *et al.*, 2012a) have been cited in at least five studies (Butterworth, 2012; Kurkovsky, 2013; López *et al.*, 2013; Prayaga *et al.*, 2012, Suliman & Nazeri, 2012). This is in addition to the SLR being cited in a magazine article reporting research taking place at Carnegie Mellon University (CMU) and involving the University of Pittsburgh (PITT) (Flot *et al.*, 2012). The Robot Virtual Worlds (RVW) project involves the development of software that enables programmers to test their code in a simulated environment before it is transferred to a physical robot. During future studies researchers at these two institutions aim to determine whether students learn programming better using simulated, rather than physical, robots. It is believed, once the three year RVW project is completed, that the findings may complement the research presented in this thesis. Early results released by the RVW team indicate how both physical and simulated robots aid students' learning of programming concepts (Liu *et al.*, 2013). In addition, it is reported how simulated robots allowed students to learn faster and more efficiently.

2.5 Discussion

The results of the SLR provide the platform for the research in this thesis. In this section the implications of the SLRs findings are discussed.

2.5.1 The Findings of the SLR

Various observations can be made as a result of the SLR. In regards to the mean quality score of included studies it is believed that this figure is low. As Table 2-1 shows, a high proportion of studies in the set of 36 lacked vital experimental features whilst the analysis of data was often considered to be poor. This is due to 16 included studies being experience reports. Such studies do not score well against the quality assessment criteria used. To address the shortcomings identified it is recommended that future work investigating the teaching of programming using robots should:

- Ensure that an adequate and justified strategy is used to recruit participants
- Collect and analyse data in a way that directly addresses the research issue
- Consider the relationship between participants and the researcher(s)
- Report empirical data collected as opposed to offering a narrative summary, brief overview or “lessons learned” account

Removing studies that scored poorly in the quality assessment did not significantly change the results of the SLR. When low scoring studies are removed from the aggregation the percentage of studies reporting robots to be effective, however, does increase. This was the case for the use of physical robots alone, the use of simulated robots alone and the use of physical and simulated technology together. It should be noted that the results of the SLR do not offer conclusive evidence that one type of robot is more effective. This is because of the relatively small sample sizes involved when low scoring literature is removed. As a result, and due to included studies using a wide range of methods to collect data, statistical analysis has not been undertaken.

One large experiment included in the SLR (Fagin & Merkle, 2003) reports a year-long study that compared the performance of over 800 students, on identical tests, during robotics and non-robotics based laboratory sessions. This study was awarded the maximum score of 11 for the quality assessment. The study conducted by Fagin and Merkle (Fagin & Merkle, 2003) was the only one included in the SLR that reported negative results in regards to the effectiveness of using robots as programming teaching tools. The authors identify a range of reasons why this may have been the case and state how the use of a robot simulator may have helped to overcome the issues students encountered. These results may not be more valuable than those studies that involved fewer participants. This is because the programming language implemented during the Fagin and Merkle study was based on a scaled-down version of Ada. The more elaborate features of Ada cannot be executed on the RCX robot hardware used (Fagin, 2000). Due to a reduced version of Ada unlikely being as conceptually difficult for novices to learn as others programming languages this study is not considered, therefore, to offer definitive evidence. The SLR has helped to establish the following:

- The Java programming language is the one that has been most frequently adopted
- The use of physical robots is more commonly reported than the use of simulated robots
- Where a primary study takes place, questionnaires are the most commonly reported method used to evaluate the effectiveness of robotic interventions
- The number of participants who have taken part in research to evaluate the use of robots to teach introductory programming varies greatly from study to study

Whilst caution must be used due to the questionable quality of some included studies, the SLR indicates that the use of robots can be an effective learning tool when used in an introductory programming course. This is because three quarters of studies state that robots are effective. The most interesting finding to arise from the SLR, however, is that the use of simulated robots may potentially be more effective than physical ones. As no studies report simulated robots to be ineffective, the use of simulated robots may be just as, if not more, effective than physical robots. It should be noted, however, that the use of physical robots has been much more commonly reported.

2.5.2 Rigour of the SLR and Identification of Prominent Sources

As only two additional studies were found during the ‘Snowballing’ validation process it is considered that the search strategy adopted was effective and rigorous. It is thought that one of the additional studies (Wong, 2001) was not found initially as it is not indexed by the electronic resources, or covered by the manual search, used during the SLR. The second additional study (Imberman & Klibaner, 2005), however, could have been identified as it is indexed in the ACM Digital Library. After analysis of the title, abstract and keywords of this article it was noted how the term ‘programming’ does not appear. Programming terms such as, “repetition, selection and the use of basic functions” are instead referred to. As ‘programming’ was used in every search term it is probable that this study was overlooked as it was not returned during the automatic searches. It is considered that this is not a weakness of the search terms themselves. This is because it is not unreasonable to assume that if an article discusses the teaching of programming then the term ‘programming’ would appear in either the title, or abstract, of the paper (as is the case with the other 35 studies included). This highlights how some relevant literature can be missed and demonstrates how appropriate terms must be used by authors in titles, abstracts and keywords if secondary reviewers are to successfully locate their work. It was also noted when analysing the background and references sections of included studies that a large number of citations were repeated in different articles. This increases confidence in the results of the SLR, indicates that all important work has been identified and validates the search terms used.

Finally, the sources of studies included in the SLR have been examined to identify prominent journals or conferences consistently publishing work related to the teaching of introductory programming using robots. Of the 36 included studies 17 were published in conference proceedings, 15 in journals and four elsewhere (for example as technical reports). The references of included studies have been examined to determine if any sources have published a substantial number of studies included in the review. Two such sources have been identified: The Journal of Computing in Small Colleges (which published eight included studies) and the SIGCSE Technical

Symposium on Computer Science Education (an annual conference and also found to have published eight studies). These two sources contributed 16 of the 36 included studies. It is recommended that these sources are candidates for a manual search if future work is to build upon this SLR. However, at this point, it has been decided that this is not necessary as the post-SLR validation exercise would have likely uncovered relevant references not already found. The sources of the remaining 20 studies were varied and no other conference or journal publication was identified as contributing a substantial number of articles.

2.5.3 The Potential to Investigate the Effectiveness of Simulated Robots

A large proportion of included studies related to simulated robots discussed Karel the Robot. The Karel paradigm was created by Pattis (Pattis, 1981; Pattis *et al.*, 1995) and became a widely known CS teaching tool (Anderson & McLoughlin, 2006). Karel shares similarities with the Logo Turtle (Papert, 1980) although Logo is unable to interact with its environment (Brusilovsky *et al.*, 1998). The constructionism learning theory Papert advocated as a result of work related to Logo established the principal of “learning by making”. Constructionism has, and continues, to influence the use of robots in an educational context (Ackermann, 2001). Constructionism builds on the constructivist approach outlined by Piaget (Piaget & Cook, 1952).

The popularity of Karel has resulted in descendants in various programming languages (Watson & Harrison, 2012). Karel is a basic virtual robot that navigates in a simple two-dimensional, grid-based, world and supports a limited set of instructions such as move forward and turn left (Edwards, 2003). Whilst the Karel approach has been reported as successful (Becker, 2001; Borge *et al.*, 2004; Lemone & Ching, 1996), concerns have been raised. Karel has the academic function of teaching the basics of imperative programming and is neither a full-featured, nor sophisticated, machine (Oliveria *et al.*, 2009; Kelleher & Pausch, 2005). The simplistic nature of Karel, in particular that the grid-based environment can be restrictive, has come under criticism. Karel is limited to moving one grid (or ‘block’) at a time and can only ever face North, South, East or West (Untch, 1990). As a result, programs that learners create are often constrained due to the limited

commands that can be issued or because of Karel's environment. Due to the simplistic nature of the approach it is possible for students to completely master Karel in just a few hours (Szweda *et al.*, 2012b). The relevance of Karel to today's students has also been questioned. Previous work has likened Karel to a programmable cursor, and suggests that it is difficult to maintain the interest of today's "Plug & Play" generation using a text-based graphical representation, as has been the case with some implementations (Anderson & McLoughlin, 2006).

The SLR has revealed a gap, in existing knowledge, related to the teaching of programming using simulated robots. As discussed in the chapters that follow, this thesis addresses this gap by reporting an empirical investigation that considers the use of a robot simulator as a tool to support the learning of programming. As the SLR has identified deficiencies in existing work the research that has been undertaken is considered necessary, valuable and as contributing to knowledge. The shortcomings of existing work, identified after the analysis of the literature, are discussed below. How these issues will be addressed by the research project is also presented.

- The results of the SLR demonstrate how a limited number of studies (in total seven) have investigated the teaching of programming using simulated robots. Whilst in some circumstances such a number could be indicative of a complete body of research, this is not the case because the quality of included studies has been judged as poor. At the time of the SLR there was not a substantial quantity of high quality research that has investigated simulated robots as tools to teach introductory programming. The supplementary search of the literature, completed two and a half years after the SLR, also demonstrates how there remains a need to investigate the use of simulated robots as programming teaching tools. The research presented seeks to address this issue by performing a rigorous, thorough and transparent empirical investigation into the use of a robot simulator in such a manner.
- A large percentage of identified studies (related to both physical and simulated robots) were found to lack vital experimental features or inadequately reported results. Due to this, it is not possible to draw strong conclusions. This research project will address these shortcomings by following the recommendations outlined in Section 2.5.1, specifically by: ensuring the use of

an adequate and justified recruitment strategy; collecting and analysing data in a way that appropriately addresses the research issue; considering the relationship between participants and the researcher(s); presenting and discussing empirical data collected as opposed to offering a narrative summary, brief overview or “lessons learned” account.

- A limited number of included studies have considered the use of simulated robots and there is significant potential for further research. Indeed, only two studies included in the SLR (Ladd & Harcourt, 2005; Wu *et al.*, 2008) discuss the use of a simulator that attempts to replicate the behaviour and movement of a physical robot in the real-world (i.e. one that is capable of turning at a variety of angles, can move in different directions at a range of speeds and has various sensors). Instead, most studies focus on the use of Karel (*n.* 5). It is debatable whether Karel can be considered as a complete robot simulator or whether the approach only adopts robots as a metaphor. The research project presented enhances knowledge in this area through the implementation of a robot simulator, modelled on a real-world robot, in a number of introductory programming workshops. The simulator developed during the project, called Kebot, has not been used previously in an education research context while the supporting 10-hour programming workshop was newly created for the purposes of this work. Kebot allows simulated agents forwards and backwards movement through 360 degrees, visual arena backgrounds to be loaded, and the introduction of ‘2D’ and ‘3D’ objects that allows for interaction through various sensors. Due to these factors, Kebot is considered to offer a substantially different approach than other simulators that have been used to support the teaching of introductory programming previously.

2.6 Summary

This chapter has reported on an analysis of the literature that was undertaken to investigate the effectiveness of robots as tools to support the learning of introductory programming. Use of the Systematic Literature Review (SLR) methodology has been described. SLRs offer a trustworthy, rigorous and auditable approach and can enable gaps in existing research to be identified. The SLR reported in this chapter is the first work of its kind. The results of the SLR provide the justification for the research activities reported in the remainder of this thesis. As the SLR was completed in early-2011, a supplementary search of the literature has also taken place. This ensures that the findings of all recent relevant studies have been considered.

During the SLR, nine electronic databases, the proceedings from six conferences and two journals were searched for research relevant to the teaching of programming using robots. After applying exclusion criteria, and after performing validation exercises, 36 studies were identified and accepted into the final set of included literature. In total, 75% of studies report that robots are an effective teaching tool and can help novices to learn programming. However, the potential to further investigate methods for the implementation of robots was considered to remain. This is particularly true in regards to the use of simulated robots, as limited high quality research has been found to consider such an approach.

This chapter has identified how research into the use of simulated robots, as tools to support the learning of programming, is required. The results of the SLR and supplementary literature search provide the foundation upon which future work has been built. In the remainder of this thesis research will be presented that addresses the gap in knowledge identified by the SLR. This research involves the design and execution of empirical investigations that determine the effectiveness of a robot simulator to support the learning of introductory programming concepts. Other literature that has significantly influenced this research project, including that related to education and research design, is described in alternative areas of this thesis.

Chapter Three

Exploratory Studies

The Literature Review established the need to investigate the effectiveness of using simulated robots as tools to support the learning of programming. This is because limited research has considered such technology as a programming teaching aid. After completion of the SLR, computer software that replicates the movement and behaviour of robots virtually was used during exploratory empirical studies. This chapter provides details of these activities and the outcomes of two day-long workshops, involving 23 trainee high school teachers and 10 in-service high school staff, are discussed. During the sessions an existing robot simulator was used to introduce programming concepts. Pre- and post-workshop questionnaires were used to collect participants' views on the potential of simulated robots as a means of supporting the learning of programming. Potential improvements to such a tool were established. Perceptions of simulated robots were positive despite the limitations of the workshop and simulator. It was observed that, despite a number having programming experience, many participants had difficulty completing rudimentary programming tasks. Furthermore, whilst most felt programming should be taught in high schools, less than half of participants said they had the confidence to teach the subject. Lessons learned during the exploratory studies influenced the research reported in the remainder of this thesis.

3.1 Introduction

The work reported in this chapter had three aims:

1. To investigate the potential of using simulated robots to support the teaching of introductory programming concepts
2. To investigate what perceptions trainee ICT/CS teachers, and in-service ICT/CS high school staff, had about programming
3. To gain experience conducting empirical research, and developing instruments for research purposes, in advance of later studies

Two studies were performed and existing computer software was used in workshops involving trainee ($n = 23$), and in-service ($n = 10$), Information and Communication Technology (ICT)/Computer Science (CS) high school staff. These sessions offered a preliminary investigation into the use of simulated robots to teach programming. This research was undertaken to gain feedback on how best to develop a robot simulator and associated workshop for future studies. The workshops had an informal research design. Nonetheless, lessons learned had a significant influence upon the research activities that followed and important observations were made.

Previous work (Drummond, 2009) suggested that it would be interesting to study teachers' perceptions of computing. It was considered it would be valuable to investigate this given that the teaching of CS and ICT in UK high schools has been undergoing a period of reform (McAuley, 2012). Traditionally, CS has not been considered as a core curriculum subject in the UK (Drummond, 2009). This has resulted in most high school students not being introduced to programming (Carter, 2006). From 2014, however, computing will be regarded as a science in high schools and teaching approaches will have to be transformed¹. This is despite research identifying problems that may occur due to such an overhaul (Wells, 2012). Work reported in this chapter has

¹ Coughlan, S. (4th February 2013). Computer science part of English Baccalaureate. *BBC News*. <http://www.bbc.co.uk/news/education-21261442> (Accessed 9th October 2013)

allowed evidence to be collected from educators who, due to these reforms, will be required to teach programming in the future. Some of the work presented in this chapter has been disseminated at the Koli Calling Computing Education Research Conference (Major *et al.*, 2011b).

3.2 Method

No specific experimental methodology was adopted during either exploratory study and this research was not designed to be replicated. The format that was followed is outlined in Section 3.2.3. The studies allowed an opportunity to seek an initial insight, to gain feedback and to help with the generation of ideas. Other reasons why this work was organised was because it:

- Provided an opportunity to gain experience arranging and conducting empirical research involving a number of participants
- Allowed the chance to practice the development and use of data collection instruments for research purposes
- Helped to identify potential problems that could have had a detrimental impact upon more critical empirical investigations that would come later in the research project
- Was led primarily by the PhD supervisor (Theocharis Kyriacou - TK) with support from the author (LM). This afforded an opportunity to learn from, and be guided by, a researcher with greater teaching experience
- Helped to identify improvements/changes to the existing robot simulator

3.2.1 Overall Approach

Details of the approach taken are outlined in this sub-section.

Pre-Existing Robot Simulator (PERS)

The robot simulator used during the exploratory studies was designed to replicate the behaviour of the Mark III robot sold by the Portland Area Robotics Society². The Mark III is a general purpose miniature mobile robot which costs approximately £60/\$100 (unassembled). Pictured in Figure 3-1 is a Mark III robot while in Figure 3-2 the Mark III topology is displayed. The simulator is referred to in this thesis as the Pre-Existing Robot Simulator (PERS). The PERS is written in Java.

The acronym PERS is used to differentiate between two robot simulators used to achieve the objectives of this thesis. The simulator used during the exploratory studies has no official name. The PERS was developed in the School of Computing and Mathematics at Keele University (by TK) in 2008. Initially developed for robotic research and other purposes, the software was created without any systematic research into the teaching of programming and thus lacked significant features and uses in this context. The PERS has, however, been used in several widening participation activities that aimed to demonstrate the possibilities of robotics and computer programming to school-aged children. The first version of the PERS was released to the public in 2009 followed by an update in 2011. The simulator is freely available for download and the software licence permits unrestricted use and modification³. The PERS was developed before more substantial empirical research (reported in the chapters that follow) was undertaken. An adapted version of the PERS, named Kebot, was used during this later research. Modifications to the PERS (completed by the author) were undertaken after considering educational software guidelines, other literature and lessons learned during the exploratory studies. As a result of this, a significant number of new features were introduced and unnecessary features removed. The changes made, in addition to the reasons for them, are presented in Chapter Four.

² <http://www.junun.org/MarkIII> (Accessed 9th October 2013)

³ http://www.scm.keele.ac.uk/staff/1_major/files.php

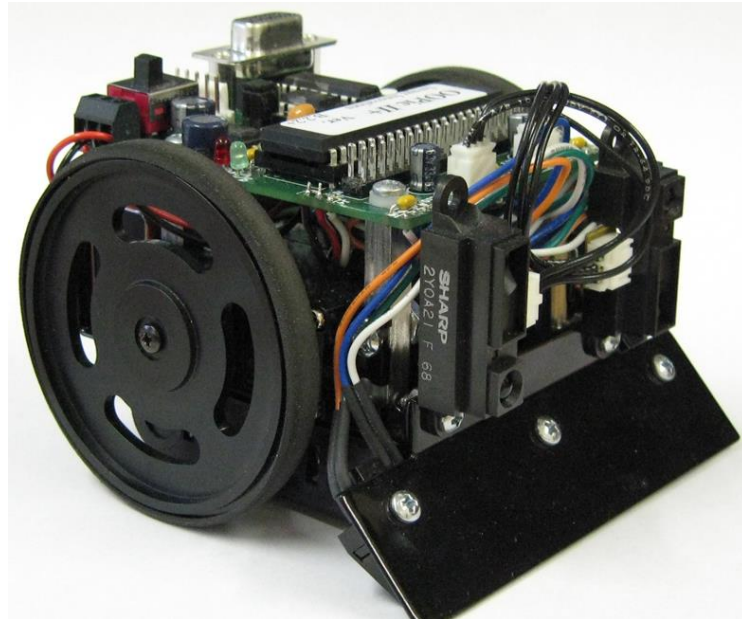
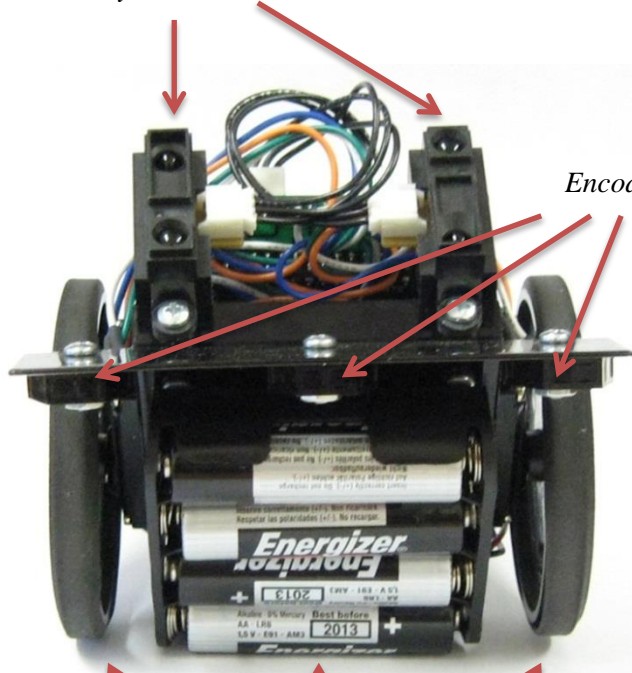


Figure 3-1. A photograph of a Mark III robot after which the Pre-Existing Robot Simulator is modelled (Width - 10cm, Length - 10cm, Height - 8cm).

Proximity Sensors



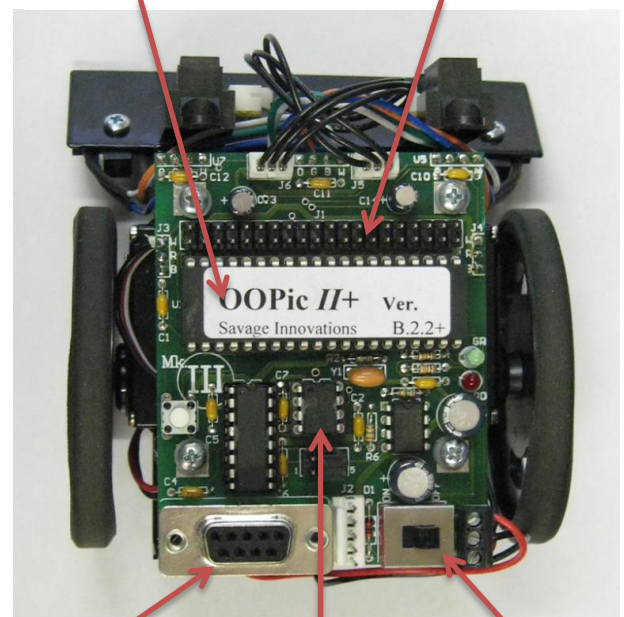
Encoder Sensors

Power Sources

Actuators (Wheels)

Microcontroller

Expansion Port



Programming Port

Long-term Memory

Power Switch

Figure 3-2. Mark III Robot Topology.

The BlueJ⁴ Integrated Development Environment (IDE) is used with the PERS. BlueJ has been employed on CS courses offered at Keele University since 2005, and both workshop leaders (TK and LM) have knowledge of the tool. When the PERS is loaded in BlueJ users are presented with a class diagram as pictured in Figure 3-3. Classes associated with the PERS are visible including those related to sensors (EncoderSensor, RangeScanner, InfraRedProximitySensor) and classes required for the simulator to function (FileExtensionFilter, GraphicsMethods). The user can access all classes although it is only intended for code to be placed in one of four simulated robotic agents classes (named Newton, Ada, Marie, Albert). These agent classes inherit the attributes of one of two sensor classes (MarkIII or MarkIIIRadar) that in turn are subordinate to the Robot class. To launch the simulator the `main()` method is selected after clicking on the RobotSimulation class.

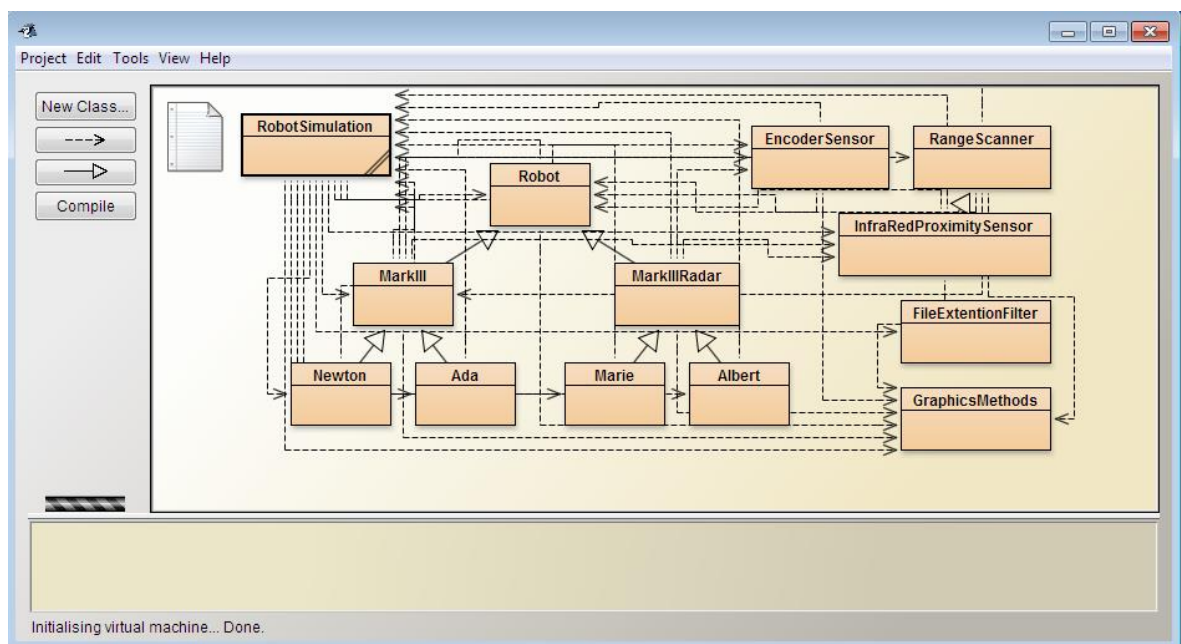


Figure 3-3. The Pre-Existing Robot Simulator class diagram when loaded in the BlueJ IDE.

Three pre-programmed methods can be called by the user to control the four robotic agents used:

`setTVelocity` - Sets the translational velocity of the robot. Values passed to this method can be positive (to move forwards) or negative (to move backwards)

⁴ <http://www.bluej.org>

`setRVelocity` - Sets the rotational velocity of the robot. Values passed can also be positive (to rotate to the left) or negative (to rotate to the right)

`delay` - Causes a delay in execution which prolongs the previous instruction, effectively allowing time for translational or rotational movement to be specified (e.g. move forward for five seconds)

When the simulator is loaded, a blank arena (into which the robotic agents can be placed), and a variety of options, are presented. The arena is viewed from a two dimensional (2D), top down, perspective. In Figure 3-4 key features of the PERS Graphical User Interface (GUI) are presented. Once a robot has been selected from the robot menu, it is placed into the arena by clicking on the desired location. The simulation is run by clicking 'Start'. The corresponding code of the selected robot class is then executed and the simulated agent performs the programmed instructions.

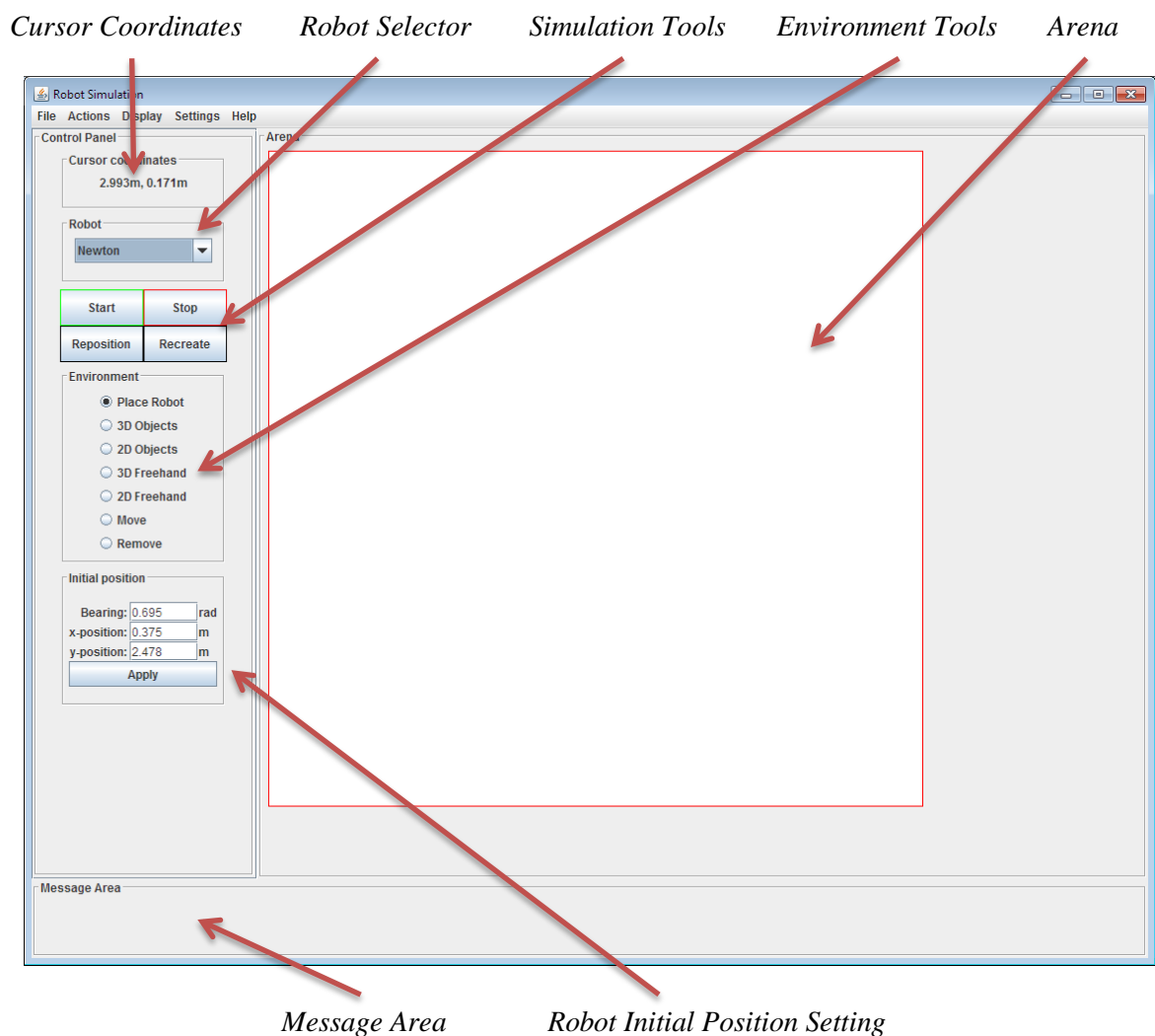


Figure 3-4. Annotated Screenshot of the Pre-Existing Robot Simulator GUI.

The PERS offers a number of tools and features that can be accessed from the GUI, or drop-down menus, including:

- *Cursor Coordinates*: Details the location of the cursor to one thousandth of a metre (1mm)
- *Grid*: Displays an overlay grid to aid the positioning of robots and/or drawing of objects. The ability to increase/decrease grid spacing is possible
- *Initial Position Setting*: Allows the robot to be positioned precisely (to 1mm) and the bearing (the direction the simulated agent faces) to be set (to one thousandth of a radian)
- *Message Area*: A permanently displayed text area for system and other messages
- *Sensor Coverage*: Displays the range of a robot's proximity sensors
- *Trajectories*: Allows a robot's path to be viewed. Trajectories can be saved and loaded
- *Velocity Error*: Allows a random error to affect a simulated agent's performance
- *2D/3D Pixel Width*: Allows the user to define the width of the freehand drawing tools (between a range of 1mm and 1.0m)

All simulated agents are modelled after the Mark III robot (pictured in Figure 3-1) and the proximity sensors, encoder sensors and actuators have been reproduced. The simulated agents have restrictions on maximum translation, and rotation, speeds (0.3 metres per second and 3.14 radians per second respectively). The simulated robots can navigate all areas of their environment and are not limited to moving in set directions (as is the case with Karel the Robot's grid-based world discussed in the Literature Review chapter). The robot arena itself replicates a 3 metre by 3 metre (3m x 3m) square. The arena background cannot be modified and remains permanently white although it is possible to introduce objects. 2D objects akin to flat lines or surfaces can be drawn by the user on the arena floor (over which robots can traverse) as can 3D objects (which act as impassable walls and must be avoided for the simulation not to end). A robot's sensors are able to detect such objects and can modify its behaviour based on them (encoder sensors detect whether 2D objects are present while proximity sensors detect 3D objects within a range of 0.8 metres). Both 2D and 3D objects are drawn using environment tools which act in a manner similar to a basic

‘paint’ software application. In Figure 3-5 a screenshot of the PERS, with a simulated agent and 2D (coloured black) and 3D (coloured grey) objects drawn in the arena, can be seen.

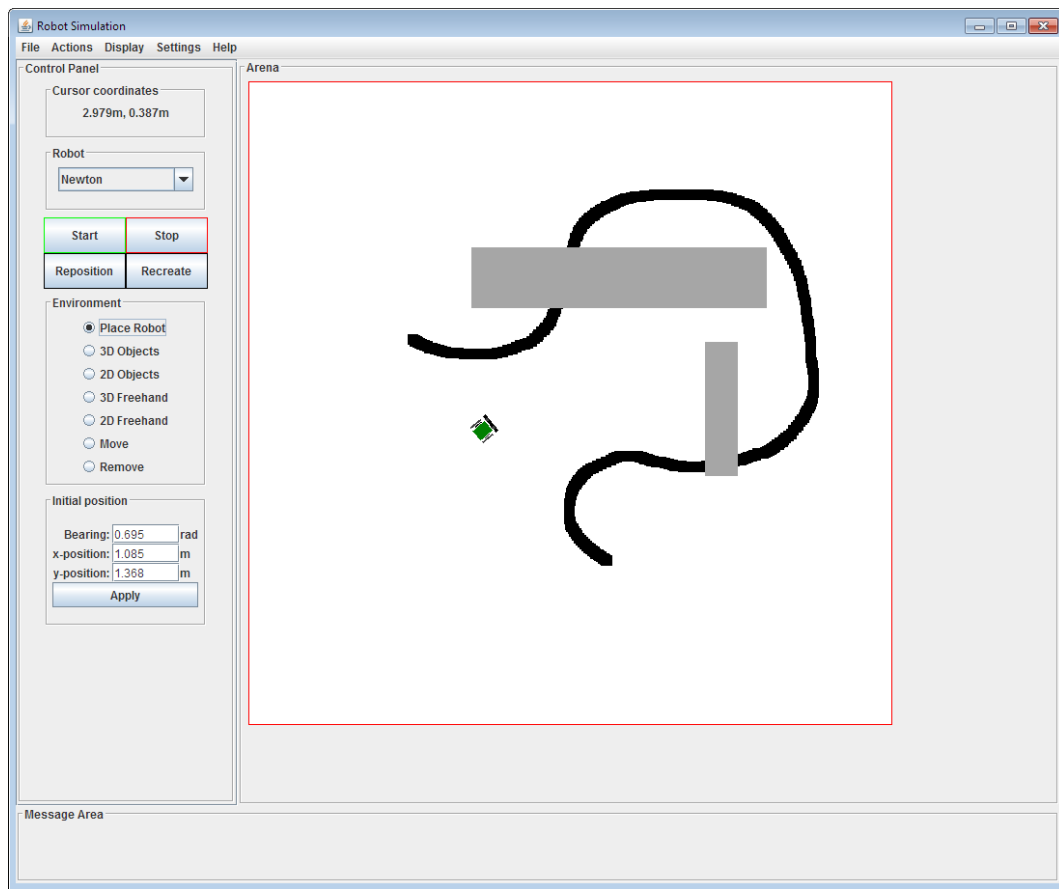


Figure 3-5. Screenshot of the Pre-Existing Robot Simulator with a simulated agent, and examples of 2D and 3D objects, visible in the arena.

The Workshop

The workshops run during the exploratory studies were led by the PhD supervisor (TK), with support from the author (LM), as this allowed a chance to observe a researcher with greater teaching experience introducing fundamental programming concepts. The workshops took place over a single day and lasted around five hours. The length of the workshops was decided on after holding discussions with participants about their availability and what best suited them.

Using the PERS as a teaching aid topics introduced included:

1. Simple expressions and basic Java syntax
2. Program flow control and sequence
3. The fundamentals of iteration and selection
4. The calling of a limited number of pre-programmed methods

Prior teaching experience dictated the order in which concepts were introduced and the workshop procedure is outlined in Section 3.2.3. Concepts delivered were not selected according to the results of any past research. Instead they were chosen due to: the limitations of the PERS (as, in the context of teaching programming, the software was known to have a number of restrictions); past teaching and learning experience of the workshop leaders (as this knowledge could be adapted to support participants during the workshop); time constraints of participants (as this limited how long the sessions could be and what could be taught). Prior to performing this exploratory work the factors outlined were not considered likely to affect the purpose of undertaking it. This is because the research aimed only to gain an initial insight into the potential of using a robot simulator to support the teaching of programming. A variety of tasks and challenges were devised to introduce concepts. The same format was followed during both workshops and the same presentation was delivered. This presentation is available online⁵.

Ethical Considerations

To ensure that the research thesis remained ethical, an application was submitted to the Keele University Ethical Review Panel (KUERP). It is KUERP's duty to assess whether a proposed application of research methods is acceptable ethically. A number of steps were taken to ensure ethical clearance was secured. Full ethical approval was first granted in November 2010 and was later updated in April 2012. This demonstrates how the research design is ethically sound. The relevant Approval Confirmation Letter can be viewed in the Appendices.

⁵ http://www.scm.keele.ac.uk/staff/l_major/files.php

3.2.2 Data Sources

Two questionnaires were used to address the aims identified in Section 3.1. Copies of these questionnaires can be found in Appendix A3. These instruments were developed after studying literature related to questionnaire design (Oppenheim, 2000) and by consulting the other workshop leader:

Exploratory Questionnaire One (EQ1) – Consisting of open and closed questions EQ1 was distributed before the workshop to determine participants' thoughts on, and previous experience of, programming. This questionnaire was devised to determine participants:

- Level of past programming experience
- Familiarity with programming
- Enjoyment of past programming experience
- Opinion on how difficult their past programming experience was
- Familiarity with introductory programming concepts
- Thoughts on the current teaching of programming in high schools
- Own confidence in their ability to teach programming in a high school
- Perceived difficulty teaching programming using their existing knowledge
- Gender

Exploratory Questionnaire Two (EQ2) – Consisting of open and closed questions EQ2 was distributed after the workshop to determine participants’ thoughts on, and experience of, using the PERS as a tool to support the learning of introductory programming. EQ2 was devised to determine participants:

- Enjoyment of their programming experience during the workshop
- Difficulty in completing the programming tasks set during the workshop
- Thoughts on the workshop session in general
- Own confidence in their ability to teach programming in a high school
- Opinion on whether they would consider using a robot simulator in their own lessons
- Opinions on the aspects of the PERS they liked the least and the most
- Thoughts on a robot simulator as a tool to introduce basic programming concepts to high school pupils
- Perceived difficulty teaching programming in a high school having now seen a tool such as the PERS

Supplementary sources of information were also used during the exploratory studies including:

- Observational data (informally observing participants’ conduct and making notes for future reference)
- Discussion data (informal discussions with participants to gain a further insight into views on a simulator approach and associated workshop)
- Researcher Discussions (the two workshop leaders discussing their experience running the sessions in addition to ways that they may be improved)

3.2.3 Workshop Procedure and Setting

Both workshops were held in a computer laboratory located in the School of Computing and Mathematics at Keele University. This laboratory has 19 PCs. Each workshop took place over a single day and lasted five hours. Upon entering the laboratory participants were shown to a PC, read an information sheet (containing details about the research) and completed a consent form. An opportunity to ask questions was offered and it was made clear that withdrawal from the study was possible at any time. Code numbers were not assigned to participants. Once participation had been confirmed, copies of the first questionnaire, EQ1, were distributed. The questionnaires were completed anonymously and posted into a sealed box. A presentation of around 45 minutes then took place. This introduced robotics and the PERS. Instructions on how to use the PERS were then provided before the BlueJ IDE was demonstrated. Once these preparatory steps were completed the PERS was used, to demonstrate programming concepts, before participants attempted to solve a range of previously devised challenges using the software. After each task was attempted the group was reconvened to discuss solutions to the problem set. A number of challenges were completed and included instructing simulated robots to: complete a ‘figure of eight’; follow objects by programming following behaviour; avoid objects by programming avoiding behaviour; seek out objects by programming seeking behaviour. Figure 3-6 shows examples of some challenges. In Figure 3-7 a solution to one of the tasks completed (instructing avoiding behaviour) is presented. This solution is written in Java and demonstrates the kind of programs created. Experimentation and discussion were encouraged to prompt feedback that would determine the potential of using a robot simulator as a programming teaching aid. Informal notes and observations were made by the author to improve the design and conduct of future studies. The two workshop leaders were available to offer assistance when necessary. Once all programming challenges were completed the second questionnaire was distributed (EQ2). The session was then concluded.

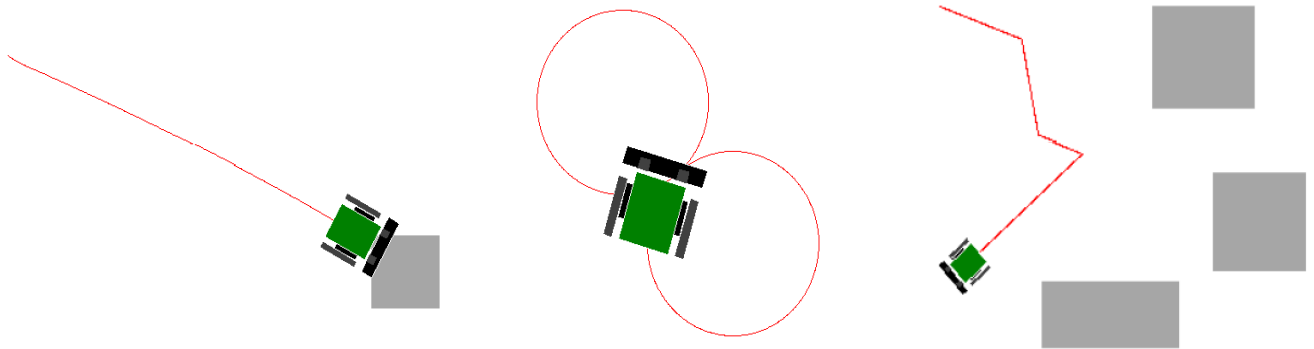


Figure 3-6. Examples of some of the challenges completed by participants
 (Left-to-right: An agent exhibiting 'seeking' behaviour, an agent performing a 'figure of eight', an agent exhibiting 'avoiding' behaviour).

```
double leftPSValue = getPSensorValue(0);
double rightPSValue = getPSensorValue(1);
```

```
if(leftPSValue < 0.3)
{
    setTVelocity(0.0);
    setRVelocity(-1.0);
    delay(1.0);
}
else if(rightPSValue < 0.3)
{
    setTVelocity(0.0);
    setRVelocity(1.0);
    delay(1.0);
}
else
{
    setTVelocity(0.1);
    setRVelocity(0.0);
}
```

Declaration of **Left** and **Right** (Proximity) Sensors

If **Left** Sensor detects an object within 0.3 meters

Stop moving
 Rotate 1.0 radian to the right
 For 1 second

Else if **Right** Sensor detects an object within 0.3 meters

Stop moving
 Rotate 1.0 radian to the left
 For 1 second

Else

Move forward 0.1 meters per second
 Do not rotate

Figure 3-7. One solution to the 'avoiding behaviour' task set during the workshop (with simple explanation).

3.2.4 Data Collection and Analysis

As presented in the following sub-sections, both questionnaires (EQ1 and EQ2) have been subject to quantitative and qualitative analysis and descriptive statistics, figures and the tabulation of data have been used. More rigorous statistical tools have not been employed as the exploratory studies were designed to be an initial investigation rather than a comprehensive one. Moreover, as a large proportion of collected data are examples of participants' subjective opinions, a mainly qualitative approach was considered better suited. Responses to 'open' questions (where participants answered in their own words and not from a pre-defined selection of responses) were analysed by identifying trends in the replies received and by categorising this information into groups. Quotes have been used to illustrate examples of identified trends.

3.2.5 Limitations of the Exploratory Studies

The exploratory studies did not encompass a systematic search for rival explanations. Rival explanations are defined as alternative themes that may be responsible for the results generated during a study (Yin, 2009). By not embedding a search for rivals in the study design there is a risk that results may have been influenced by bias or other factors. The design and wording of the questionnaire instruments may also be considered as a limitation. This is because some questions may be potentially viewed as 'leading', and as restricted choice questions (where only a limited number of responses are available) may have compelled some participants to select an option that did not accurately reflect their thoughts. As participants were instructed not to respond if they felt unable to answer, however, the risk of such factors having a significant impact has been minimised. Another potential issue is that questionnaires are examples of self-reported data and this can lead to issues such as selective memory and exaggeration. The fact that the questionnaires were not trialled in advance could have also led to issues during the workshops and when later attempting to interpret data. Neither the robot simulator, nor workshop, were developed based on past research or were influenced by any formal guidelines. Informal sources of data (such as non-systematically

recorded observations by the workshop leaders) have been used to draw conclusions. Such information is not verifiable and could have been subject to misinterpretation or mistake when being recorded. Finally, all participants volunteered to be involved in the research and this could indicate how those taking part are more motivated to learn about programming than other similar groups. These limitations are valid and warrant consideration. As the exploratory studies were only intended to offer an insight in the use of a robot simulator as programming teaching aid, however, the limitations are not considered capable of having a detrimental effect on the results generated. Reasons why this is the case are outlined in Section 3.5. A more complete discussion on the limitations and threats to validity of empirical studies that involve the use of questionnaires is offered in Chapter Eight.

3.3 Exploratory Workshop One (EW1)

In this section details of the execution and results of the first exploratory workshop (EW1) are presented.

3.3.1 Study Execution

23 trainee teachers (15 male, eight female) enrolled on an ICT/CS Postgraduate Certificate in Education (PGCE) course at Keele University (2010/2011 Academic Year) took part in EW1. All participants were expected to qualify to deliver ICT and/or CS material in high schools six months after the workshop took place. All had gained classroom experience by the time of the study. The PGCE course co-ordinator at Keele University was contacted to invite the participants. EW1 took place in April 2011 and last around five hours. Due to a higher number of participants attending the session than was anticipated, six had to share three PCs between them while the remaining 17 were allocated a PC each. No other deviations to the workshop procedure occurred.

3.3.2 Results - Exploratory Questionnaire One (EQ1)

EQ1 was completed by participants before the workshop. This questionnaire aimed to determine past programming experience and opinions on the teaching of programming.

Participants' Past Programming Experience

Participants were asked how many of the following statements applied to them:

1. "I have no past programming experience"
2. "I have programmed because a university (or other) course required me to"
3. "I have programmed as a past job required me to"
4. "I have previously programmed out of personal interest (e.g. as a hobby)"
5. "I have previously programmed to learn a new skill"
6. Other (*specify*)

If Option One was chosen the first part of the questionnaire was to be skipped. 22 participants had programming experience and completed the questionnaire in full. Of these, 19 had programmed as part of a university (or other) course, five had programmed out of personal interest, three had programmed during a past job, three had programmed to learn a new skill. Participants were asked to specify all programming languages they were familiar with, and their self-perceived knowledge of each (Beginner, Intermediate, Expert), as documented in Table 3-1.

<i>Language</i>	Beginner	Intermediate	Expert	Total Number of Participants With Experience
Java	14	1	0	15
VB.Net / VB / VBA	5	8	0	13
C / C# / C++	6	5	0	11
HTML / XHTML	3	1	0	4
ActionScript	1	2	0	3
SQL	1	1	1	3
PhP	0	2	0	2
Javascript	1	1	0	2
Algol	1	0	0	1
Basic	1	0	0	1
Pascal	1	0	0	1
Python	1	0	0	1
ASP.Net	0	1	0	1
Cobol	0	1	0	1
SmallTalk	0	1	0	1
Powerhouse	0	0	1	1

Table 3-1. Programming languages EW1 participants had experience using arranged according to their self-perceived knowledge of each.

Of the 22 participants with programming experience the minimum number of languages used was one (reported by seven participants all of whom had used Java) and the maximum number was six (reported by two participants). The mean number of languages previously used was 2.9.

To determine familiarity with programming fundamentals participants were asked to select which concepts they had used. Results can be seen in Table 3-2.

<i>Programming Concept</i>	<i>Total Number of Participants</i>
Strings	19
Variables	19
Iteration (e.g. while loops, for-loops)	19
Arrays	17
Object-oriented programming	17
Selection (e.g. if, if...else)	17
File Input/output	16
Methods/Functions/Subroutines	15
Graphical User Interfaces (GUIs)	13
Recursion	8
Expressions	6
Multithreading	3

Table 3-2. Programming concepts arranged by the number of EW1 participants who stated they had used each concept in their past code.

Participants' experience of learning programming is documented in Table 3-3.

<i>“Did you find programming challenging?”</i>			
	<i>Challenging</i>	<i>Neither challenging nor trivial</i>	<i>Trivial</i>
Number of Responses	19	3	0

<i>“Which of the following best describes your past programming experience?”</i>			
	<i>Didn't Like</i>	<i>Indifferent</i>	<i>Enjoyed</i>
Number of Responses	5	8	9

Table 3-3. EW1 participants' experience of learning programming.

Participants’ Opinions on Teaching High School Students Programming

The second section of EQ1 focused on opinions of teaching programming and was completed by all 23 participants. Two questions determined thoughts on the importance of teaching programming in high schools in addition to establishing participants’ confidence teaching the subject. Participants were also asked how difficult they felt it would be to teach elementary programming concepts using their existing knowledge. Responses can be seen in Table 3-4.

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	19	4	0
%	83%	17%	0%

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	10	8	5
%	43%	35%	22%

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Number of Responses	14	6	1	2
%	61%	26%	4%	8%

Table 3-4. EW1 participants’ opinions on teaching high school students programming (pre-EW1).

3.3.3 Results - Exploratory Questionnaire Two (EQ2)

EQ2, which was completed after the exploratory workshop, aimed to evaluate the potential effectiveness of a robot simulator as a programming teaching tool and to determine what effect the session had upon participants' opinions of programming.

Participants' Opinions of a Robot Simulator as a Tool to Teach Programming

Participants were asked, "In regards to today's programming experience, which of the following best describes your enjoyment of today's session?". Responses are displayed in Figure 3-9. Participants were also asked, "Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught today, to novice programming students?". Collected data is presented in Figure 3-8.

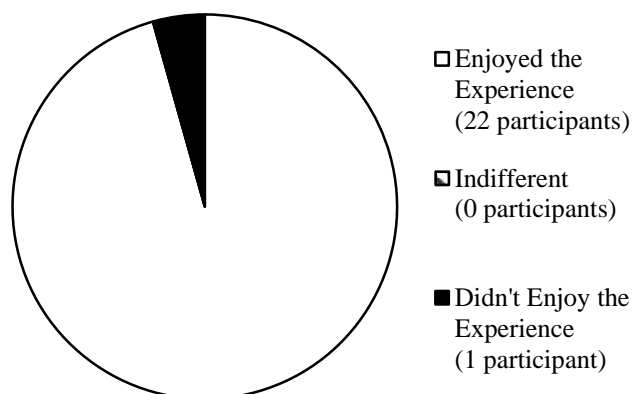


Figure 3-9. EW1 participants' enjoyment of the programming experience using the PERS.

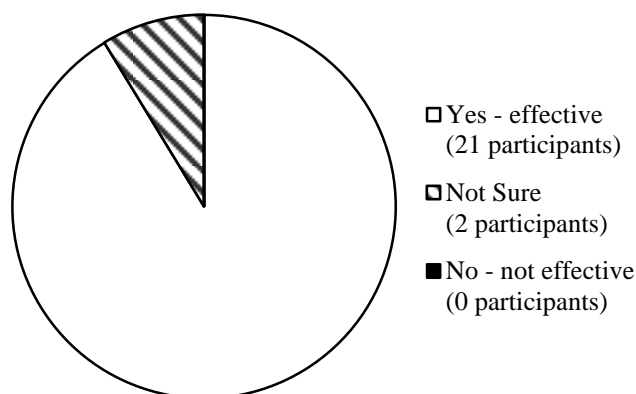


Figure 3-8. The views of EW1 participants on whether such a robot simulator offers an effective method of introducing programming concepts to novices.

As shown in Table 3-5, participants were then questioned on how challenging they felt the tasks set during the workshop to have been and whether they would consider using the simulator.

<i>“In regards to today’s session, which of the following best describes how challenging the programming tasks have been?”</i>			
	<i>Difficult</i>	<i>Neither easy nor difficult</i>	<i>Easy</i>
Number of Responses	12	11	0

<i>“Would you consider using the Robot Simulator as a tool to teach programming in your own lessons in the future?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	22	1	0

Table 3-5. EW1 participants views on the difficulty of the workshop tasks and whether they would use the PERS in the future.

Finally, as shown in in Table 3-6, participants were instructed to score on a scale of one (not at all effective) to five (extremely effective) their thoughts on the effectiveness of the: PERS, programming support received, presentation delivered and the teaching environment.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
Robot Simulator	4.57	0	0	0	10	13
Programming Support	4.35	0	0	5	5	13
Workshop Presentation	4.13	0	1	4	9	9
Environment	3.96	0	3	2	11	7

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 3-6. EW1 participants' opinions on the effectiveness of elements of the workshop (arranged by the number of participants who selected each option).

Participants’ Opinions on Teaching High School Students Programming

EQ2 contained two questions also used on the EQ1 questionnaire. Participants were again asked, “With your current knowledge would you be confident teaching high school students about introductory programming concepts?”. In addition, participants were again questioned on whether they felt it would be difficult to teach elementary programming concepts to high school students. See Table 3-7 for participants’ responses and comparison to those received in response to EQ1.

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>				
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>	
Number of Responses	13	6	4	
%	57%	26%	17%	
Change from ISQ1 Responses	+3	-2	-1	

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Number of Responses	13	7	2	1
%	57%	30%	8%	4%
Change from ISQ1 Responses	-1	+1	+1	-1

Table 3-7. EW1 participants opinions on teaching high school students programming (post-EW1).

Analysis of Participants’ Opinions of a Robot Simulator to Teach Programming

The final three questions of EQ2 aimed to identify:

1. Which aspects of the robot simulator (if any) were the most liked
2. Which aspects of the robot simulator (if any) were the least liked
3. General opinion of the workshop session

These were ‘open’ response questions whereby participants could reply in their own words. A response was not compulsory. After analysis several trends have been identified. These are illustrated by using quotes provided.

The idea that the robot simulator was enjoyable to use was mentioned with comments such as, *“Fun and a good challenge”*. The visual and tangible nature of the robotic agents also appeared to help engagement with the programming tasks, *“The simulator provides visual feedback on erroneous code which is a very useful tool to identify and correct errors”* and, *“... you can see the end results to see if changes need to be made”*. The accessibility of the approach and that simulated

agents, “... *allowed you to see easily what your programming did*” was also stated. How high school students would get “*immediate results*” from the simulator and that the programming ideas which underpin the software would, “... *challenge the pupils*” was also commented on. Four participants remarked how the simulator and its interface helped to make programming concepts more understandable as it was good “*seeing programming and its effects*”.

In regards to aspects of the robot simulator that were least liked the fact that the, “*interface is initially bewildering... (and that) variables used could have more user friendly names*” was mentioned. Aspects of the PERS, such as the use of bearing and positioning functions, were specifically highlighted as potentially “*confusing for students*”. Some participants were, “... *not sure when the program had started... (and that) some indication would be good*”. Several also stated concerns about the suitability of the simulator for all high school students. One participant responded that Java, “... *was a complex language... particularly for low literacy students*”. Another suggested they were, “... *not sure whether the tasks would be too difficult for lower Key Stage 3 students (aged 11 – 14)*”. The structure of the session was found to be a cause for concern, “*Activities need to come quicker in the lecture, less activities and more time on them*”.

The overall opinion of robot simulator and the workshop was positive. It was commented how the workshop was, “... *excellent – best session we have had on the PGCE*” while another stated the simulator “*links well to teaching A-Level ICT*”⁶. The workshop itself was described as being a, “... *very good session... I would use many of the ideas to present to pupils in my class*”. In regards to the teaching of programming one participant said how they had, “... *struggled with Java previously but the (session) was well taught and (the simulator) engaging*”. Another commented that they would have liked, “... *to know how to get the program onto robots*”. These sentiments were shared by several participants and the potential implications of this are considered in Section 3.5.

⁶ A-Levels are the UK’s main post-high school and pre-University qualification and are normally undertaken by students aged 16 to 18 years old.

3.4 Exploratory Workshop Two (EW2)

In this section results of the second exploratory workshop (EW2) are presented.

3.4.1 Study Execution

10 high school staff (five male, five female), all of whom were employed at a local high school in either a teaching ($n = 7$) or technical support ($n = 3$) capacity, took part. Three of the teachers (who were all qualified) were male and four female while two of the technicians were male with the other female. Data relating to specific aspects of teaching, collected from non-teaching staff, has been omitted from the analysis. One of the teachers involved made initial contact with the author about the prospect of taking part in the research.

EW2 took place in September 2011. Each participant was assigned to an individual PC. EW2 lasted around four hours 15 minutes (45 minutes less than EW1 as some of those taking part had prior commitments). The same content was covered as during EW1, however, as EW2 participants stated that they were happy to take fewer breaks. No other deviations to the procedure occurred.

3.4.2 Results - Exploratory Questionnaire One (EQ1)

Before the workshop participants completed EQ1. The same questionnaire was used as during EW1 to determine past programming experience and opinions on the teaching of programming in high schools. Data collected from teaching and technical support staff is presented.

Participants' Past Programming Experience

Five of the seven teachers who took part in EW2 had no programming experience and, therefore, did not complete the first part of EQ1. The two remaining teachers had experience in 11 programming languages between them. Both had learned programming as part of a university (or other) course. One teacher had used Java and SQL and described their knowledge as 'Beginner' (out of a choice of 'Beginner', 'Intermediate' and 'Expert'). The other teacher had experience in

nine languages to an intermediate level (Assembler, Basic, C, Cobol, Forth, Fortran, ML, Pascal, Perl) and had also programmed out of personal interest and during previous employment.

Two of the technicians who took part in EW2 had never programmed before the workshop (and skipped part one of EQ1). The remaining technician had programmed due to the requirements of a past job and out of personal interest. This technician had experience using four languages (HTML, PHP, PL/SQL, VB.Net) and described their self-perceived competence in each as being ‘Intermediate’ (with the exception of VB.Net which they described as ‘Beginner’).

In Table 3-8 responses from EW2 participants with programming experience, in regards to their enjoyment and perceived difficulty when previously programming, are displayed. In Table 3-9 details of programming concepts these participants had previously used can be seen.

<i>“Did you find programming challenging?”</i>			
	<i>“Challenging”</i>	<i>“Neither challenging nor trivial”</i>	<i>“Trivial”</i>
Teachers (n. 2)	2	0	0
Technicians (n. 1)	1	0	0

<i>“Which of the following best describes your past programming experience?”</i>			
	<i>“Didn’t like programming”</i>	<i>“Indifferent”</i>	<i>“Enjoyed programming”</i>
Teachers (n. 2)	1	0	1
Technicians (n. 1)	0	1	0

Table 3-8. EW2 participants' views on their past programming experience.

<i>Programming Concept</i>	Total Number of Teachers (n. 2)	Total Number of Technicians (n. 1)
Object-oriented programming	2	1
Selection (e.g. if, if...else)	2	1
Methods/Functions/Subroutines	2	1
Strings	1	1
Variables	1	1
Iteration (e.g. for-loops)	1	1
Arrays	1	1
File Input/output	1	1
Graphical User Interfaces (GUIs)	1	1
Expressions	1	1
Multithreading	1	0
Recursion	1	0

Table 3-9. Programming concepts arranged by the number of EW2 participants who stated that they had used each concept in their past code.

Participants' Opinions on Teaching High School Students Programming

The second section of EQ1 focused on opinions of teaching programming. Questions were used to determine views on the importance of teaching programming in high schools, in addition to establishing the confidence and perceived difficulty of teaching staff in their ability to deliver the subject. Responses have been tabulated in Table 3-10.

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>				
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>	
Teachers (n. 7)	5	1	1	
Technicians (n. 3)	3	0	0	
<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>				
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>	
Teachers (n. 7)	2	0	5	
<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Teachers (n. 7)	5	1	0	1

Table 3-10. EW2 participants' opinions on teaching high school students programming (pre-EW2).

3.4.3 Results - Exploratory Questionnaire Two (EQ2)

Participants' Opinions of a Robot Simulator as a Tool to Teach Programming

Responses provided by EW2 participants, in regards to their enjoyment and difficulty of their programming experience during the workshop, can be seen in Table 3-11.

<i>“In regards to today’s programming experience, which of the following best describes your enjoyment of today’s session?”</i>			
	<i>“Enjoyable”</i>	<i>“Indifferent”</i>	<i>“Not Enjoyable”</i>
Teachers (n. 7)	7	0	0
Technicians (n. 3)	3	0	0

<i>“In regards to today’s session, which of the following best describes how challenging the programming tasks have been?”</i>			
	<i>“Difficult”</i>	<i>“Neither easy nor difficult”</i>	<i>“Easy”</i>
Teachers (n. 7)	4	2	1
Technicians (n. 3)	0	3	0

Table 3-11. EW2 participants' enjoyment, and difficulty, of their programming experience during the workshop.

Participants' opinions of the effectiveness of the PERS, as a tool to support the learning of programming, are displayed in Table 3-12. Finally, to establish how effective individual elements of the workshop were perceived to have been, participants were instructed to provide their opinions on the effectiveness of the: PERS, programming support, presentation delivered and the teaching environment. Data collected from teachers and technicians has been combined and is presented in Table 3-13. One technician did not complete this question and this is why only nine responses are documented.

“Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught today, to novice programming students?”

	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Teachers (<i>n.</i> 7)	6	1	0
Technicians (<i>n.</i> 3)	3	0	0

Table 3-12. EW2 participants' opinions on whether such a robot simulator offers an effective method of introducing programming concepts to novices.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
Robot Simulator	4.56	0	0	1	2	6
Programming Support	4.56	0	0	0	4	5
Workshop Presentation	4.44	0	0	0	5	4
Environment	4.67	0	0	0	3	6

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 3-13. EW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.

Participants' Opinions on Teaching High School Students Programming

EQ2 included two questions also used in the EQ1 and these related to the teaching of programming in high school. As before teaching staff were asked about their confidence teaching programming and how difficult they thought this would be. See Table 3-14 for responses in addition to a comparison to those received when the same questions were used on EQ1.

“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”

	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Teachers (<i>n.</i> 7)	2	2	3
Change from EQ1	0	+2	-2

“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”

	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Teachers (<i>n.</i> 7)	4	0	1	2
Change from EQ1	-1	-1	+1	+1

Table 3-14. EW2 participants' opinions on teaching high school students programming (post-EW2).

EW2 participants, who were teachers, were also asked whether they would consider using the robot simulator in their own future lessons. Six participants stated that they would while one indicated that they were unsure.

Analysis of Participants' Opinions of a Robot Simulator to Teach Programming

The final three questions of EQ2 aimed to identify which aspects of the robot simulator were the most liked, which aspects of the robot simulator were the least liked in addition to establishing any general opinions of the workshop session. 'Open' questions were used. Trends in replies have been identified and information categorised. Trends are illustrated by using examples of quotes provided. Teacher and technicians responses have been considered as one.

Two main positives were identified after analysing feedback. These relate to the way in which the robot simulator functions or to the tangible and visual nature of simulated robots. The fun nature of the simulator, in addition to the challenge of, *"being able to make it work"* were highlighted as strengths. Two participants also stated that they enjoyed using the simulator for, *"... trial and error (and) testing things out"*. In regards to the nature of simulated robots, *"being able to ... watch the robots behaviour"*, *"visual results"* and *"see(ing) the outcome on the simulator"* were mentioned.

EW2 participants offered few negatives about the approach. Two participants did, however, mention how they struggled with Java syntax. One also stated that they did not like how they could not, *"move freehand objects"*. General comments were made by some participants in response to the final question on EQ2. These were positive and included:

- *"Thoroughly enjoyable and geared towards the audience. Really enjoyed it"*
- *"Thank you! I enjoyed doing something we don't do often"*
- *"Interesting and challenging"*

3.5 Discussion

As stated in Section 3.1, the exploratory studies had three aims:

1. To investigate the potential of using simulated robots to support the teaching of introductory programming concepts
2. To investigate what perceptions trainee ICT/CS teachers, and in-service ICT/CS high school staff, had about programming
3. To gain experience conducting empirical research, and developing instruments for research purposes, in advance of later larger studies

The research reported in this chapter served as an initial investigation into the viability and potential of using a robot simulator to support the learning of programming. Such work was motivated by the results of a SLR and, as is outlined in this discussion, the findings had implications for research that followed. In Section 3.2.5 the limitations of the exploratory study design are summarised. Before the workshops, it was also identified how the pre-existing robot simulator (PERS) had several shortcomings due to it initially being designed for non-educational purposes. As a result of these limitations the findings documented only offer an insight into the use of a robot simulator, as a tool to support the teaching of introductory programming, and not a thorough evaluation. It was recognised from the outset that more substantial work would have to be undertaken to provide a comprehensive evaluation on the use of a robot simulator as a programming teaching tool. Such work was carried out later in the research project by building on the experience gained conducting the exploratory studies. The exploratory studies, therefore, had a significant influence on the research activities that followed later. In the remainder of this subsection the three exploratory study aims will be considered.

Exploratory Study Aim One: To investigate the potential of using simulated robots to support the teaching of introductory programming concepts

One aim of the exploratory studies was to determine the potential of using a robot simulator to support the teaching of introductory programming concepts. Despite initially being envisaged for alternative purposes, use of a PERS allowed an opportunity to seek an initial insight, gain feedback and to help with the generation of ideas.

The results of EQ2 demonstrate how participants from both groups almost unanimously enjoyed their programming experience using the PERS and that most believe a robot simulator offers a valuable means of introducing programming concepts. In terms of effectiveness, the PERS was scored highly when rated by respondents on a numeric scale. Moreover, the majority of teachers stated that they would consider using such software in their own lessons. The qualitative analysis of responses to open questions provides further evidence that the use of a robot simulator, as a programming teaching aid, was well received. Across both groups participants identified how they believe that a robot simulator is fun and effective to use, offers strong visual feedback, helps to demonstrate the effects of changing code and makes it easy to identify, and correct, programming errors. The BlueJ IDE was also found to be well suited for use during such a workshop although this was expected given that the software is designed to support the introduction of programming. Before the sessions it was anticipated that responses to the PERS would, in the main, likely be good as previously identified literature highlighted how robots offer an enjoyable means of introducing programming. This was because past research suggests that robots appeal to a wide variety of people no matter what their age or experience (Hirst, 2003). Indeed, robots can be enjoyable for both teachers and students to work with (Fagin, 2003). Such technology can also encourage a positive attitude to STEM (Science, Technology, Engineering and Mathematics) subjects (Brauner *et al.*, 2010) which is an important consideration for high school educators looking to deliver a broad curriculum.

As participants volunteered to be involved in the research it is not unreasonable to assume that they may have had an interest in learning about programming, or robots, and would be motivated. It was not predicted, however, that opinion across both groups would be almost entirely positive in regards to the robot simulator approach. The pre-identified limitations of the PERS, coupled with the workshop not being based on the results of previous research, do not appear to have had a negative impact upon participants' views. The exploratory studies are believed to have established the potential and viability of using a robot simulator as a means of supporting the introduction of programming concepts.

Despite these positives, however, in their questionnaire responses participants identified several shortcomings and potential improvements. Other feedback that was provided, in addition to observations that were made, also found this to be the case. The complexities of the PERS were highlighted as the main issues during, and after, the workshop. Specifically participants brought attention to: the initially confusing 'dry' interface; the use of complex variable and method names; the fact that several features of the software were redundant; the fact that it was possible to 'lose' a robot on the screen due to how zoom function could be used; design issues such as not knowing when the simulator was running and inaccessibility of some features (such as the 'Clear Environment' option); the fact that important software messages were often missed. Such factors were mentioned as being potentially confusing for new students to programming, especially considering a substantial amount of new knowledge would already have to be acquired given the complex nature of programming. It is thought that these issues need not be an inherent feature of all robot simulators. Indeed, after the exploratory studies, it was identified how most of the problems could be eliminated (or at least minimised) given modification to the PERS. One issue that was apparent in some participants' responses, however, could be of relevance to all robot simulators that are used to teach programming. During both workshops a number of participants expressed a desire to transfer code they had created onto an actual physical robot. Some voiced disappointment when they were informed that participants would not be doing this during the workshop. Potentially the workshop leaders were not clear when delivering the sessions that the approach was

based entirely on the use of simulated technology and that the coding of physical robots would not take place. It is possible, however, that participants' responses are indicative of how, for some people, the desire to code a simulated robot will always be secondary to the programming of an actual physical robot. Another concern is how some participants questioned the suitability of the software for use by high school students of all abilities. It is believed that the nature of the PERS itself, due to it being overly complex and having redundant features, may be responsible for such feedback. A more restrictive approach initially, with elements of programming and the simulator being introduced incrementally, could counter such concerns. Indeed, a more restricted approach may present the opportunity to strengthen knowledge of basic programming concepts before the introduction of more advanced topics and/or simulator features.

The workshop delivered was well received and almost all participants seemed to benefit from, and enjoy, the experience. The manner in which concepts were introduced (with fundamentals being introduced via a presentation before participants practiced, and discussed, the use of these as a group) was praised by some. Suggested improvements to the structure, however, were noted and these included the use of a less generic presentation and moving more rapidly onto programming tasks. When examining the results of the exploratory studies the issue of the “good subject” effect must be taken into account. This occurs when participants try to determine what a researcher wants and adjust their behaviour (or responses) accordingly (Jackson, 2011). Feedback generated during the workshops could have been influenced by such bias. The group dynamic itself could have also prejudiced the information provided by some (Kitzinger, 1994). This is because peer influence may have led some participants to modify their behaviour.

Exploratory Study Aim Two: To investigate what perceptions trainee ICT/CS teachers, and in-service ICT/CS high school staff, had about programming

The previous programming experience of participants was mixed, although most of the EW1 group had some programming experience whilst the majority of EW2 participants did not. A large

proportion of those involved reported the tasks set during the workshop as being difficult. Indeed, only one participant found the tasks to be ‘easy’. Observations and discussions during the workshop also confirmed this to be the case. This is despite over two-thirds of EW1 participants previously using Java and having collective experience with 22 programming languages. One potential reason why some participants struggled may be due to a lack of understanding about programming. In the case of those with programming experience the reasons for this are unknown, but may be because such participants had not programmed for a long time or as they never truly mastered the concepts in the first place. The effect that the nature of the workshop had upon performance must also be considered. The intensive nature of the sessions, coupled with the fact that a significant amount of information had to be digested at once (as the features of the PERS and the programming concepts taught were introduced concurrently), may be responsible for why some of those involved found the challenges difficult. The PERS itself may have had an effect. Potentially the PERS may have added a layer of complexity in addition to the difficult task of re-familiarisation with, or new acquisition of, programming knowledge. The exploratory studies were not designed to consider whether this was the case, although it was acknowledged that future research could investigate this.

In regards to the teaching of programming in high schools almost all participants believed programming should be taught to computing and ICT students. This demonstrates how participants, in the main, believe programming to be an important subject for students to learn. The majority of teachers involved (whether trainee or qualified), however, stated that they would not be confident teaching basic programming concepts. Such a finding is a cause for concern as it may demonstrate how programming knowledge, and confidence in their ability to teach it, is lacking for some of those who will be responsible for delivering the subject given recent reforms to the teaching of high school CS in the UK. As a final point, if programming is to become an established and well regarded part of the high school curriculum then much needs to be done to prepare educators so that they are confident and able to teach the subject. Workshops that serve to inform high school teachers on the best programming teaching practices, in a manner similar to those undertaken, may

be a positive step in the direction of improving the teaching of the subject in high schools although the effectiveness of such an approach would have to be thoroughly examined. Such workshops may offer an opportunity to inspire educators to teach subjects that they do not ordinarily consider delivering. Recent work supports this hypothesis and in-service teachers have been successfully taught programming concepts during an intensive three-day workshop (Saad *et al.*, 2012).

Exploratory Study Aim Three: To gain experience conducting empirical research, and developing instruments for research purposes, in advance of later larger studies

As a result of performing the work reported in this chapter, experience has been gained by the author in the following aspects:

- In arranging and conducting empirical research, in addition to managing a research project and liaising with potential participants. As programming concepts had to be explained, confidence has also been acquired in the effective presentation of such information to others.
- In developing, and implementing, instruments for the collection of research data. This involved the consideration of visual presentation techniques in addition to ensuring that collected data would help to fulfil the objectives of the research.
- In identifying potential issues that could have had a detrimental impact upon later empirical investigations. Important lessons, such as ensuring that the research environment is properly prepared and capable of hosting the participants involved were learned.
- In observing a more experience researcher (TK) introducing fundamental programming concepts in a workshop environment. The exploratory workshops also allowed a chance to gain informal feedback, from participants, in relation to the structure of such sessions.
- In regards to issues related to participant involvement. This included the consideration of: participant confidentiality, factors related to the storage of personal data in addition to health and safety issues.

3.6 Summary

In this chapter the results of two exploratory empirical studies have been reported. A pre-existing robot simulator (referred to in this thesis as the PERS), that replicates the movement and behaviour of a robot virtually, was used during these. This allowed an opportunity to seek an initial insight, gain feedback and helped with the generation of ideas. Whilst the research design of the exploratory studies was relatively informal, several potentially important findings have been discussed. Two day-long workshops, involving 23 trainee high school teachers and 10 in-service high school staff, were undertaken. During these sessions the PERS was used to introduce programming concepts. Pre- and post-workshop questionnaires have helped to determine participants' thoughts on the potential of simulated robots as a means of supporting the learning of programming. Feedback was also gained on how best to develop such a tool, and associated workshop, in the future. Perceptions of simulated robots were found to be positive despite the limitations of the workshop and robot simulator that were used. Areas for improvement were also noted and later research has been able to build on this. It was observed how, despite a number of participants having programming experience, many had difficulty completing rudimentary programming tasks. Furthermore, whilst most felt that programming should be taught in high schools, less than half of participants said they had the confidence to teach the subject. The exploratory studies have allowed experience to be gained in the conduct of such empirical research. In the following chapter details of how the PERS was developed, to achieve the objectives of this research, are outlined.

Chapter Four

The Kebot Robot Simulator

In this chapter details of a robot simulator, used to achieve the research objectives of this thesis, is presented. Influenced by the findings of the Systematic Literature Review and exploratory studies already discussed, the pre-existing robot simulator was further developed. This modified simulator, named Kebot, allowed for an investigation into the effectiveness of using simulated robots as programming teaching tools. The modifications that were completed are outlined and information about the functionality of Kebot given. The programming concepts that were selected to be taught using Kebot are provided. Reasons why such concepts were chosen are justified.

4.1 Introduction

As outlined in Chapter Three, during the early stages of the research project a pre-existing robot simulator (referred to in this thesis as the PERS) was used during two exploratory empirical studies. Whilst the exploratory studies were modest in terms of scale and design, important lessons were learned. Participants' perceptions of simulated robots were positive despite the limitations of the PERS and the exploratory workshop. To support the objectives of this thesis a modified version of the PERS, named Kebot, has been developed. Kebot is a robot simulator designed to support the learning of introductory programming. Work reported in the remainder of this thesis investigates the effectiveness of Kebot. The name Kebot was derived from two words – Keele Robot (KEele-roBOT).

The decision to use a modified version of the PERS was influenced by the positive feedback provided, in regards to this software, during the exploratory studies. Observation of participants, and discussions between the two workshop leaders, also established the potential of the PERS to support the learning of introductory programming concepts. Modifications were, however, required to this simulator. This is because the software was considered by those who took part in the exploratory studies as being overly complex for use as a programming teaching aid.

4.2 Modifying the Pre-Existing Robot Simulator

In this sub-section details of the modifications made to the pre-existing robot simulator (PERS) are provided.

4.2.1 Advantages of modifying the PERS

During the exploratory studies it was established how, for the purposes of supporting the learning of introductory programming, the PERS did not require a substantial overhaul. It was decided, therefore, that this existing software should be adapted rather than new software created. This strategy was considered most likely to lead to a successful project outcome. This is because it was identified how the most critical component of the research project would be the evaluation of the simulator by multiple participants. Modifications made to the PERS, and the development of the associated programming exercises, took around five months in total. It is predicted that if a new simulator was developed from scratch then this development time would have been at least 12 months. As a result, it is considered that a substantial amount of time has been saved by modifying the PERS and this allowed greater time to be spent recruiting participants and preparing for empirical research.

Use of a modified version of the PERS was considered to have other advantages. The exploratory studies can be viewed as a substantial test of the PERS and this enabled potential issues and improvements to be identified in advance of more substantial, and critical, research. Use of a modified version of the PERS, therefore, minimised the risk that the execution of later research would be impeded by factors related to the robot simulator implemented (such as software bugs or crashes). Potentially, this would have not been the case if an alternative robot simulator (which had not undergone a similar evaluation process) was used. Kebot, like the PERS, is written in Java. The author is familiar with this language and it is also taught extensively on CS courses at Keele University.

4.2.2 Modifications made to the PERS

The changes made to the PERS, which resulted in the robot simulator referred to as Kebot, were undertaken by the author. Information related to the development of educational software informed the interface and design alterations that were made. The following guidance was consulted:

- *Design Guide for Developers of Educational Software* (Beale & Sharples, 2002)
- *Educational Software Systems Guidelines for the Design of Educational Software* (ANSI, 2001)
- *Predicting quality in educational software: Evaluating for learning, usability and the synergy between them* (Squires & Preece, 1999)

As a direct result of discussions with participants during the exploratory studies, other changes were also made. This included, amongst other things, modification of the software so that it worked in degrees (rather than radians) as it was believed that this would establish a link with the Mathematics syllabus taught in UK High Schools. Indeed, it is important for educational software to speak the user's language with words, phrases and concepts that are familiar (Squires & Preece, 1999). In Table 4-1 substantial modifications that were made to the PERS, in addition to justifications for them, are presented. The modifications were made to simplify and optimise the simulator so that it was better suited for educational purposes. In addition, a number of smaller modifications were made and new features and capabilities added. This involved further developing the software so that it would better support the learning of programming and the associated tasks reported in Chapter Five.

Modification Made	Justification for Modification
Simplification of the Graphical User Interface (GUI) by removing features not required for the teaching of introductory programming concepts. This is because, as outlined in Chapter Three, the PERS had a number of features not used during the exploratory workshops including the: ‘Initial Position’ option; ‘Reposition’ button; ‘Cursor co-ordinates’ information; ‘Grid’ and ‘Grid Spacing’ options; ‘Velocity Error’ option; the ability to Save, Clear or Load robot trajectories.	GUI’s should remain simple so only relevant information is presented to the user (Beale & Sharples, 2002). An aesthetic and minimalist design is important and features that are irrelevant or rarely needed should be removed. This is because every extra unit of information in a dialogue competes with relevant units and diminishes their relative visibility (Squires & Preece, 1999). As none of these functions were observed to be used by participants during the exploratory studies, it was considered that they were redundant for the purpose of teaching of introductory programming concepts.
Modification of the robot arena so that it is not scrollable (and, therefore, not zoom-able) and remains a constant size	Scrolling should be avoided when it moves critical information off the bottom of a screen (Squires & Preece, 1999). By removing the ability to scroll the risk that users will “lose their robot”, as was observed to be the case during the exploratory studies, has been eliminated.
Modification of particular GUI elements (e.g. ‘Start’ and ‘Stop’ buttons) to make important information clearer in addition to improving the aesthetic style of the simulator	A pleasing visual style is valuable for users of educational software (Beale & Sharples, 2002). Display techniques should also be used to attract attention to important information and features (ANSI, 2001).
The removal of a permanently displayed ‘Message Area’, which was mostly redundant, and use of visual pop-up messages for critical information instead	Messages must be clearly visible and a user must be aware of important issues (Squires & Preece, 1999). Previously the ‘Message Area’ was constantly on display in the bottom corner of the application GUI. Some participants noted during the exploratory studies how important messages were overlooked when using the simulator. The use of pop-up messages to display important information reduces this risk.
The creation of a ‘Clear Environment’ button on the GUI	Important and widely used options should be visible and the user should not have to remember information from one part of the dialogue to another (Squires & Preece, 1999). As ‘Clear Environment’ was noted to be extensively used by participants during the exploratory studies, this option has now been placed on the main GUI for ease of access.
Use of simpler method and sensor names so that it is clearer what the purpose of each is	Everyday language and an avoidance of technical terms are important factors for users of educational technology (Beale & Sharples, 2002). Several participants also stated how the use of complicated method names led to confusion during the exploratory studies.
Concealment of unnecessary code, removal of the ‘Initialise’ and ‘Finalise’ methods, simplification of the class diagram and obvious visual pointers to guide users	Users new to a topic need simple visual aids that make pointers extremely clear while irrelevant information should not be displayed (ANSI, 2001).

Table 4-1. Main modifications made to the PERS in addition to justification for them.

4.3 Kebot: A Robot Simulator for Supporting the Learning of Introductory Programming

In this sub-section the Kebot simulator is described. Information relating to the programming exercises completed, however, is not provided. This is because details of these exercises are presented along with the study methodology in Chapter Five. Information presented in the following chapter, relating to these programming exercises, helps to further demonstrate the software functionality of the Kebot simulator. As stated previously, the name Kebot was derived from two words - Keele Robot (KEele-roBOT). The BlueJ IDE was used with Kebot. When Kebot is loaded in BlueJ the same screen is shown as in Figure 4-1. User's code is placed in one of five simulated robotic agent classes (named Gates, Berners, Jobs, Gosling, Page). As demonstrated in Chapter Five, each of these robots offers different functionality. An example screenshot of the code editor is shown in Figure 4-3. Right-clicking the Kebot class presents a list of class operations. The simulator is launched by selecting the `main()` method. This class diagram is much simplified compared to the one first presented when the PERS is loaded in BlueJ.

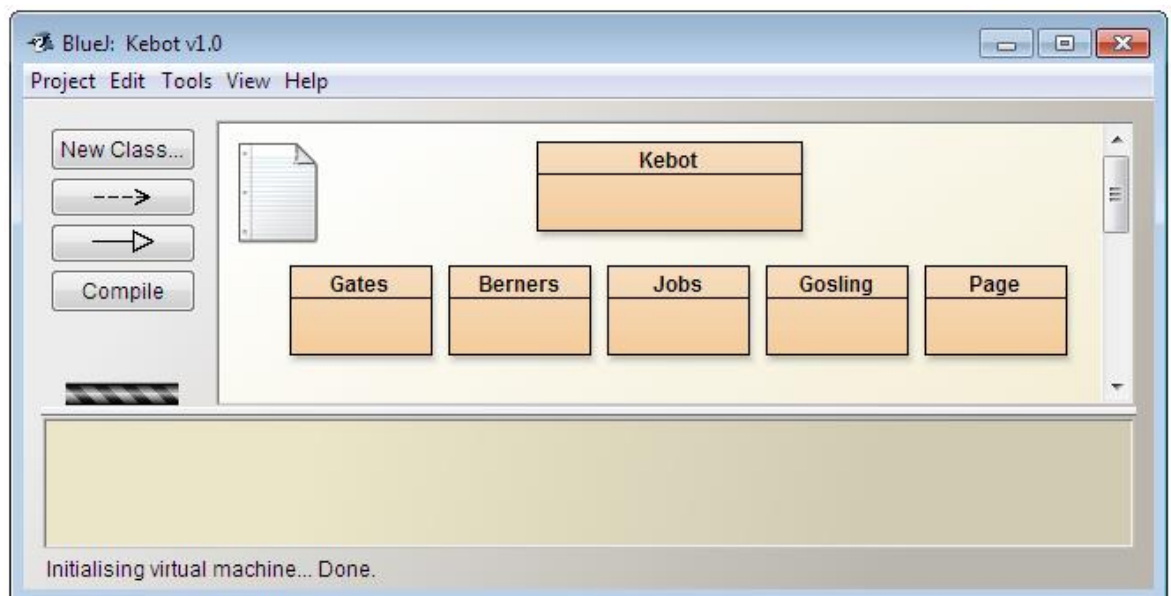


Figure 4-1. The Kebot Class Diagram when loaded in the BlueJ IDE.

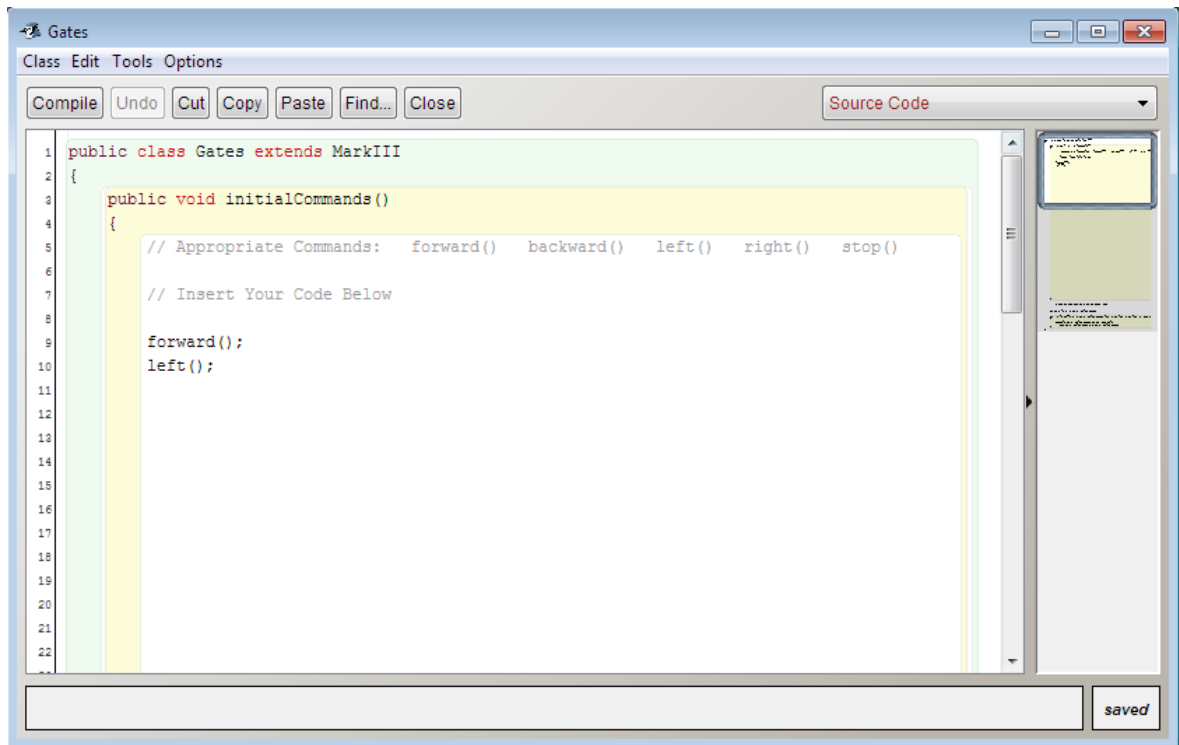


Figure 4-3. A screenshot of the code editor.

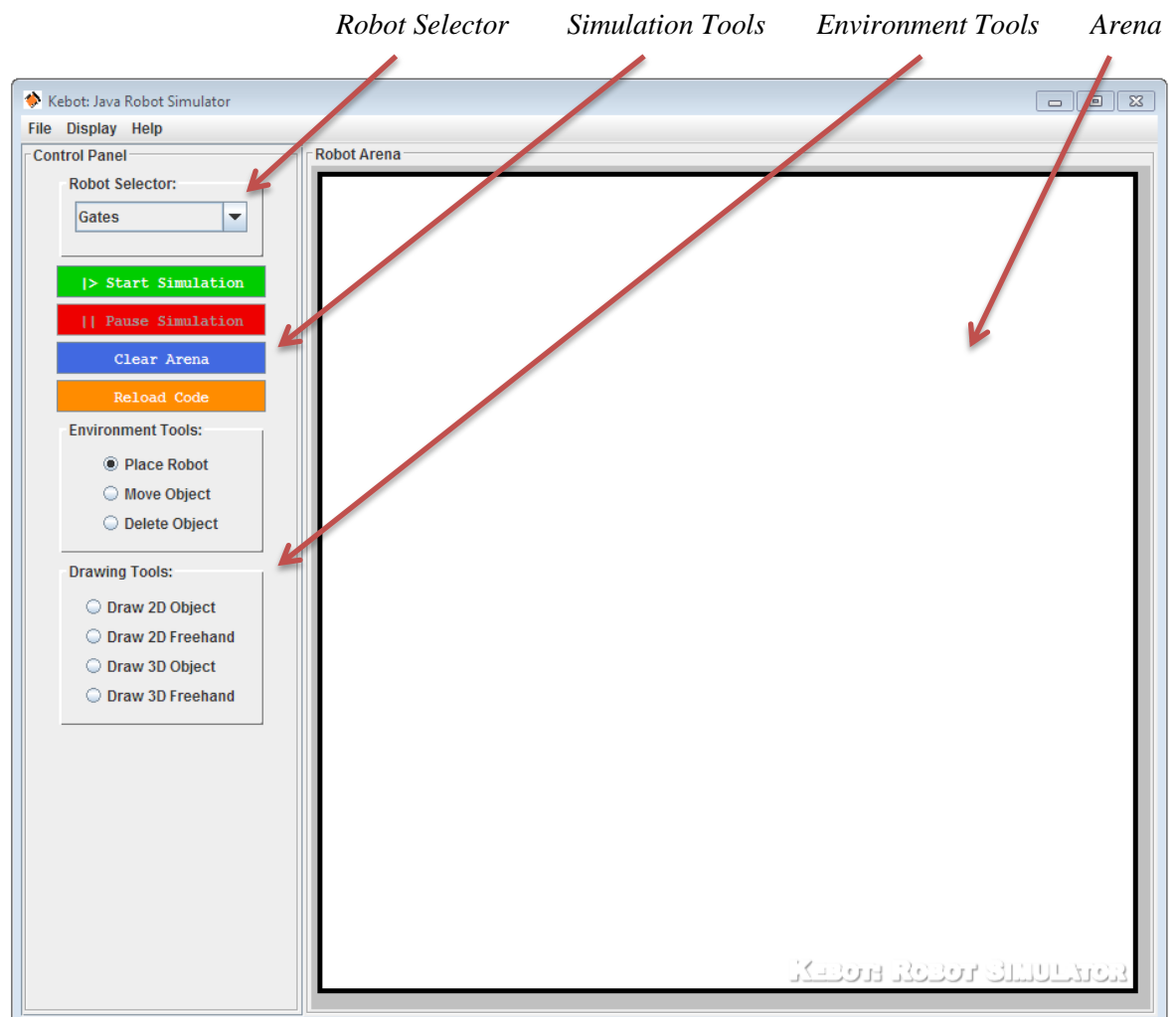


Figure 4-2. Annotated screenshot of the Kebot robot simulator.

Features of the Kebot GUI are highlighted in Figure 4-2. When Kebot is loaded a blank arena, into which one of the five robotic agents can be placed, is presented. This arena is viewed from a top-down, 2D, perspective. The robotic simulation is controlled by selecting the Start and Pause buttons. Various Java programming methods are used to instruct the Kebot robotic agents depending on the task being completed (details of which are provided in the following chapter). The simulated robots have a full range of 360 degree movement although there are restrictions on the maximum translational and rotational speeds achievable (0.3 metres per second and 180 degrees per second respectively). These speed limitations ensure that simulated agents always exhibit behaviour that is comparable to a real-world robot. Both 2D and 3D objects can be drawn by the user in the arena. Arena backgrounds can be saved and loaded and this has allowed for the creation of interesting and imaginative programming tasks. Important messages appear in a pop-up window. Kebot's drop-down menus only contain tools that are used during the relevant programming tasks (such as an option to display speed and distance travelled information). Information related to control methods, sensors and drawing tools is discussed in the following chapter along with further details of associated workshop content.

Examples of features new to Kebot include the ability to load a range of different arena backgrounds, the creation of an information panel that displays details of variables such as distance travelled and other new functionality that is explained further in Chapter Five. Kebot is considered to have been optimised for supporting the teaching of introductory programming concepts after the review of educational software guidelines and use of the PERS during two exploratory studies.

The Kebot simulator allows:

- Real-time interaction with a simulated robot
- Users to customise and modify the robots' environment with objects as they see fit
- Coding in 'real' Java (as opposed to visual event-driven programming environments such as Scratch or other 'lighter' educational programming languages)
- The creation of imaginative tasks due to features such as 'Load Background'

- An accurate representation of a real-world robot which users can better relate to (compared to more restricted simulated environments where the robot inhabits a grid-based world)

See Chapter Five for specific details of how Kobot is used to support the learning of introductory programming concepts in practice and for additional information about changes that were made to the PERS. Limitations of the Kobot simulator are also discussed along with limitations of the research design and other potential issues. The Kobot robot simulator, and associated material, can be freely downloaded online¹. In Appendix A14 the specifications of Kobot's control methods are provided. This includes information in regards to what functionality each method offers. In the following sub-section the programming concepts that were selected to be taught using Kobot are discussed.

¹ http://www.scm.keele.ac.uk/staff/l_major/files.php

4.4 Deciding What Concepts to Teach Using Kebot

To finalise the functionality of Kebot, and before the associated workshop could be developed, a decision needed to be made in regards to what programming concepts would be taught using the software. The ACM and IEEE Joint Task Force Computer Science Curriculum Report 2008 (ACM/IEEE, 2008) had a significant influence upon which fundamentals were selected.

4.4.1 ACM/IEEE Computer Science Curricula 2008

Since 1968 the ACM and IEEE Computer Society have provided computing curriculum guidance at approximately ten-year intervals. With the publication of the most recent curricula, in 2001, a commitment was made by the ACM and IEEE to provide advice on a more regular basis (ACM/IEEE, 2008). The next major update to the guidelines is due to be released in late-2013 (Sahami *et al.*, 2011). However, in 2008, an interim report was published with the intention of bridging the gap between volumes and to recognise the rate of change in the discipline (ACM/IEEE, 2008).

Through a process of consultation and discussion with leaders from industry and academia, the ACM and IEEE Joint Task Force attempt to help computing educators address the challenges they face (Sahami *et al.*, 2011). The guidance they give provides direction on how best to introduce computing topics. For the purposes of this research the sub-section of the ACM/IEEE interim report entitled ‘Programming Fundamentals’ (specifically Fundamental Constructs) was considered relevant. For each unit identified by the ACM and IEEE a minimum core coverage time is outlined. What topics should be taught within each unit, and several learning objectives, are detailed. The ACM and IEEE are internationally recognised organisations and their standardised curricular guidelines transcend geographic boundaries (Douglas *et al.*, 2010). By basing this research on guidance provided by the Joint Task Force it ensures that the findings generated are capable of having a global impact.

To teach the Fundamental Constructs identified by the ACM/IEEE a core coverage time (defined as the minimum class time required) of nine hours is recommended. In Table 4-2 the Fundamental Constructs identified by the ACM/IEEE can be seen. Table 4-3 details learning objectives related to the successful learning of these concepts. Bloom's Taxonomy (Bloom *et al.*, 1956; Anderson *et al.*, 2005) has influenced these learning objectives (Gluga *et al.*, 2012a). The learning objectives defined in the ACM/IEEE curricular show a spread of competence levels ranging from Bloom Knowledge (e.g. *describe*) to Bloom Synthesis and Evaluation (e.g. *implement*) (Gluga *et al.*, 2012b). Krathwohl defines a scheme for classifying educational objectives and states how factors, related to a learner's behaviour, can be used to consider how to improve the planning and delivery of educational material (Krathwohl, 2002). Learning objectives are useful for assessing whether the intended aims of a course have been met (Bonner, 1999).

<i>Fundamental Programming Constructs</i>
Basic syntax and semantics of a higher-level language
Variables, types, expressions and assignment
Simple Input/output
Conditional and iterative control structures
Functions (methods) and parameter passing
Structured decomposition
<i>Minimum core coverage time: 9 hours</i>

Table 4-2. Fundamental programming constructs identified by the ACM/IEEE.

<i>Fundamental Programming Constructs – Learning Objectives</i>
1. Analyse and explain the behaviour of simple programs involving the fundamental programming constructs covered
2. Modify and expand short programs that use standard conditional and iterative control structures and functions (methods)
3. Design, implement, test and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional/iterative structures and the definition of functions (methods)
4. Choose appropriate conditional and iteration constructs for a given programming task
5. Apply the techniques of structured decomposition to break a program into smaller pieces
6. Describe the mechanics of parameter passing

Table 4-3. Learning objectives for the fundamental programming constructs identified by the ACM/IEEE.

Use of the ACM/IEEE guidelines was considered appropriate because:

They fitted the optimal workshop format – During discussions with educators (when conducting the research described in Chapter Three and when preparing for that reported in Chapters Six and Seven) it was recognised, for the workshop to be able to involve a range of participants of different ages and at different stages of education, that the optimum workshop duration was no more than two days (around six hours per day and 12 hours in total). Due to computing educational reforms, that were on-going in the UK whilst undertaking this work, it was considered pragmatic to structure the associated workshop so that the involvement of high school participants (and recent high school leavers) could be accommodated. Previous work demonstrates how it is possible to run introductory programming sessions, with the support of robotic tools, in a timeframe similar to that suggested by the ACM/IEEE. Wu (Wu *et al.*, 2008) reports on workshops over several weeks during which students met two hours a week. Becker (Becker, 2001), meanwhile, states how basic programming concepts were introduced to students over a five week period.

They increased the chance of recruiting participants – When preparing for the research it was recognised how restrictions on the amount of time participants would have available would need to be taken into account. Balance needed to be achieved between the requirement to perform a substantial study and making the prospect of taking part attractive enough. There were limits on how long the associated workshop could last, if high school or Further Education (FE) learners and staff were to be involved, as regulations such as the National Curriculum have to be considered. Whilst educational providers are afforded some lenience in what they choose to teach, significant divergence from pre-planned syllabuses is not permitted. As it was also considered that those involved may be doing so at the expense of other educational commitments (as participants may have been required to miss classes in other subjects for the duration of the workshop) the educational welfare and other obligations of potential participants would have to be respected if the research was to remain ethical. The success of the exploratory study workshop format, coupled with negotiations that took place with potential participants during the planning stages,

demonstrated how a workshop lasting for a similar duration as suggested by the ACM/IEEE would be the most appropriate.

They are suitable for supporting the intentions of this research – As outlined in Chapter Five, the case study methodology was selected to support this research. Underpinned by the findings of the SLR this is an exploratory study which aims to seek an insight, and to generate new ideas, through the undertaking of novel and innovative research. As case studies involve using multiple sources of evidence to draw valid scientific conclusions, there is no set limit for how long an investigation should last. Instead, a decision is made by the researcher(s) when they feel data saturation will be reached and the objectives of the study fulfilled. This work was not intended to investigate the use of a robot simulator during a full programming course (e.g. one that takes place over a whole semester). It was only intended for Kebot to be used to teach introductory programming concepts. The fundamental programming constructs identified by the ACM/IEEE were, therefore, considered sufficient for the purposes of supporting this work.

Consultation of other guidance suggests that they are suitable and relevant – As presented in Section 4.4.2, an examination of other sources of evidence indicated how the ACM/IEEE guidelines are appropriate for the purposes of this research.

Updated ACM/IEEE computing curriculum guidelines are due to be released in late-2013 (Sahami *et al.*, 2011). These revised guidelines came too late for use during this research. A draft version of the updated curriculum, entitled ‘Ironman’, has been released by the ACM/IEEE Joint Task Force². Whilst the content of this draft is subject to change, it appears the Programming Fundamentals section of the Interim Report will be incorporated into a new section named ‘Software Development Fundamentals’.

² Available online at: <http://ai.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf> (Last accessed 9th October 2013)

4.4.2 Consideration of other curriculum guidance

Other sources of guidance were consulted before finalising the decision to use the ACM/IEEE guidelines. The British Computer Society (BCS) is the association responsible for maintaining the standards of British IT and computing educational qualifications (BCS, 2012) through a process of Higher Education (HE) course accreditation (Sandy *et al.*, 2007). It was found, after examining literature on course accreditation³, that the BCS offer no specific advice related to the teaching of programming. Indeed, the term ‘programming’ is not used in the BCS accreditation documentation. The Quality Assurance Agency (QAA) is an organisation that safeguards standards in the UK’s HE sector. The QAA provide subject benchmark statements that provide a means for the academic community to describe the nature and characteristics of courses in a given subject area. These statements detail general expectations about standards for the award of qualifications. The latest set of computing benchmarking standards was released in 2007⁴. The QAA guidance was, however, not considered substantial enough to underpin this research. This is because only an abstract overview of broad programming topics, and a list of generic learning objectives, is provided.

Another source of information deemed as potentially appropriate was curricular advice issued to all British high schools and FE institutions. Such guidance sets out requirements that must be met in order to ensure a high quality, and standardised, level of education for persons aged up to 18 years old. Whilst some curriculum guidance relating to computing was found to be available (supplied by exam boards such as OCR⁵), a decision was taken not to use this. This was because of the likelihood that such guidance would be subject to change due to the launch of a new curriculum in 2014 (Copping, 2012), and given planned changes to how computing will be taught in British high schools from this point (Wells, 2012). It was considered too great a risk to base this research on material that was potentially subject to modification. Also, by not selecting guidance intended for

³ Available online at: http://www.bcs.org/upload/pdf/hea-guidelinesfull-2012_1.pdf (Last accessed 9th October 2013)

⁴ Available online at: <http://www.qaa.ac.uk/Publications/InformationAndGuidance/Documents/computing07.pdf> (Last accessed 9th October 2013)

⁵ <http://www.ocr.org.uk>

use in one country only, it ensures that the results of the research can have an international impact. Linked to the teaching of computing in UK schools is the Computing at School (CAS) group. CAS is a de facto community of practice formed in 2009 (Bradshaw & Woollard, 2012) that has brought together educators, with representatives from industry, to support the teaching of computing. The CAS group has released literature with the aim of influencing policy related to the teaching of computing⁶. Whilst most advice provided by CAS was not disseminated until the later stages of this project, for the reasons discussed in Section 4.4.1 and the preceding paragraph, the ACM/IEEE guidance is considered better suited for the support of this work.

When undertaking the supplementary search of the literature, current research that is taking place at Carnegie Mellon University was identified (Flot *et al.*, 2012). The authors of this research (who cite the SLR completed as part of this thesis) outline a number of concepts, the teaching of which, they believe should be supported by the ideal robot simulation environment. Whilst not published until the latter stages of the research project, the concepts identified by this work can be used as a retrospective checklist to demonstrate the validity of selecting the ACM/IEEE programming fundamentals. As evidenced in Table 4-4, there are similarities between the concepts outlined by the ACM/IEEE and those identified by Flot (Flot *et al.*, 2012).

Fundamental programming concepts identified by the ACM/IEEE Joint Task Force		Concepts that the ideal robot simulator environment should teach*⁷
Basic syntax and semantics of a higher-level language	↔	Programming syntax. Algorithmic thinking
Variables, types, expressions and assignment	↔	Statements and structures. Variables
Simple Input/output	↔	Using sensor feedback
Conditional and iterative control structures	↔	Conditional operators
Functions and parameter passing	↔	Functions. Passing parameters
Structured Decomposition	↔	Debugging programs

Table 4-4. Outlining the similarities between the fundamental programming concepts identified by the ACM/IEEE and Flot *et al.*

⁶ Available online at: <http://www.computingatschool.org.uk> (Last accessed 9th October 2013)

⁷ According to Flot *et al.*, 2012. Note, three topics considered beyond the scope of this study (including robot mathematics and controlling motors and servos), which are thought to specifically related to the study of robotics, have been omitted from this list.

4.4.3 How did selected programming concepts influence the design of Kebot?

The fact that a set of pre-defined programming principles were identified prior to the development of Kebot, and a minimum recommended coverage time for each concept was provided, influenced what features were incorporated into the final version of the software. This is because, after the consultation of relevant literature (details of which are presented in Chapter Five), workshop tasks were planned prior to modifying the software. Due to this, time was expended on the creation of only necessary software features that would be utilised during the workshops. The identification of programming concepts that would be taught prior to modifying the PERS also influenced the decision of what features to remove. The set of learning objectives outlined by the ACM/IEEE likewise influenced Kebot's design. This is because the objectives guided the substance and detail of software features (and programming tasks). Details of the programming tasks completed using Kebot are given in the next chapter.

4.5 Summary

In this chapter details of a robot simulator, that has been developed to achieve the research objectives of this thesis, have been presented. Influenced by the findings of the Systematic Literature Review and exploratory studies, the features of the pre-existing robot simulator were expanded. This modified simulator, named Kebot, allowed for an investigation into the use of a robot simulator as a programming teaching tool. The modifications that were completed have been outlined and information about the additional functionality of Kebot given. The programming concepts that were selected to be taught using Kebot have also been provided. Reasons why such concepts were chosen have been justified. As is outlined in the following chapter, the Kebot robot simulator has been used during an empirical case study. This case study allowed for an investigation into the effectiveness of using simulated robots as programming teaching tools.

Chapter Five

Case Study Methodology

In this chapter a study design, devised to achieve the research objectives of this thesis, is presented. Using the case study methodology empirical research has been undertaken. Case studies are highly flexible and well suited to supporting a variety of research projects. Rationale for, and discussion on the appropriateness of, the research design is given. An introduction to the sources of data used, and the participants involved, is presented. The Kobot robot simulator was used and this allowed for an investigation into the effectiveness of simulated robots as programming teaching tools. An associated 10-hour workshop was created to support the teaching of introductory programming concepts. An overview of the structure and content of this workshop are provided and limitations of the study design are discussed.

5.1 An Introduction to the Case Study Methodology

Case studies are strategies for research that involve an empirical investigation using multiple sources of evidence (Robson, 2011). The methodology has been used to support the conduct of research in a variety of fields including education (Merriam, 1998), health science (Baxter & Jack, 2008), robotics (Burdea *et al.*, 2012), software engineering (Verner *et al.*, 2009), initial teacher training (Bradshaw *et al.*, 2011), the teaching of programming (Jones, 2010) and others (Yin, 2009). The results of case study often lead to new insights, the building of theory and have high validity with practitioners – the ultimate users of research (Dul & Hak, 2012). Indeed, the methodology is able to provide a deeper understanding than controlled experiments (Runeson *et al.*, 2012) whilst remaining capable of achieving scientific objectives (Lee, 1989).

Case studies are well suited for research where the boundary between phenomenon and context is not clearly evident (Yin, 2009). This is because case studies investigate real-world occurrences in their real-world settings (Robson, 2011). To ensure the validity of a case study, a data triangulation strategy must be adopted. Triangulation involves using different sources of information to take multiple measures of an event (Gibbert & Ruigrok, 2010). The findings generated by each source can then be compared and contrasted with one another. Advantages of triangulation include, “increasing confidence in research data, creating innovative ways of understanding a phenomenon, revealing unique findings, challenging or integrating theories and providing a clearer understanding of the problem” (Thurmond, 2001). Triangulation can, however, be time consuming (Gibbert & Ruigrok, 2010).

5.2 Case Study Design

5.2.1 Aim

This case study was designed to investigate the effectiveness of simulated robots as introductory programming teaching tools. As discussed later in this chapter, the Kobot robot simulator has been used in a 10-hour workshop (described in Section 5.4) to achieve this aim. Experience gained earlier in the research project (during two exploratory studies), in addition to the findings reported by existing literature, were the main factors that shaped the case study design and the manner in which it was implemented. The study presented was exploratory as it aimed to seek new insights and to generate new ideas (Runeson *et al.*, 2012). It is also considered to be positivist as past evidence has been examined, a range of variables have been measured, propositions have been tested and inferences can be drawn from the samples involved to stated populations (Klein & Myers, 1999). The creation of a protocol ensures reliable, transparent and targeted work that considered potential problems in advance (Yin, 2009) and helps to guarantee that a rigorous methodological path has been followed. A protocol, based on one described by Brereton (Brereton *et al.*, 2008), was developed. Details of the protocol were disseminated at the Psychology of Programming Interest Group Annual Conference (Major *et al.*, 2012b). The case study asks the same research question identified in Chapter One:

Is a robot simulator an effective tool for supporting the learning of introductory programming?

This study is a multiple-case case study. Figure 5-1 shows the study design. *Case One* (discussed in Chapter Six) involved trainee ICT/CS high school teachers, most of whom had some programming experience. The experiences of 22 trainees are considered as part of this case. *Case Two* (discussed in Chapter Seven) involved students, aged 16 to 18 years old, enrolled on a FE course at the time. The experiences of 21 students are considered as part of this case. Data collected from *interviews* with three in-service teachers (each of whom was associated with one of the student cohorts), and *observations* by the workshop leader, have also been used as additional data sources.

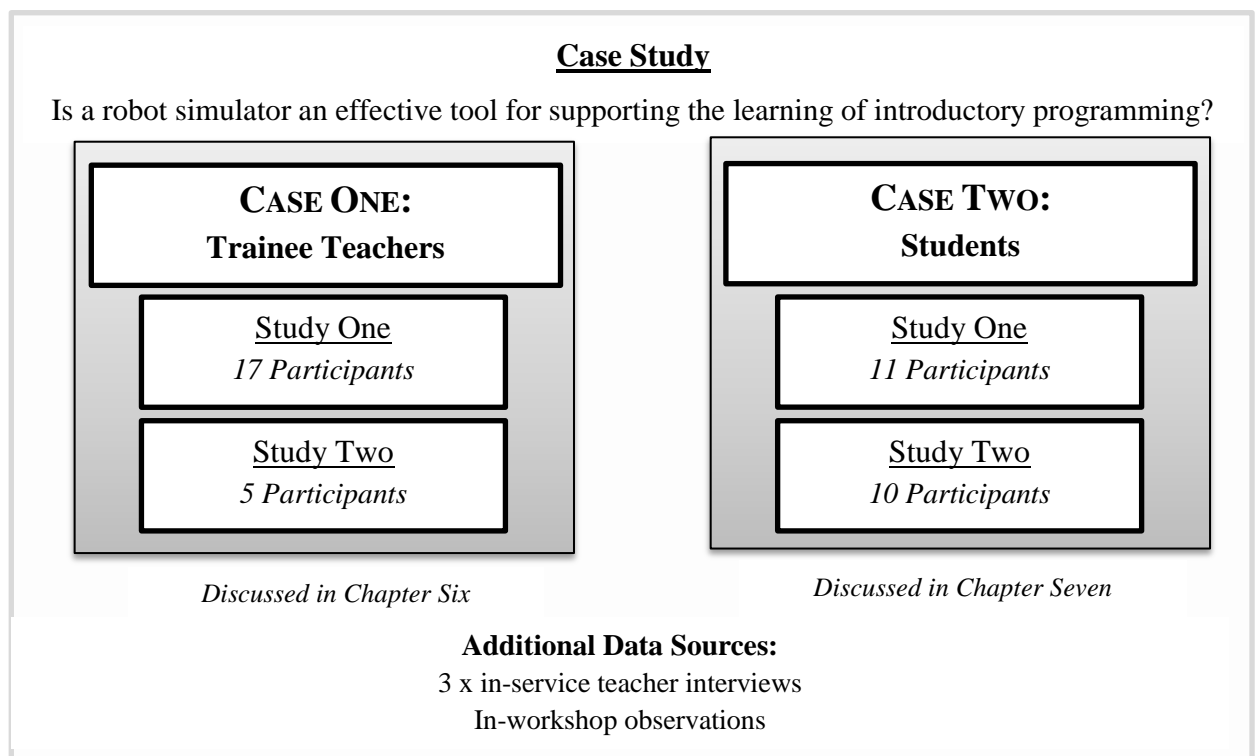


Figure 5-1. The multiple-case case study design.

5.2.2 Propositions

Propositions help to maintain the focus of a case study, increase the likelihood of its successful completion in addition to helping guide data collection and analysis (Baxter & Jack, 2008). Propositions also influence where to look for relevant evidence (Yin, 2009). Four propositions were formulated after considering the findings of the SLR, the objectives of this research, the backgrounds of potential participants and the experience gained during the exploratory studies. These propositions are considered in Chapter Eight (Discussion) and were:

P1 A robot simulator is an effective tool for supporting the learning of introductory programming

P2 A robot simulator improves novices' perceptions of programming

P3 A robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience

P4 A robot simulator improves trainee ICT/Computer Science (CS) teachers' confidence in their ability to teach introductory programming

5.2.3 Data Sources

Several data sources have been used during the case study. The research question and propositions are addressed as follows:

- By using **questionnaires** to determine participants views on Kebot, and programming, before and after the workshop
- By maintaining a **log of events** that occurred when running each of the workshop sessions
- By administering (and later scoring) **programming tests**, which have been constructively aligned with several learning objectives, to determine programming progress of student participants
- By **interviewing** three current teachers to determine their views on the effectiveness of the simulator

Further details of the data collection instruments used, and the data analysis strategy chosen, are provided in Chapters Six and Seven.

5.2.4 The suitability and relevance of the case study methodology for this research

A case study was chosen to support this research because:

- The methodology has previously been used to explore topics including education, robotics, the teaching of programming and software engineering. The work that is presented has been influenced, in some way, by each of these fields. As a result, case study was identified as being suitable for use during this research also.
- Whilst this work is not truly cross-disciplinary (as it is grounded in the Computer Science field), the flexibility of the methodology would enable any cross-disciplinary (or otherwise

unexpected) themes to be addressed. This is because case studies are well suited for investigations where the boundaries are not clear (Yin, 2009).

- The methodology is suited to supporting exploratory work into new areas (Runeson & Höst, 2009). As established in Chapter Two the fact that there is limited related research, coupled with the novel nature of the intervention (as the robot simulator and the workshop were both developed to address the objectives of this project), demonstrates how this study is exploratory. The findings of case studies are also highly valued by practitioners (Dul & Hak, 2012) and, therefore, the potential impact of this research upon current practice was likely to be enhanced by the use of the methodology.
- They allow for a mixed method research strategy as the exactness of quantitative, and ‘richness’ of qualitative, approaches can be combined (Runeson *et al.*, 2012). Combinations of qualitative and quantitative data can help to enhance understanding (Seaman, 1999).
- Compared to alternative research approaches (such as controlled experiments) the methodology is able to provide a greater insight whilst ensuring scientific rigour. This is because case studies allow a fuller understanding of how, or why, an intervention worked (Paculla *et al.*, 2011). Indeed, case study is a valuable research method that facilitates contributions to scientific knowledge (Flyvbjerg, 2006).

5.2.5 Consideration of other potentially relevant research methods

Before selecting the case study methodology to support this work, the use of a survey, action research and experimental strategies were considered. These methods have their own advantages but, as discussed below, were deemed less suitable. Other study types have also been used in the conduct of empirical research including experience reports, meta-analysis, example application and discussion (Zannier *et al.*, 2006). While all of these study types were considered incapable of supporting this project independently, as documented later, some of these principles have been

incorporated into the case study design. This was possible due to the flexibility of case studies (Robson, 2011).

The *survey* approach refers to a group of methods with an emphasis on quantitative analysis. Traditionally a large number of standardised data sets are collected through techniques such as questionnaires or telephone interviews before being examined (Gable, 1994). Surveys provide only an overview of a studied field (Runeson *et al.*, 2012) yielding little information on the underlying meaning of the data (Gable, 1994). Use of a survey alone was considered inadequate for achieving the objective of the research, specifically an investigation into the effectiveness of a robot simulator as a tool to support the learning of introductory programming. To accomplish the research aims, an in-depth investigation was required and not just a summary of the research area.

Action research has the purpose of influencing or changing some aspect of whatever is the focus of the research (Robson, 2011). The technique shares similarities with the case study methodology. While case study is purely observational, however, action research is focused on a change process (Runeson *et al.*, 2012). Action research is an iterative procedure which involves researchers and practitioners acting together on a cycle of tasks including problem diagnosis, action intervention and reflective learning (Avison *et al.*, 1999). Researchers attempt to solve a real-world problem while simultaneously studying the experience of solving the problem (Davidson *et al.*, 2004). The intention of this project was to explore a particular research area. While the findings of the work may have an impact upon future policy and practice, it was never intended for the work to improve the process of teaching programming during the project's execution. Instead this is an area more suited for future research, based on the results of the case study.

Experiment involves measurement of the effects of manipulating one variable on another (Robson, 2011). The classical method for identifying cause-effect relationships is to conduct controlled experiments (Sjoberg *et al.*, 2005). Quasi-experiments are related to controlled experiments and involve subjects not being randomly assigned to treatments. They are normally used when (typically for ethical reasons) subjects must be allowed to choose their treatment or when the

treatment cannot be allocated at random (Easterbrook *et al.*, 2008). Experiments are performed when an investigator can manipulate behaviour directly, precisely and systematically (Yin, 2009). This notion of control is important and variables other than the chosen independent variables must not be allowed to affect the conduct of an experiment (Easterbrook *et al.*, 2008). If critical variables are ignored or constrained then the experimental results might not generalise to real-world settings (Easterbrook *et al.*, 2008). If sufficient control is not possible then an experiment cannot be performed and a case study is the preferred technique (Pfleeger, 1994). The repeatability of the situation being investigated must also be considered. An experiment should not be performed if replication is not possible (Pfleeger, 1994). It should also be considered that bias can enter into the conduct of experiments, as with any form of research, and that they are limited in their ability to ask “how” or “why” questions (Yin, 2009). As described below, Sjøberg (Sjøberg *et al.*, 2005) identifies a number of threats that can negatively impact upon the findings of an experiment:

- *Ambiguous Temporal Precedence* – Lack of clarity about which variable occurred first may lead to confusion about which variable was the cause and which was the effect
- *Selection* – Differences over conditions in respondent characteristics that could also cause the observed effect
- *History* – Events occurring concurrently with the treatment could cause the observed effect
- *Maturation* – Naturally occurring changes over time could be confused with a treatment effect
- *Testing* – Exposure to a test can affect scores on subsequent tests, an occurrence that can be confused with a treatment effect
- *Instrumentation* – The nature of a measure may change over time and could be confused with a treatment effect

All of these factors can critically bias the results of an experiment. Similarly, they can also adversely influence the findings of a case study. However, unlike experimental research, the case study methodology allows for an active investigation into such matters as part of the study design (Yin, 1999). This is called searching for rival explanations and involves systematically looking for

alternative themes, divergent patterns and other explanations that may account for the results of the research (Patton, 2002). Details of the rival explanations analytical strategy, adopted during this study, are discussed in Chapter Eight.

The fact that experimental research requires decisions in advance, in regards to which variables to ignore, is one reason why an experimental strategy was not used. It was considered that important variables may be disregarded during an experiment that could lead to valuable findings being overlooked (Easterbrook *et al.*, 2008). By conducting a case study the risk of important findings been missed is minimised as multiple sources of information (and methods) are used to gather a rich set of data. Another reason why an experimental research strategy was not selected is because it was considered that it would be too problematic to exert the required control over all possible variables. If control of all variables cannot be guaranteed then an experimental research strategy should not be chosen (Pfleeger, 1994). For this work, control over the research setting, participant involvement and test instrumentation was identified as being potentially challenging.

5.2.6 The Ethical Approval Process

To ensure that the research remained ethical, an application was submitted to the Keele University Ethical Review Panel (KUERP). It is KUERP's duty to assess whether a proposed application of research methods is acceptable ethically. A number of steps were taken to ensure ethical clearance was secured. Full ethical approval was first granted in November 2010 and was later updated in April 2012. This demonstrates how the research design is ethically sound. The relevant Approval Confirmation Letter can be found in Appendix A4.

5.3 Verification of the Case Study Approach

As they differ from controlled empirical studies, methodological practices must be applied when performing a case study to ensure the value and generalisability of results (Runeson *et al.*, 2012). A comprehensive literature review, during the early stages of a project, is one activity that offers a solid foundation for a subsequent case study (Verner *et al.*, 2009). It is believed that the Literature Review presented in Chapter Two satisfies this criterion. Triangulation (of both data sources and methods) is one means of obtaining valuable results. The need for triangulation is clear when relying primarily on data that is qualitative, but it can also help to compensate for measurement or modelling errors when performing quantitative research (Runeson *et al.*, 2012). As discussed in the following chapters, triangulation has been used to ensure rigorous findings. Demonstration of a chain of evidence in the final write-up can also ensure rigorous case study research (Verner *et al.*, 2009) and this was taken into account when presenting evidence, discussing findings and drawing conclusions.

Advice provided by Yin (2003) and Runeson and Höst (2009) was considered when developing the case study protocol and reporting the findings of the research. Yin outlines the following factors as being typical of exemplary case studies (Yin, 2003): the study is of a significant topic; the study must be complete in that the boundary of the case is made explicit, there is a comprehensive collection of appropriate evidence, there are no significant constraints on the conduct of the study; the study must consider alternative perspectives on the topic; the study must present sufficient evidence when reporting the results and disseminating the artefacts of the study; the reports of the study must be well written; the case study must respect ethical, professional and legal standards. Guidelines have also been produced that help to ensure high quality case study research. In Appendix A15 details of how a number of case study design criteria have been addressed is presented (Runeson & Höst, 2009).

A strategy for ensuring reliable case study findings, examining rival explanations, has also been adopted and embedded in the data collection and data analysis stages. Reporting that a case study

sought out, considered and did not find evidence to support a number of plausible rival explanations which may account for the research findings enhances the credibility of a case study and helps to counter the suggestion that the results are shaped by any predispositions or biases. Yin (Yin, 2009) lists many types of potential rivals while Rosnow and Rosenthal (Rosnow & Rosenthal, 1997) discuss factors that can impact upon the results of research involving human subjects. Relevant rival explanations are discussed in Chapter Eight.

The creation of a protocol ensures reliable, transparent and targeted case study research that considers potential problems in advance (Yin, 2009). A protocol, based on one described by Brereton (Brereton *et al.*, 2008), was developed as part of this project. The protocol went through several iterations and was reviewed by two PhD supervisors (PB and TK) in addition to an independent expert (Barbara Kitchenham of Keele University). The protocol also underwent peer review and was disseminated at the Psychology of Programming Interest Group Annual Conference (Major *et al.*, 2012b). Whilst the logic of replication differs for case studies compared to formal experiments (Runeson *et al.*, 2012), the protocol could also be used as a basis for future studies.

5.4 Workshop Content

In this section details of how the concepts identified by the ACM/IEEE (outlined in Chapter Four) were introduced, to evaluate the effectiveness of the Kobot simulator as a tool to support the learning of introductory programming, are provided. The full set of slides and other materials used during the workshop are available online¹. Some fundamentals were not introduced in the order listed by the ACM/IEEE and were taught over the entire workshop (e.g. structured decomposition). Previous research (details of which is outlined in the sub-sections that follow) has influenced how concepts were introduced. Experience gained during the exploratory studies has also influenced the workshop structure. The workshop content was reviewed by a PhD supervisor (TK). Three novice programmers were involved in a trial of the workshop in order to ensure that there were no issues with the structure or content. Due to other time commitments, these participants were only available to attend the trial workshop for one working day (around eight hours). As a result, a decision was made to only deliver the workshop content and to not test the instrumentation intended for data collection purposes (e.g. programming exercises). Whilst the decision to not trial these instruments may be potentially responsible for some later issues, as discussed in Chapter Eight, such factors did not significantly impact the findings generated. The author was the workshop leader for all workshops. Tasks completed by participants were pre-planned although experimentation, questions and feedback were actively encouraged. The workshop involved:

- The *presentation, demonstration* and *discussion* of programming concepts
- A *task* phase where programming challenges were attempted using these concepts
- Opportunity to *reflect* and *reinforce* knowledge by asking questions and viewing model code

Parallels can be drawn between this three-stage process and one outlined by Griffiths and Blat (Griffiths & Blat, 2004): initial fascination (with the intervention), problem solving (using the intervention) and reflection (following use of the intervention). Table 5-1 displays the workshop format with details of when data collection activities were undertaken provided in bold.

¹ http://www.scm.keele.ac.uk/staff/1_major/files.php

	Main Programming Activity	Relevant Section in Thesis	Time Required
<i>Day One</i>	Basic syntax and semantics of a higher-level language	5.4.2	1 hour
	Variables and Constants	5.4.3	30 minutes
	Logical Expressions	5.4.4	1 hour
	Counting, JOptionPane and Nested Statements	5.4.5	45 minutes
	In-Workshop Programming Exercise One		15 minutes
	Introducing the Remaining Data Types	5.4.6	15 minutes
	While and Do While Loops	5.4.7	45 minutes
	Method Creation	5.4.8	30 minutes
<i>Day Two</i>	Conditional and Iterative Control Structures	5.4.9	2 hours 15 mins
	In-Workshop Programming Exercise Two		15 minutes
	For Loops	5.4.10	45 minutes
	Arrays	5.4.11	30 minutes
	Post-Workshop Programming Exercise		45 minutes

Table 5-1. The workshop format.

Day One was designed to introduce fundamental programming concepts and KeBot. During Day One full control over robots' movement was not exerted in the early stages. It was intended that Day Two of the workshop would be more flexible with participants creating varied solutions to problems that were set. This is because those involved would assume greater control over the robotic agents and would make greater use of sensors. Programming constructs introduced during Day One would continue to be used, although the nature and difficulty of the challenges was increased for the second part of the workshop. To achieve the objectives of the research, Day Two was also designed to enable a greater evaluation of the robot simulator. As a result, more data collection activities took place on the second day.

After a discussion of relevant pedagogy the programming concepts outlined in Table 5-1 are introduced as follows: aim of introducing the fundamental; consideration of relevant literature and/or experience gained during the exploratory studies; programming activities planned as a result. Note that only substantial tasks are discussed at length, often with the help of figures and code examples for illustration purposes. Information provided is only intended to offer an overview of workshop content. As stated in the introduction to this sub-section, all materials used during the workshop can be viewed online.

5.4.1 The Influence of Pedagogy on the Workshop Design

When developing the workshop the constructivism theory was considered. Constructivism proposes that humans generate knowledge from the interactions between their experiences and their ideas and that knowledge is constructed rather than discovered (Papert, 1980; Piaget, 1967). As the constructivist approach suggests that an individual's learning improves when they are involved in building something and is focused on "learning-by-doing", it was considered to be applicable to the research presented. This is because constructivism and learning with robots are linked (Alimisis *et al.*, 2007), because programming can be viewed as a constructivist activity (Wulf, 2005) and as the theory is applicable to CS-education in general (Ben-Ari, 2001).

Constructive Alignment is a variant of constructivism and combines an understanding of the nature of learning to an aligned design for outcomes-based education (Biggs, 2003). Constructive Alignment involves all components of a teaching system – including the curriculum, teaching methods and assessment tasks – being aligned. The use of a Constructive Alignment strategy has been implemented during an introductory programming course (Thota & Whitfield, 2010) and the theory was considered when designing the workshop. Indeed, the assessments administered during the workshops were devised to match the objectives identified by the ACM/IEEE.

Enquiry-based learning (and related theories such as Problem-based learning) involves individuals assuming control by taking an active role in the acquisition of knowledge (Drummond, 2009). The workshop presented allows for enquiry-based learning as programming knowledge was built from the experience using Kibot. Scaffolding is the process by which an "expert" helps a learner to complete a complex task that they could not ordinarily without support and guidance (Reiser, 2004). Once learners demonstrate a sufficient level of knowledge, the implemented scaffolding can be gradually reduced (Lajoie, 2005). During this project the workshop leader acted as the "expert" and participants' knowledge of introductory programming concepts was gradually built up (particularly during Day One). Once it was believed participants would be equipped with the

knowledge to succeed, the emphasis of the workshop shifted from being one of instruction to one based on experimentation as tasks became less restrictive.

Two other factors needed to be considered when structuring the workshop. Threshold Concepts are akin to a portal that can open up a new and previously inaccessible way of thinking about something (Meyer & Land, 2006). They represent a transformed way of understanding without which the learner cannot progress. In practice, learners can find themselves in a state of ‘liminality’ whereby they cannot go backwards (or unlearn) but at the same time are unable to progress without understanding new subject knowledge (Meyer & Land, 2006). Similarly, Cognitive Load Theory provides a basis for understanding the learning process as it uses an information processing model to describe how the mind acquires and stores knowledge in addition to providing an explanation for the limitations imposed by working memory (Shaffer *et al.*, 2003). When teaching learners an unfamiliar subject, more time has to be expended for solving problems as information stored in long-term memory is not available (Youssoof *et al.*, 2007). This can lead to an overload of working memory which must be resolved before meaningful learning can again occur (Paas *et al.*, 2004).

Due to the intensive nature of the workshop, Threshold Concepts and Cognitive Load Theory needed to be considered. The following strategy was adopted to limit the potential influence of such factors:

- When introducing a concept for the first-time, small manageable tasks were set for the participants. This allowed any potential problems with understanding to be identified early.
- Ensuring that participants were given sufficient time, opportunity and practice to develop their understanding. Referring back to concepts throughout also allowed knowledge to be reinforced.
- Ensuring that the nature of the workshop was ‘hands-on’, interactive and responsive. As participants spent most of their time during the workshop learning by doing (i.e. creating programs), rather than simply absorbing information (in the manner of a ‘traditional’ lecture), it allowed an opportunity to monitor progress. Extra assistance could then be afforded to any participant that was considered to be struggling.

5.4.2 Basic syntax and semantics of a higher-level language

Aim: To present the Kobot robot simulator, and the BlueJ Integrated Development Environment (IDE), in addition to introducing basic syntax and semantics of the Java programming language (Required Time – 1 hour).

Previous research, as documented in Table 5-2, had a significant influence on how the early stages of the workshop were planned. Details of the workshop structure are then presented.

Consideration of relevant research	Workshop actions taken as a result
Teaching programming through lectures with worked examples, followed by practical sessions, is not effective (Youssoof <i>et al.</i> , 2005)	The introduction to the workshop was brief and worked examples were used sparingly throughout
When programming, students should not start from scratch. Instead small changes to existing code should be made and learners should go through a sequence of exercises of which they can understand each step (Kolling & Rosenberg, 2001). Providing learners with incomplete programs can reduce cognitive load (Garner, 2001)	In the early stages partially complete code was provided and instructions given on how to modify this. Initial exercises also remained simple to familiarise participants with the simulator environment and style of learning
Using pseudo code can be beneficial. This is because a theoretical introduction is essential, particularly for learners with no (or little) prior CS-related experience (Grandell, 2006)	Pseudo code was used extensively, especially during the introductory part of the workshop, as it was anticipated that a large number of participants would have little programming experience
The more practical and concrete the learning situations and materials are the more learning takes place (Lahtinen <i>et al.</i> , 2005). It is important to consistently refer back to previous programming examples (Haberman & Kolikant, 2001)	The nature of the Kobot simulator ensures practical and concrete learning situations. Programming knowledge is also constantly reinforced during the workshop by referring back to previous examples and completed exercises
“Fill-in-the-blanks” is an educational pattern where new programmers are given part of a program and instructions on how to fill in missing sections (Bergin, 2000). The approach has been successful although it is best if generated code is put immediately to some use so students can see the effect of their work	Participants given code similar to previous examples (although incomplete). Instructions provided on how to ‘fill in’ well specified “blanks”. Upon successful compilation a robotic agent performs simple on-screen actions (ensuring code is put to immediate use)

Table 5-2. Details of previous research that influenced how basic Java syntax and semantics were introduced during the workshop.

Welcome to Kobot

The first part of the workshop involved: completion of consent forms and other administrative tasks; general introductions to programming and the Java language; detailing the structure and style of the workshop; providing details of the research being undertaken to comply with ethical procedures; discussion on what a robot simulator is and on how Kobot is used to support the learning of programming. Following this, a hands-on introduction to Kobot (including instructions on how to write code, compile code and run the software) was provided. The five Kobot robotic agents (named Gates, Berners, Jobs, Gosling and Page) were then introduced.

Code Modification

Following this opening, Java concepts were taught using Kobot as a teaching tool. By adhering to advice offered by previous research (evidenced in Table 5-2), it was possible to ensure that this initial exposure to programming was encouraging. An introduction to pre-defined robot movement methods, which are used during the early part of workshop, was given – `forward()`, `backward()`, `left()`, `right()`, `stop()`. Elements of the Kobot software were then gradually presented and users elicited simple behaviour from a robotic agent. The early part of the workshop involved use of the Gates robot. The Gates class, like those of the other robotic agents, comes with some prepared programming code. This code instructs Gates to move forward before turning. Participants compiled this program, placed Gates in the arena and observed the execution of this simple program. The Java syntax was then discussed and small modifications to the code were invited. This and other early tasks aimed to familiarise users with the simulator. From the outset participants were encouraged to break down their programs into smaller components so that they are easier to conceive and understand. Discussion on syntax errors, and strategies to overcome them, also took place. As participants became more familiar with the simulator, additional functionality was introduced (such as 2D and 3D drawing tools that allows interaction with the robot environment). The first substantial challenges to be completed revolved around

instructing the Gates robot to complete a square. Initially, participants expanded the code already prepared in the Gates class so that an on-screen square was drawn. Following this, the ‘Square Drawer’ arena was introduced (as seen in Figure 5-2). The objective of the Square Drawer task was to instruct the Gates robot to perform a trace that resembles a square within the bounds of another square drawn in the arena. This task allowed the mechanics of parameter passing, in addition to the role of sequence in programming, to be explained. Parameters were passed that corresponded to the number of seconds the `forward` instruction was to be completed for. The ‘Trajectories’ option within Kobot was used so the trace of where the robot had been could be seen. The Square Drawer task also introduced the ‘Load Arena’ functionality. This enabled various backgrounds (an example of which is shown in Figure 5-2) to be drawn onto the arena floor and this feature would be used extensively during the workshop. The Load Arena functionality allowed for the creation of visual and imaginative programming tasks. Sample code for this task, before and after the introduction of parameters by participants, is shown in Figure 5-3.

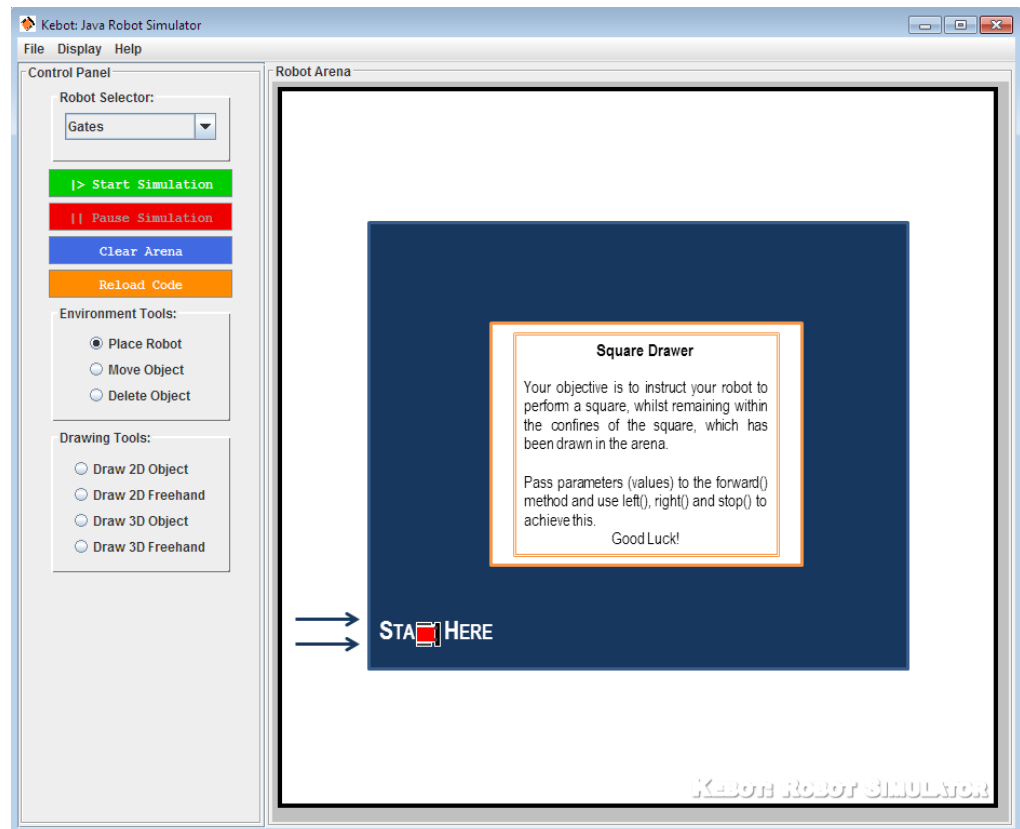


Figure 5-2. Screenshot of a robotic agent completing the 'Square Drawer' task.

<code>forward();</code>	<code>forward(12);</code>
<code>left();</code>	<code>left();</code>
<code>forward();</code>	<code>forward(10);</code>
<code>left();</code>	<code>left();</code>
<code>forward();</code>	<code>forward(12);</code>
<code>left();</code>	<code>left();</code>
<code>forward();</code>	<code>forward(10);</code>

Figure 5-3. Sample code completed during the 'Square Drawer' task before (Left) and after (Right) the introduction of parameters (representing time in seconds).

Line Tracer

After outlining ways to solve the Square Drawer task a similar, but more advanced, challenge was completed. 'Line Tracer' involved the passing of parameters to pre-programmed methods in order to instruct a robotic agent to follow (or 'trace') a pre-drawn route. Some values to be passed as parameters were provided whilst others were not. As it was envisaged that a correct solution would be more complex than other programs produced to this point, participants were encouraged to plan their code in advance. Example code from the Line Tracer task can be seen in Figure 5-4 while a screenshot of it is shown in Figure 5-5. Participants were also invited to introduce 3D objects into the arena to 'test' how accurately their prepared solution traced the line. After discussion of potential solutions an introduction to Java code conventions and good programming practices was presented before a recap of topics covered thus far. A discussion on syntax errors again took place.

```

forward(5);
left();

forward(6);
right();

forward(4);
left();

forward(5);
right();

forward(4);

stop();

```

Figure 5-4. The solution to the 'Line Tracer' task.

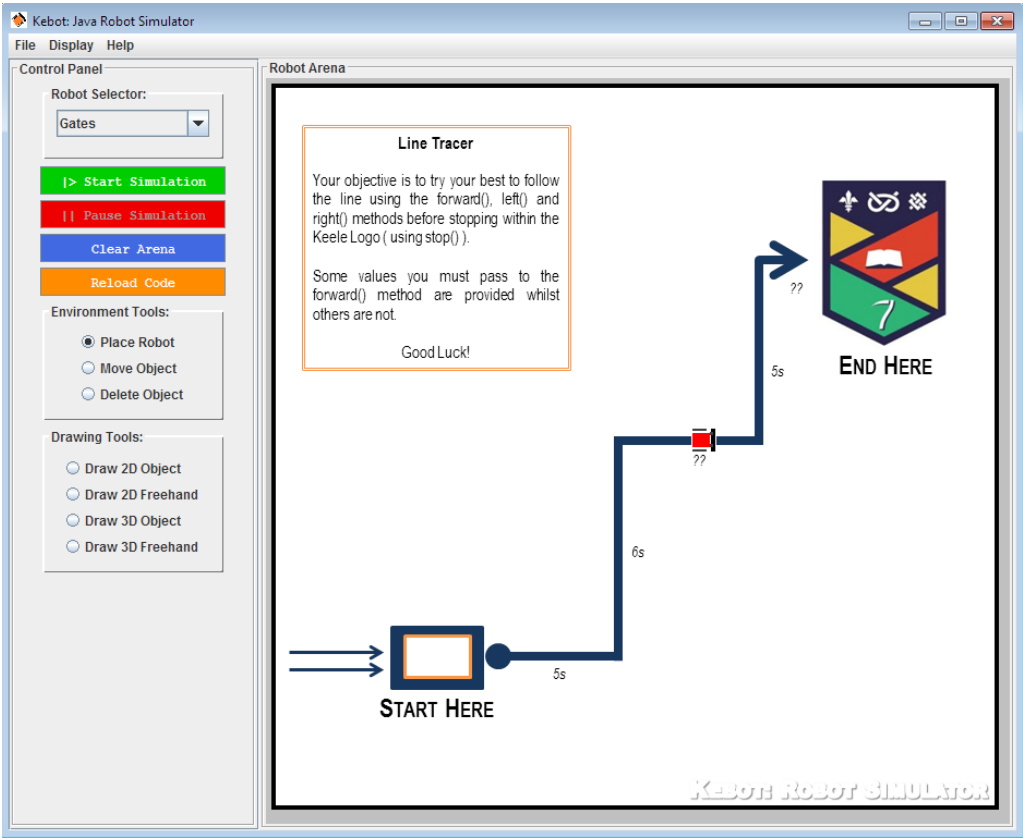


Figure 5-5. A screenshot of the 'Line Tracer' task.

5.4.3 Variables and Constants

Aim: To introduce Variables and Constants and to give a sufficient grounding so participants are successfully able to declare, and use, these during the workshop (Required Time – 30 minutes).

Consideration of relevant research	Workshop actions taken as a result
Visualising the dynamic changes that take place during the execution of a program is valuable for new learners (Jain <i>et al.</i> , 2006). Fagin (Fagin, 2003) describes how: variables can be taught by having a robot work with a quantity that changes while a program is running. This may be the distance a robot travels as it gives a visual analogue for a variable and enables students to see how a variable’s value can change. To demonstrate the notion of a constant using a robot, a quantity is required that does not change (e.g. a robot making a 90 degree turn)	As discussed in Chapter Four, an Information Panel, containing the value of variables and constants used in participant’s code, could be displayed within the Kebot simulator (shown in Figure 5-6). This software functionality (which was an additional modification made to Kebot from the PERS) allows the values of variables and constants to be monitored as a program runs
To do useful work students should be taught how programs must operate on data such as numbers, characters etc. These data items have to be stored in the computer’s memory. The use of analogies can be helpful for teaching the concept of variables (Currie, 2006)	A ‘shoebox’ analogy (i.e. different containers that store something, and which come in various sizes and types depending on what is being put in them) was used to introduce variables and constants. This analogy was also built upon later in the workshop when introducing the remaining numeric data types

Table 5-3. Details of previous research that influenced how variables and constants were initially introduced during the workshop.

Short tasks were set to allow differentiation between when, and how, variables and constants would be used in programming code. Key characteristics of these two constructs were discussed in addition to requirements for using them. These tasks also allowed concepts presented earlier in the session, such as basic Java syntax, to be reinforced. This section of the workshop aimed to convey basic information about constants and variables, such as naming conventions and their limitations, as both would be used extensively from this point in the workshop. Integer and doubles were used as example data types. The remaining data types were introduced later.

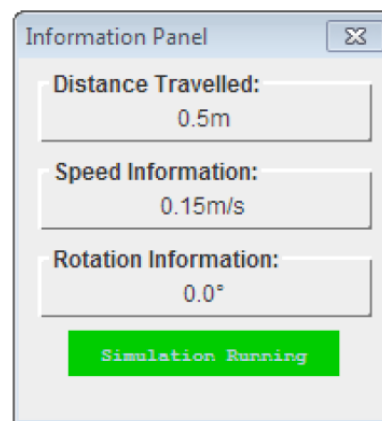


Figure 5-6. Screenshot of the 'Information Panel' available as an option within the Kebot simulator.

5.4.4 Logical Expressions

Aim: To inform participants that an Logical Expression is a programming construct that can be made up of a combination of Variables, Operators and Method invocations that evaluate to a single value (Required Time – 1 hour).

Basic Logical Expressions

Logical expressions were introduced as being the evaluation of one or more conditions (of either true or false) that are made up of a combination of methods, variables and operators. Having already encountered method invocation and variable declaration, an introduction to basic Java operators (AND && OR || Relational < > <= >=) was provided. For this section of the workshop the Berners robotic agent was used. This robot differs from the Gates robot as the information pane (shown in Figure 5-6) automatically loads when Berners is selected and Berners

starts from the same position each time the arena is loaded. The Berners class comes with the code shown in Figure 5-7. In this code an integer variable, `x`, is declared with a value of one. There is also an `if` statement which is executed if the value of `x` is greater than one (which is always the case in this program). The `forward` method is called each time the `if` is executed. Also declared is a double variable (named `distance`) that stores how far the robot has travelled (in ‘meters’). This variable stores the value provided by the pre-defined `distanceTravelled` method. The value provided by `distanceTravelled` corresponds to the value that is shown under ‘Distance Travelled’ in the pop-up Information Panel (pictured in Figure 5-6).

```
double distance = distanceTravelled();

int x = 1;

if(x > 0)
{
    forward();
}
```

Figure 5-7. The programming code which comes preloaded in the Berners robot class.

The first objective for participants is to modify the code that is provided and to instruct Berners to move forward, if distance travelled is less than 1 meter, using the distance variable already declared. Only three changes need to be made to the code provided to complete this first task. Specifically, modifications are required to the condition of the `if` statement so that the `distance` variable is referred to rather than `x`, the less than and not the greater than operator is used and the value evaluated is changed to 1.0 as opposed to 0. This section of the workshop also offered a first exposure to `if-else` and `if-else if-else` statements. These would be used extensively as the workshop advanced and the implications of using such concepts (such as the importance of ordering conditions logically) were outlined. How the condition of an `if` statement can be used to evaluate two values (e.g. `distance > 1.0 && distance < 2.0`) was discussed. It was

also outlined how `if-else if-else` statements can be used to select which block of code to execute.

The Pauser Robot

Building on this introduction to basic logical expressions, more substantial tasks were completed. These included a number of so-called 'Pauser' challenges where the Berners robot was instructed to move forward, until a certain distance had been covered, before pausing for a set period of time. A screen shot of one of the Pauser tasks can be seen in Figure 5-8. The objective for this first task was to instruct the robot to move forward, for the amount of time required to reach the first on-screen rectangle, before pausing for several seconds. As the Pauser challenges progressed they allowed for greater programming freedom and participants could choose how to tackle the tasks. The complexity of the Pauser tasks also gradually increased. An example solution to one of the final Pauser tasks is provided in Figure 5-9 with an explanation of the code. As can be seen from this code example, participants were required to demonstrate knowledge of a number of concepts including comparison operators, method invocation, parameter passing, selection and basic syntax.

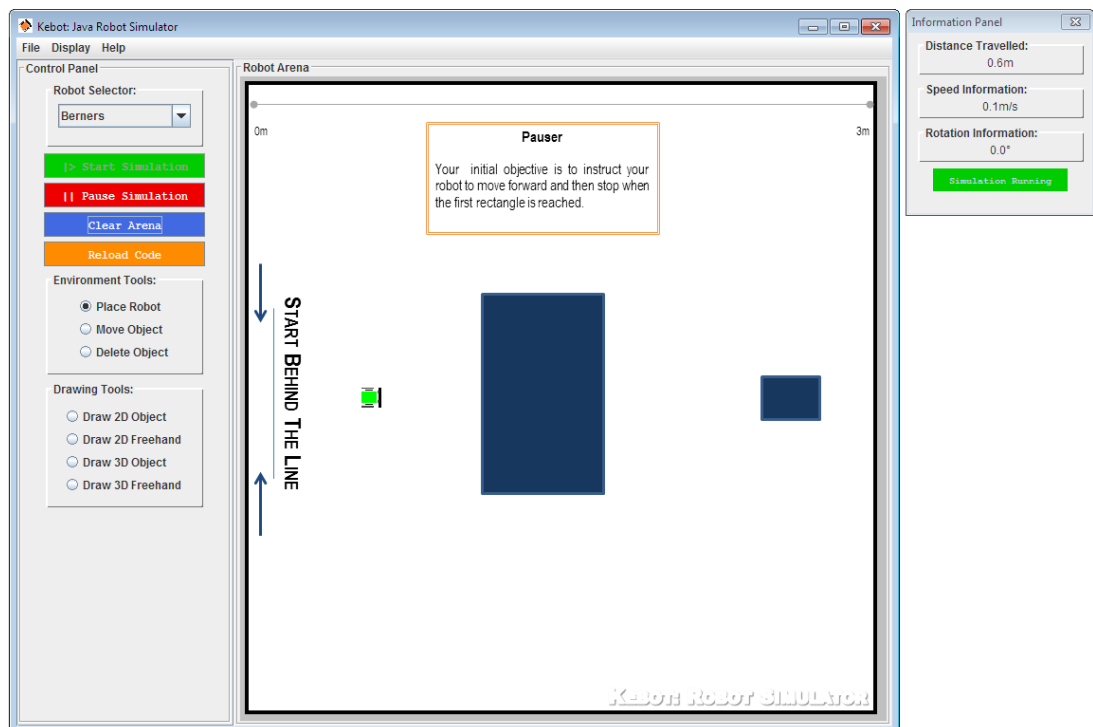


Figure 5-8. Screenshot of a robotic agent completing one of several 'Pauser Robot' tasks.

<pre> if(distance < 1.0) { forward(1); } else if(distance > 1.0 && distance < 1.2) { pause(5); forward(4); } else if(distance > 1.5 && distance < 1.7) { pause(5); forward(6); } else { stop(); } </pre>	<p>If Distance Travelled is less than 1 meter</p> <p>Move forward for 1 second</p> <p>Else if Distance Travelled is greater than 1.0 meters and less than 1.2 meters</p> <p>Pause for 5 seconds</p> <p>Before moving forwards for 4 seconds</p> <p>Else if Distance Travelled is greater than 1.5 meters and less than 1.7 meters</p> <p>Pause for 5 seconds</p> <p>Before moving forwards for 6 seconds</p> <p>Else</p> <p>Stop</p>
---	---

Figure 5-9. Example code produced during the 'Pauser' workshop tasks (with explanation).

The Shapes Arena

The introduction of logical expressions culminated with the 'Shapes Arena' (see Figure 5-10). During this challenge pseudo code was provided and previous knowledge, and examples, had to be adapted. The task aimed to ensure that the principles encountered to this point were understood.

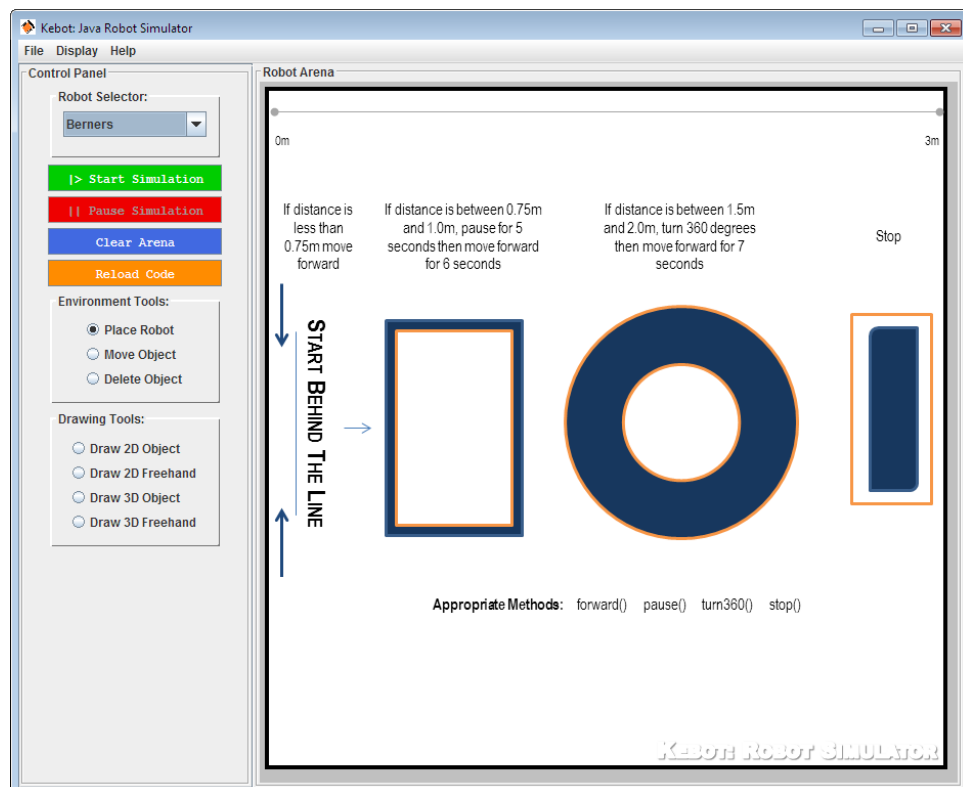


Figure 5-10. Screenshot of the 'Shapes Arena' task.

The Shapes Arena task can be seen in Figure 5-10. The main objective of the task was to use four methods (`forward`, `pause`, `turn360`, `stop`) to instruct Berners to:

- Move forward if total distance travelled is less than 0.75 meters
- If distance travelled is between 0.75 meters and 1.0 meters to pause for five seconds before moving forward for six seconds
- If distance travelled is between 1.5 meters and 2.0 meters turn 360 degrees then move forward for seven seconds
- Else stop

5.4.5 Counting, JOptionPane and Nested Statements

Aim: To introduce the concepts of Incremental and Decremental counting, the JOptionPane dialogue, basic Input/Output (I/O) and Nested Statements (Required Time – 45 minutes).

Consideration of relevant research:

Lawhead *et al.* (2002) describe a task to support the teaching of expressions. This involved students instructing a robot to respond, by beeping a certain number of times, when a robot's sensors detected a particular object. As described below, a modified version of this challenge was implemented during the workshop. Lawhead also describes how general I/O is a part of robot programming. This is because sensors and actuators return or receive information. As a result, the user transfers information to the robot (input) while the robot returns information to the user (output). Teaching I/O with robots was found to be a richer and more multi-faceted activity (Lawhead *et al.*, 2002). Fagin discusses how robots are interesting because they interact with their environment and this allows for the teaching of I/O (Fagin, 2003). Others describe how it is possible to teach I/O using robots (Cliburn, 2006; Summet *et al.*, 2009).

Line Counter

The 'Line Counter' task (seen in Figure 5-11) was designed to progressively introduce incremental counting and nested statements. With the help of pseudo code, the challenge culminated with participants programming a robotic agent to count how many lines (previously drawn by themselves) were present in the arena. The Line Counter task offered an introduction to the concept of input using sensors which would later be heavily expanded on. The `object2DDetected` method was used by participants. This pre-programmed method returns a Boolean value of true or false if a 2D object is identified by one of Berner's encoder sensors (more information of which is provided shortly). The `JOptionPane` dialogue was also presented and Boolean variables discussed. Much of the code required for the implementation of this feature was provided as were instructions for its use. Use of `JOptionPane` offered an opportunity to incorporate into the workshop a much used GUI tool, the pop-up dialogue, with which most participants would be familiar. `JOptionPane` also allowed another opportunity to consider the concept of output as it allows messages to be displayed on-screen. This complimented the more general discussion about output devices such as robot actuators etc.

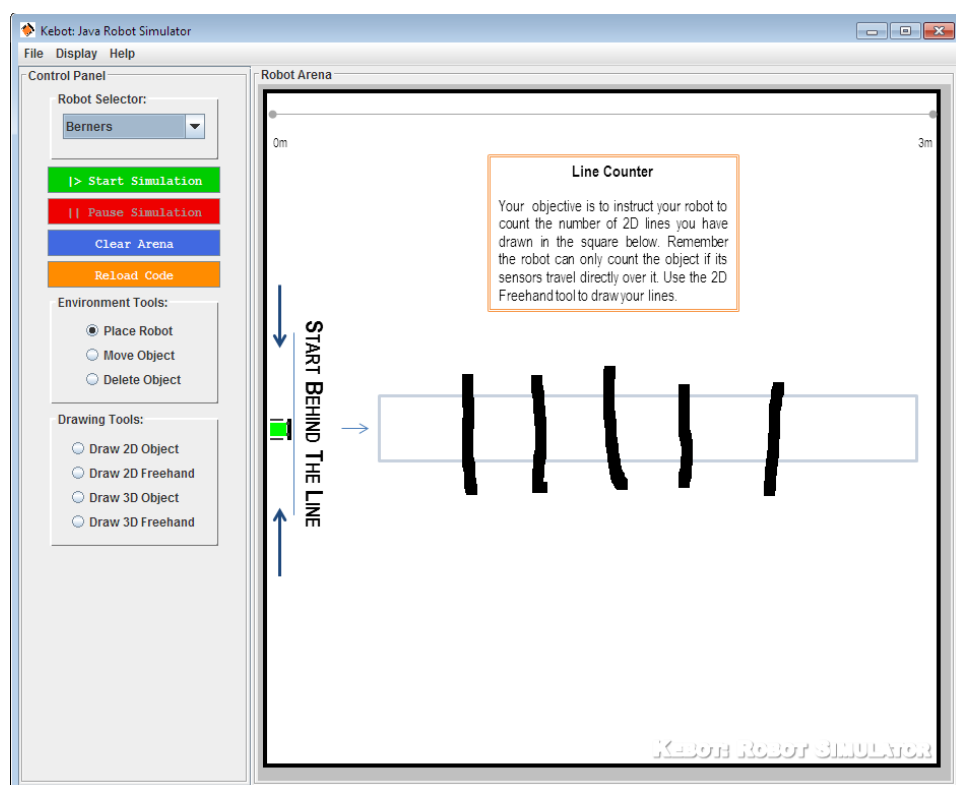


Figure 5-11. Screenshot of a robotic agent completing the 'Line Counter' task.

Boolean and Printing to the Terminal Window

Following the Line Counter tasks several short challenges were completed to investigate the Boolean data type further. The use of nested statements allowed further practice of basic logical expressions. The `System.out.println` command was presented as a means of outputting information to the terminal window. Discussion on the nature of sensors as input devices also took place that focused on how, for the robots in Kobot, the status of their sensors is constantly evaluated. Participants were asked to print to the terminal window the value of a Boolean variable called `status` in order to observe this behaviour. One of the programs completed during this part of the workshop is shown in Figure 5-12 along with an explanation.

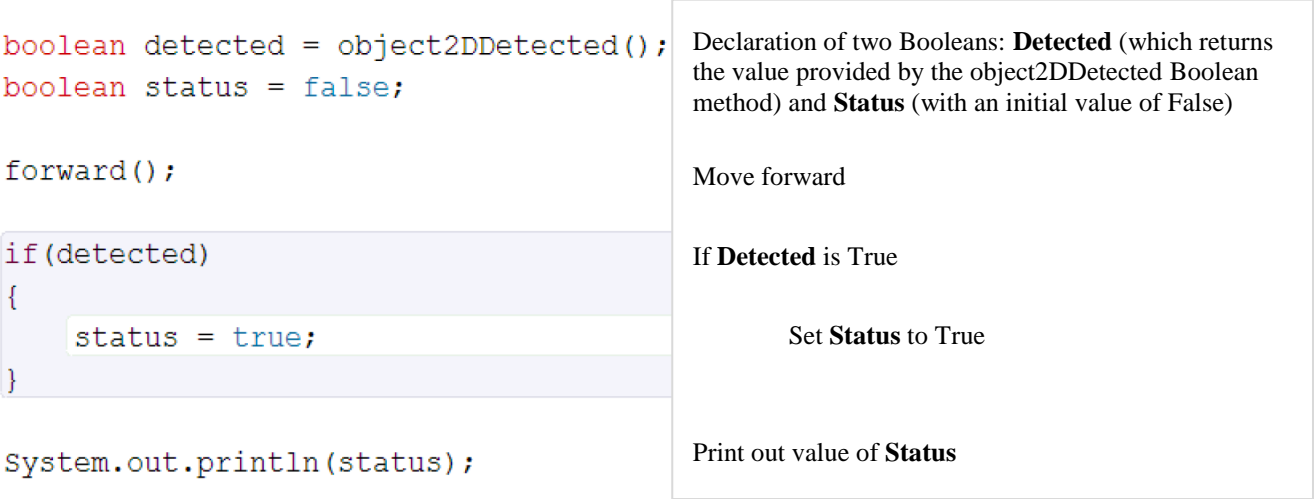


Figure 5-12. Example code produced during tasks related to Boolean variables and printing to the Terminal Window (with explanation).

5.4.6 Introducing the Remaining Data Types

Aim: To introduce participants to the remaining five Java Data Types and to give examples of when each would be used (Required Time – 15 minutes).

Consideration of relevant research	Workshop actions taken as a result
Some teaching interventions have prevented new programmers from using incorrect data types in their code (Cooper <i>et al.</i> , 2003). Visualisation tools have also been used to teach Java concepts in the past (Raab <i>et al.</i> , 2000)	Specially developed visualisation software was used, in conjunction with the Kobot simulator, which allowed the introduction of the remaining Java data types to be introduced
Access to a Java ‘Cheat Sheet’ can be helpful for the Java learning process (Daly & Waldron, 2004)	This software, named ‘Kobot: Data Types, Expressions and Operators Helper’ made use of a ‘Cheat Sheet’

Table 5-4. Details of previous research that influenced how the remaining data types were introduced during the workshop. 121

Kebot: Data Types, Expressions and Operators Helper

Three data types (Integer, Double and Boolean) were presented earlier in the workshop and previous tasks involved the use of these concepts. However, five additional types (String, Char, Byte, Short, Float) needed to be introduced. Standalone software (named Kebot: Data Types, Expressions and Operators Helper), developed by the author, allowed these concepts to be explored. In Figure 5-13 an annotated screenshot is presented that explains how the Kebot Helper was used during the workshop. Details of the Kebot Helper GUI elements are also provided. For the remainder of the session, using the Kebot simulator, information learned during this section of the workshop was reinforced as the workshop leader made reference to it at various points.

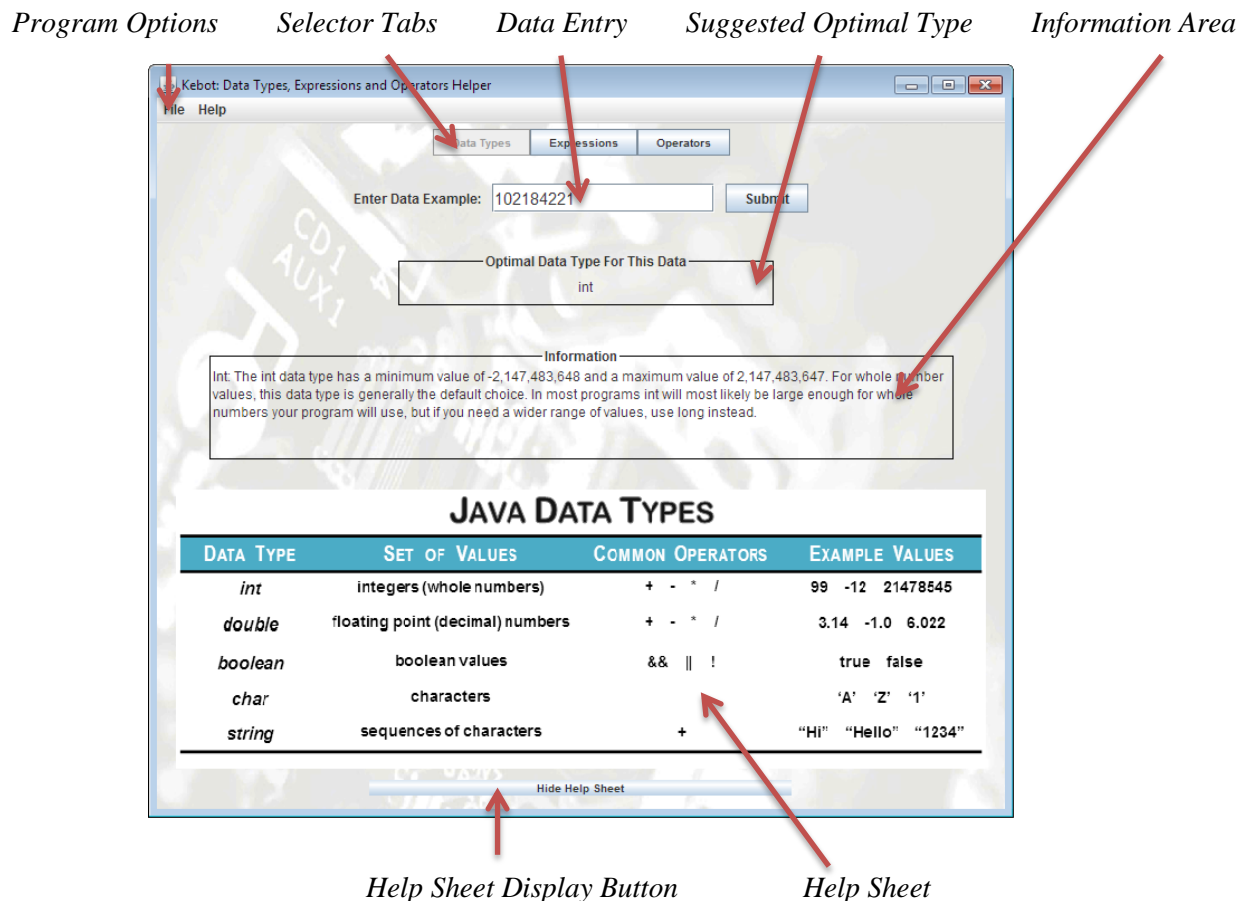


Figure 5-13. Annotated Screenshot of the Kebot Helper Program GUI.

- *Program Options*: Allows access to Help and Program Exit functions
- *Selector Tabs*: Provides the ability to select one (of three) aspects of the Helper program, although only one part of the Helper software was used during the final workshop

- *Data Entry*: A text field where data is inputted
- *Suggested Optimal Type*: A text area that provides a suggestion for the optimal data type based on information inputted into the Data Entry field
- *Information Area*: A text area that provide information related to the suggested optimal type
- *Help Sheet Display Button*: When selected either hides or displays the help sheet
- *Help Sheet*: An on-screen help (or cheat sheet) that details five primitive Java data types and some example values. This help sheet has been modelled on one suggested previously²

During the workshop, participants worked through several short challenges, using the Kobot Helper, and answered questions related to the use of data types. These tasks were intended to offer a basic understanding of when (and why) each data type could be used. This was the only part of workshop that involved use of software other than the Kobot simulator. Earlier in the project it had been intended that the Helper program would be integrated within Kobot, and would be accessed via a menu option. It was also envisaged that the Helper would play a greater role in supporting participants learning activities (note in Figure 5-13 how additional tabs are accessible from the Helper GUI but these remained unused). Such use of the Kobot Helper was, however, decided against as the workshop design progressed. This is because the aim of this research was to investigate the effectiveness of using a robot simulator, and not to evaluate software such as the Helper. During the final workshop limited use of the Kobot Helper software was made (which lasted no more than 15 minutes). The program was standalone software and did not require Kobot to function. As this segment of the workshop was only intended to offer a brief introduction to the remaining Java data types use of the Kobot Helper was considered an appropriate tool for reinforcing participant's knowledge. The Helper program was used as it offered an existing software solution, was available for rapid implementation in addition to being capable of achieving the objective of reinforcing (and introducing) participants to the remaining Java data types. The design of the Helper was influenced by the research identified in Table 5-4.

² Available online at: <http://introcs.cs.princeton.edu/java/11cheatsheet> (Last accessed 9th October 2013)

5.4.7 While and Do While Loops

Aim: To introduce the concepts of Loops in Java, using While and Do While, and to explain how iteration is used to repeat the same set of instructions until a condition is met
(Required Time – **45 minutes**).

Consideration of relevant research	Workshop actions taken as a result
Using robots it is possible to demonstrate how endless loops need not always be considered as bad (Imberman <i>et al.</i> , 2007)	Demonstrating to participants that loops are useful for repeating a single block of code, such as constantly evaluating a robots sensors
Instructing students to write code that enables a robot to navigate its surroundings is a good way of first introducing loops (Becker, 2001)	Devising several tasks that involve the use of loops to instruct a robotic agent to navigate the arena
It is best to only introduce simple loops initially to students (Lister, 2004). Requesting prompts from the user, in conjunction with loops, is an approach that can be used (Raab <i>et al.</i> , 2000)	By keeping the initial exposure to loops simple and by using participant's previous knowledge of JOptionPane to prompt limited input

Table 5-5. Details of previous research that influenced how While and Do While Loops were introduced during the workshop.

An Introduction to Loops

This section of the workshop incrementally introduced While and Do While loops through a series of tasks. For Loops were introduced later in the workshop, along with a brief introduction to Arrays, as discussed shortly. Several tasks, partly influenced by the research outlined in Table 5-5, were completed. The advantages of using loops (i.e. that they are useful when it is desired that a single block of code is executed repeatedly) was outlined as were some common problems with loops. An example of a loop within the simulator itself, designed so that the sensors of robotic agents are constantly interrogated, was discussed and a short task was completed to demonstrate how rapidly loops are able to execute. These challenges highlighted differences between While and Do While loops (i.e. that While Loops test whether a condition is true prior to looping and that Do While Loops execute code at least once as the condition is tested at the end of the loop). Example code produced during one of the Do While challenges is shown in Figure 5-14. This shows how participants implemented a Do While Loop in conjunction with several previously learned concepts (including variable declaration, method calling, decremental counting and the JOptionPane

dialogue). In this program an integer variable named `power` is initialised with a value of five. A Do While Loop is used to instruct a robotic agent (while `power` is not equal to zero) to:

- Move forward (at a speed of 0.10 meters per second)
- For a duration of one second
- Decrement `power` by a value of one
- Display the JOptionPane dialogue window displaying the current value of `power`

Once execution of the program is able to proceed beyond the Do While (i.e. because `power` is equal to zero) a second JOptionPane dialogue displays a message that states the program will end. The `System.exit(0)` method is then called and the program terminated.

```
// Insert Your Code Below

int power = 5;

do
{
    move(0.10);
    duration(1);
    power--;
    JOptionPane.showMessageDialog(null, "Power is currently " + power,
                                   "Power Information", JOptionPane.INFORMATION_MESSAGE);
}
while(power != 0);

JOptionPane.showMessageDialog(null, "The Program Will Now End",
                               "End Program", JOptionPane.INFORMATION_MESSAGE);
```

Figure 5-14. Example code produced during one of the Do While Loop tasks.

5.4.8 Method Creation

Aim: To expand and reinforce participants understanding of Methods in addition to the benefits of using them (Required Time – 45 minutes).

Consideration of relevant research	Workshop actions taken as a result
Robots provide a natural way of teaching methods. When assigned a robot behaviour that requires a series of smaller tasks, students very quickly see that these sub-tasks should be written as methods (Fagin, 2003). Methods have also been introduced as being useful when code needs to be repeated (Ladd & Harcourt, 2005)	The creation of tasks that encourage participants to consider how to optimise their code in addition to enabling them to see the advantages that using methods, in certain situations, brings
Initially allowing interaction with a robot through various methods was found to be beneficial (Cliburn, 2006) and it is possible to teach the concepts of methods and parameter passing using robots (Becker, 2001)	Throughout the workshop interaction with the robotic agents was achieved by calling various pre-programmed methods and passing parameters to these

Table 5-6. Details of previous research that influenced how the creation of Methods was introduced during the workshop.

Method Creation

Participants' experience with methods to this point involved the invocation of such constructs and the passing of parameters to them. This part of the workshop gave an opportunity to expand this knowledge and allowed participants to experiment with the creation of methods. It was believed that this would enhance understanding and would be beneficial considering that various methods had been called and used during the workshop to this point. It should be noted, however, that method creation was not explicitly outlined as a requirement in the ACM/IEEE curricular. Several short tasks involved the creation, and invocation, of simple methods created by participants. The objective of this was to highlight some benefits of using methods in programming code (e.g. when it is desired that the same piece of code will be used repeatedly) and involved participants using methods to trace shapes akin to squares and circles. The final task related to method creation involved participants creating, and invoking, their own methods during a challenge called 'Robot Dance'. This task saw participants using the `move`, `turn` and `duration` methods to program on-screen "dancing" behaviour. Example screenshots can be seen in Figure 5-15. Participants discussed amongst themselves which robot exhibited the most unique behaviour.

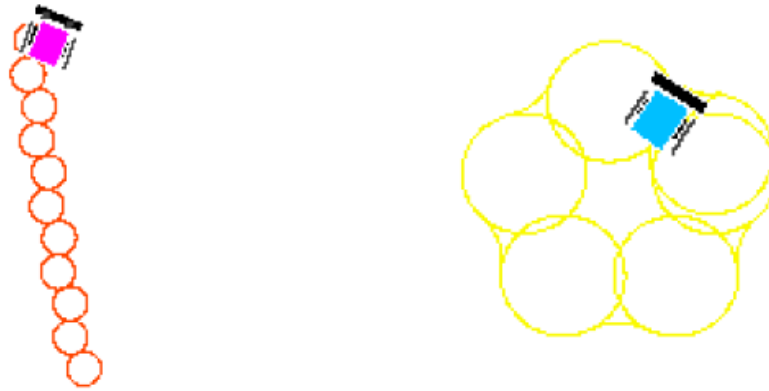


Figure 5-15. Example screenshots of robotic agents completing the 'Robot Dance' task.

5.4.9 Conditional and Iterative Control Structures

Aim: To provide the opportunity to practice and enhance programming knowledge through the introduction of additional simulator features, the undertaking of robot programming tasks and experimentation (Required Time – 2 hours).

Three methods, previously used briefly during some of the method creation tasks, were discussed in greater detail at the start of the second day of the workshop. Descriptions of these three methods are outlined in Table 5-7.

Method Name	Details
<code>move()</code>	<p>Used to move forward or backward</p> <p>Speed is passed as a parameter in a range between 0 – 0.3 m/s</p> <p>To move forward a positive value is passed – e.g. <code>move(0.1)</code></p> <p>To move backwards a negative value is passed – e.g. <code>move(-0.1)</code></p>
<code>turn()</code>	<p>Used to turn left or right</p> <p>Rotation speed is passed as a parameter in a range up to 180 degrees per second</p> <p>To turn left a positive value is passed – e.g. <code>turn(90)</code></p> <p>To turn right a negative value is passed – e.g. <code>turn(-90)</code></p>
<code>duration()</code>	<p>Instructs how long to perform the previous command for:</p> <pre> move(0.1); duration(10); </pre> <p>(e.g. move forward at 0.1m/s for 10 seconds)</p>

Table 5-7. Robot control methods used extensively during Day Two of the workshop.

This section of the workshop was the most substantial as it lasted for just over two hours. Use of the `move`, `turn` and `duration` methods enabled participants to exercise greater control over the behaviour of their robotic agents. This is because these methods have fewer restrictions than the majority of others used to this point. Tasks completed during the early workshop were more prescriptive in order to familiarise participants with fundamental programming concepts and the Kebot simulator. This is in contrast to those tasks completed during Day Two which allowed greater experimentation and could be completed in multiple ways.

‘Figure of 8’ and an introduction to Proximity Sensors

For this section of the workshop the Jobs robot was used. Early challenges were designed to demonstrate that there is often more than one way to solve programming tasks and to introduce the control methods listed in Table 5-7. An early task involved participants instructing a robot to perform a ‘Figure of 8’ (as seen in Figure 5-16 along with one of several potential solutions). Participants were provided with some partial code but were mainly expected to use their existing knowledge and a process of experimentation.

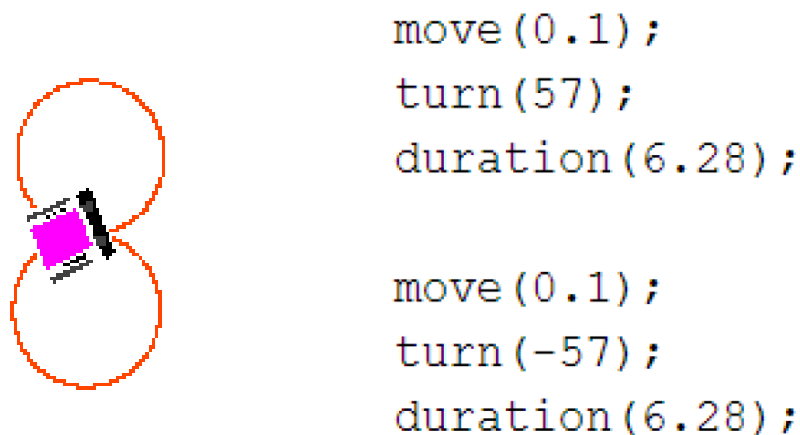


Figure 5-16. A robotic agent performing a 'Figure of 8' - one of the first challenges undertaken during Day Two of the workshop (along with one potential solution).

In addition to these new robot control methods, proximity sensors were also introduced and tasks were devised to investigate this functionality. In Kobot two proximity sensors allow a robot to detect 3D objects directly in front and this enables programs to be written whereby objects are sought out or avoided. The maximum range in which the proximity sensors can detect an object is 0.8 meters. During Day One some challenges involved use of proximity sensors through restricted pre-programmed methods. This was to ensure that participants were able to initially concentrate on the fundamental concepts being introduced, and were not preoccupied by the intricacies of more complicated means of accessing information provided by sensors. By using more advanced sensor methods during Day Two (named `leftSensorValue` and `rightSensorValue`) participants were able to assert greater control over their robots. The `leftSensorValue` and `rightSensorValue` methods are used to calculate the distance between the robotic agent and 3D objects. 3D objects, which are coloured a dark-red, are described as being akin to walls that cannot be climbed. Contact with 3D objects results in the simulation being halted and code must be reloaded in order to continue the simulation. To develop participants' understanding of the proximity sensors, the `System.out.println` command was used to determine the maximum range of these sensors. This was done by introducing, and moving, 3D objects away from a robot placed in the arena and by printing the value of `leftSensorValue` and `rightSensorValue` to the terminal window (as pictured in Figure 5-17).

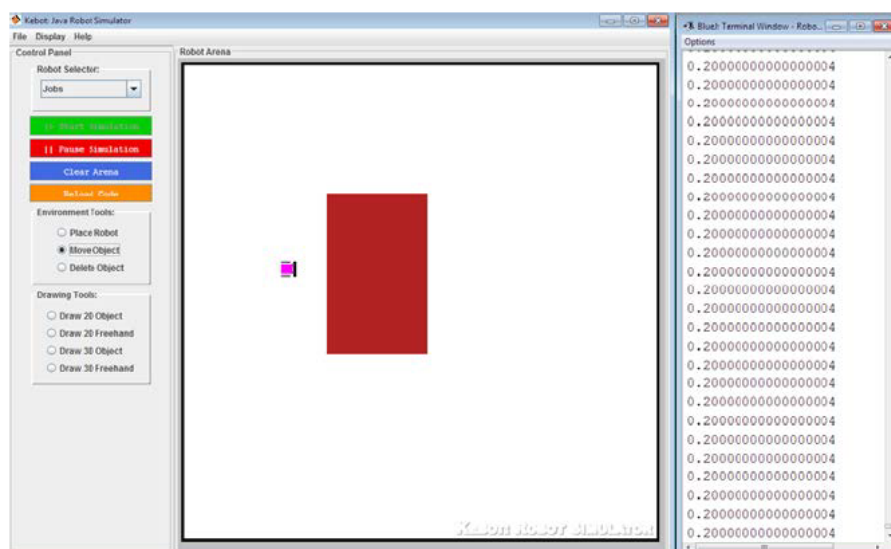


Figure 5-17. Screenshot showing a robotic agent with a 3D object drawn in the arena. The values of the agent's proximity sensors are shown printed to the Terminal Window.

In Figure 5-18 code can be seen that demonstrates how the proximity sensors are used, along with conditional statements and movement methods, to control the behaviour of robots. This example program demonstrates how the values provided by the `leftSensorValue` and `rightSensorValue` methods are assigned to variables of type `double` (in this case named `leftSensor` and `rightSensor`). An `if-else` statement has also been declared. In this example a robot will move forward (at a speed of 0.1 meters per second) *if* the values of the `leftSensor` and `rightSensor` variable are greater than 0.5 (i.e. if a 3D object is over 0.5 meters away). Otherwise the `else` statement ensures the robot will not move (represented by a speed of 0.0 meters being passed to `move`). It should also be noted that the Kobot robots come with three other sensors – encoder sensors – which are in addition to the two proximity sensors. Encoder sensors are able to detect 2D objects on the arena floor. The encoders were not used during the main workshop itself as the post-workshop assessments involved the development of programs that used these sensors (as outlined in Chapter Six). This meant that participants' code could not be copied from earlier tasks during the assessments.

```
double leftSensor = leftSensorValue();
double rightSensor = rightSensorValue();

if(leftSensor > 0.5 && rightSensor > 0.5)
{
    move(0.1);
}
else
{
    move(0.0);
}
```

Figure 5-18. Example code that demonstrates how the proximity sensors are used in the Kobot simulator.

Programming Robot Interaction and Mimicking Intelligence

Following a recap of concepts previously introduced, several substantial tasks were completed. It was intended that participants would be able to apply (and further advance) their existing programming knowledge through a process of experimentation. Participants were asked to

complete a number of challenges that focused on the elicitation of various behaviour from the Jobs robot including:

- *'Scared' Behaviour* – where a robot moves away from an object if the object is too close
- *'Seeking' Behaviour* – where a robot moves around the arena actively seeking out objects
- *Simple 'Wall following' Behaviour* – where a robot is able to 'follow' walls within a square
- *'Avoiding' Behaviour* – where a robot moves around the arena, in the process avoiding making contact with any objects that are present
- *'Curious' Behaviour* – where a robot moves away from an object if it is close, towards an object if it is far away or otherwise stays stationary.

In Figure 5-19 one simple solution to the 'Curious' Behaviour task can be seen while Figure 5-20 provides visual examples of 'seeking' and 'avoiding' behaviour.

```
double leftSensor = leftSensorValue();
double rightSensor = rightSensorValue();

if(leftSensor < 0.3 && rightSensor < 0.3)
{
    move(-0.1);
}
else if(leftSensor > 0.6 && rightSensor > 0.6)
{
    move(0.1);
}
else
{
    move(0.0);
}
```

Figure 5-19. Example code that demonstrates one solution to the 'Curious' Behaviour robot programming task.

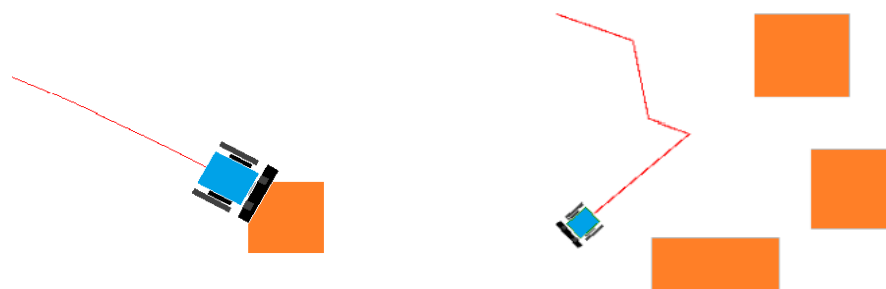


Figure 5-20. Screenshots of robotic agents exhibiting 'seeking' (Left) and 'avoiding' (Right) behaviour.

5.4.10 For Loops

Aim: To provide a basic introduction to the For Loop (Required Time – 45 minutes).

Introducing For Loops

As already outlined, participants gained experience using While and Do While Loops during the workshop. To ensure exposure to all of Java's iterative concepts, the For Loop was also introduced and several tasks were completed. Participants were already familiar with the three components of For Loops (*initialisation* – through their experience declaring variables / *test* – through their experience creating simple logical expressions / *increment/decrement* – as several tasks had involved use of these concepts). For Loops were presented as being a powerful means of iterating over a range of values. Initial challenges involved using For Loops to achieve repetitive tasks, such as printing several thousand numbers (e.g. counting up from 1 to 100 000), rapidly in the Kebot software. Examples were provided to highlight the benefits of using For Loops.

The Black Hole

The most significant For Loop challenge, the 'Black Hole', demonstrated how For Loop control variables can be passed to methods in the same manner as other parameters. Using knowledge gained during the previous tasks, participants were asked to simulate their robotic agent falling into a 'Black Hole' using a For Loop. Pseudo code was provided. A working solution can be seen in Figure 5-21 while the task itself is pictured in Figure 5-22.

```
for(int i = 0; i < 100; i++)
{
    move(0.2);
    duration(0.3);

    turn(i);
    duration(0.3);
}
```

Figure 5-21. Example code that demonstrates a solution to the 'Black Hole' task.

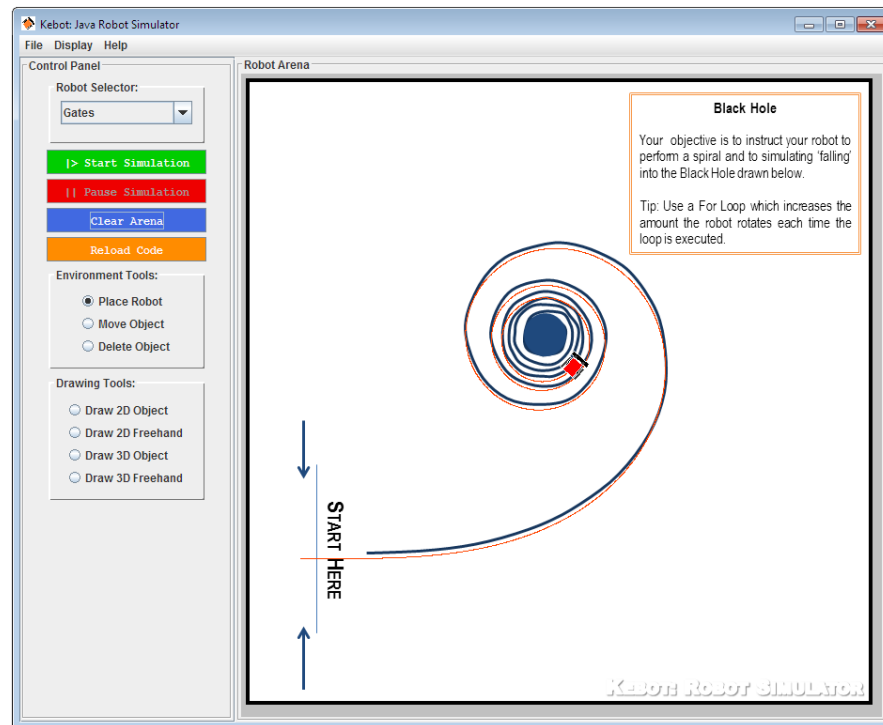


Figure 5-22. Screenshot of a robotic agent completing the 'Black Hole' task.

Following successful completion of the task it was demonstrated how, to achieve the same repetitive behaviour without using a For Loop, a substantial amount of code (in this case an additional 300 lines) would have to be written.

5.4.11 Arrays

Aim: To provide a general introduction to the concept of arrays and to allow an opportunity for the further implementation of other programming concepts learned during the workshop (Required Time – 30 minutes).

Consideration of relevant research:

Arrays are often difficult for novices to understand (Eagle & Barnes, 2008). Previous research has shown, however, that it is possible to use robots and their sensors to introduce the concept of basic arrays (Fagin *et al.*, 2001; Lawhead *et al.*, 2002).

Introducing Arrays

The ACM/IEEE computing curricula does not specify that arrays have to be taught as part of the ‘Fundamental Constructs’ module. After a discussion with a PhD supervisor (TK), the decision to introduce arrays was taken as it allowed an opportunity for participants to further apply programming knowledge gained during the workshop. It was also anticipated that an introduction to arrays would give a rudimentary understanding of the concept and would help prepare participants for any future programming activities. It was not intended, however, that a detailed overview would be provided. Knowledge of arrays is not assessed by any of the programming exercises used during or following the workshop. Arrays were introduced as being an efficient means of storing variables of the same type. Instructions were provided to participants which details how to declare and use arrays. This was more prescriptive compared to other elements of the workshop given that the introduction to arrays was viewed as supplementary (due to arrays not appearing in the list of fundamental programming constructs identified by the ACM/IEEE).

Barcode Reader

The ‘Barcode Reader’ task allowed the use of Arrays to be demonstrated. This involved participants instructing their robotic agent to ‘read’ a barcode they had drawn in the robot arena. Their robot was to move from left to right, ‘scanning’ and storing the location at which each element of the barcode was found in an array. A guide and pseudo code were provided to assist participants. It was still required, however, that programming knowledge acquired during the workshop was adapted and a range of previously learned concepts were used. After an initial attempt at solving the challenge with a ‘standard’ array, participants were encouraged to use a For Loop to print the values of multiple array elements. A screenshot of this task can be seen in Figure 5-23. Further information on the Barcode Reader task is not provided given the more prescriptive nature of the task and as the teaching of Arrays does not appear in the ACM/IEEE set of programming fundamentals.

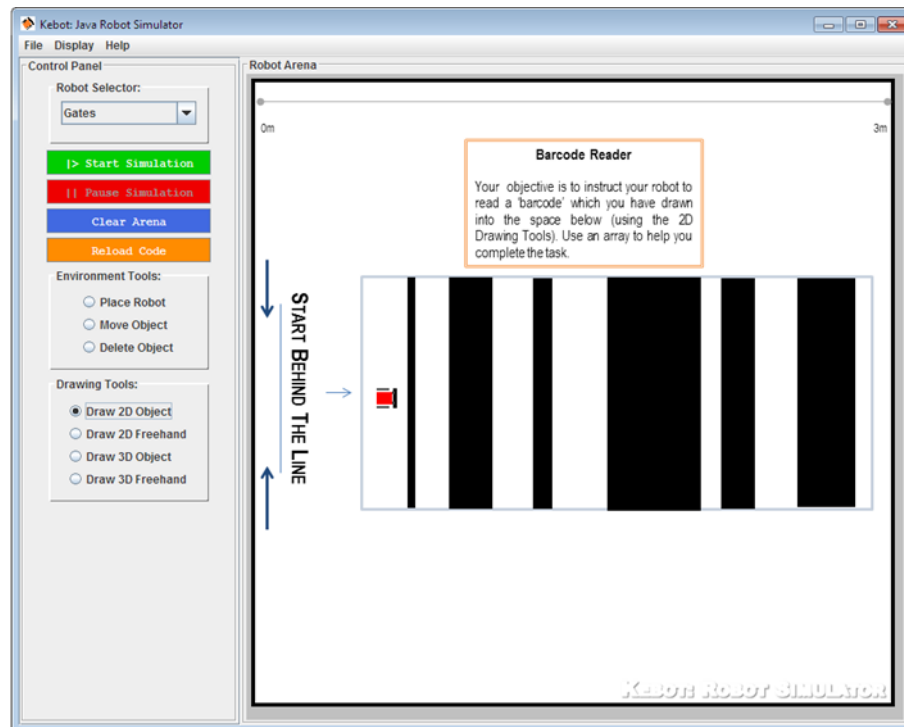


Figure 5-23. Screenshot of a robotic agent completing the 'Barcode Reader' task.

5.5 Study Limitations

Like all empirical research this case study had a number of limitations. In this sub-section 'high-level' limitations relating to information previously presented in this chapter are discussed. These limitations are applicable to all workshops that were later performed and were noted in advance of running such sessions. Limitations relating to the case study methodology, the programming concepts introduced, the workshop content/structure, the Kobot simulator, and the workshop leader are presented.

Validity is closely related to study limitations. Validity denotes trustworthiness and to what extent the results are not biased by a researcher's subjective point of view (Runeson *et al.*, 2012). A scheme previously devised to distinguish between four aspects of validity (construct validity, internal validity, external validity and reliability) has been suggested by previous research (Runeson *et al.*, 2012; Yin, 2009). Unlike the limitations of a study it is not possible, however, to evaluate all threats to validity until data analysis has been completed (Runeson *et al.*, 2012). Details of the threats to validity (such as those related to data collection instrumentation or participant

involvement) are, therefore, presented in the discussion section of this thesis in Chapter Eight. Four general sources of literature were used to help identify the limitations of this work (Creswell, 2009; Rosnow & Rosenthal, 1997; Runeson *et al.*, 2012; Yin, 2009). Observations made by the author when preparing this research, and discussions with more experienced researchers during the execution of the project, also contributed to the limitations that are outlined.

Limitations of the Case Study Methodology

Replication of case study research is not possible in the same manner as it is for other research strategies such as experiment. ‘In-case replication’ (as both the trainee teachers and student programmers workshop were run twice with independent groups), and the use of validation strategies (such as a search for rival explanations), ensure that the findings of the case study make a significant contribution to knowledge. In addition, a ‘traditional’ baseline (to which the effectiveness of the approach investigated can be compared) is not used during this case study. Due to this the results cannot provide a quantifiable measure of effectiveness that can be contrasted with other approaches (such as a statistical value that highlights numerically the effect of the intervention contrasted with alternatives). The richer findings of case study, however, compensate for this absence of a traditional baseline. Moreover, it has still been possible to establish a qualitative baseline by asking participants to compare any previous programming learning experience to the one using Kebot.

Limitations of Programming Concepts Introduced and the Workshop Itself

Only the fundamental programming constructs identified by the ACM/IEEE Joint Task Force were introduced during the workshops (with the exception of a brief introduction to arrays). Whilst the ACM/IEEE detail a number of programming concepts and learning objectives (in addition to a recommended coverage time), more specific guidance is limited. As a result, decisions needed to be made in regards to how much detail was required to cover each programming concept. Discussions were held with a PhD supervisor (TK), who has greater programming teaching experience, to develop a suitable strategy. Regardless, decisions made on what content to include

may, in some way, limit the research findings. For example, the decision to introduce only basic logical expressions during the workshop (and not arithmetic expressions), or the decision to introduce all three of Java's iterative constructs (instead of concentrating on just one) may have had an adverse effect in some way. Whilst beyond the scope of this research, as the emphasis of the project was on the teaching of introductory programming alone, the results of the case study are also not able to offer evidence that a robot simulator is useful for supporting the learning of advanced programming concepts. The language selected for use during this research, Java, may also be considered as a potential limiting factor. This is because it cannot be ascertained how generalisable the findings of the case study are compared to alternative programming languages and non-object-oriented paradigms given that Java is an object-oriented language only. Java is considered, however, to be a general purpose language and shares commonalities in its fundamentals with other popular languages such as C#³. Moreover, only basic programming concepts were introduced and more advanced language features were not drawn upon.

The nature of the workshop session must be considered because the performance of some participants, especially ones not suited to long periods of concentrated learning, may have been negatively affected by the intensive nature of the approach. This could have impacted upon effort exerted by participants. The risk of participant drop-out was also identified as a potential limiting factor. Because of this an attempt was made in advance to ensure the commitment of all persons invited to be involved. Whilst it was intended that the workshop would be relatively relaxed, without the feel of a 'formal' experiment, it was not possible to truly recreate 'real-world' conditions. This may have led to changes in participant performance that could have distorted the results of the study. Such distortion effects may have resulted in either more positive or negative results due to the fact that participants were aware that they were being observed. The use of data triangulation and multiple sources of evidence, however, is considered to have minimised the risk of such effects.

³ <http://msdn.microsoft.com/en-us/library/ms836794.aspx> (Accessed 1 October 2013)

Limitations of the Kobot Robot Simulator

The fact that the Kobot simulator was a modified version of the PERS can be considered as a potential limitation. This is because the PERS was not purpose built for supporting the teaching of programming. A traditional software design process (encompassing requirement elicitation and other similar phases) was not, therefore, undertaken. If this had been the case then some features of the final simulator may have been different. As feedback on the PERS was generated (and later acted upon) during the exploratory studies, and because educational software guidelines were considered when making modifications to the PERS, the risk that there would be significant usability issues with Kobot was, however, minimised. The Kobot software itself can also be considered to have limitations. Firstly, Kobot can be resource intensive and the performance of the simulator can diminish (although not to a critical level) on older PCs. The Java Development Kit (JDK) is also required to be installed and this can take up significant hard-disk space (> 400mb for both the JDK and Java Runtime Environment). Secondly, Kobot offers a 2D top-down perspective of the robot arena. It was identified beforehand how many potential users of Kobot would likely be familiar with advanced computer gaming systems such as the Xbox 360. A sample of such users may be inclined to compare their previous exposure to video games to the one using Kobot. It was thought that this could lead to frustration, which could have impacted upon participants attitudes during the workshop, because the simulator is not as advanced graphically as these alternatives.

Limitations of the Workshop Leader

As with all research that involves an experimenter taking an active role in the study itself, there was a risk that the workshop leader would unconsciously affect the conduct of the research. This could be because a researcher gets to know those involved personally (and potentially begins to favour particular participants) or as a researcher has a vested interest in the implemented approach being successful (having spent time developing it). The inexperience of the author must also be contemplated. It is possible that relative newcomers to research may be inclined to misinterpret results, in addition to inadvertently ignoring potentially important outliers. Maintaining an audit

trail, the adoption of a data triangulation strategy and a search for rival explanations help to reduce the possible impact of such factors. Finally, the fact that this research involves a significant teaching component must be considered. Despite having some teaching experience, the author alone was responsible for delivering the workshop sessions. As these workshops progressed this may have led to a growth in confidence and, as a result, a change in teaching style. Moreover, the fact that an inexperienced educator designed the sessions may have led to issues with some workshop content. The review of the workshop in advance of it being delivered, by a more experienced researcher (TK), and the running of a trial workshop help to reduce the potential impact of such factors.

5.6 Summary

In this chapter details of a case study, designed to achieve the research objectives of this thesis, have been presented. Case studies are highly flexible and well suited for supporting a variety of research projects. The development (and review) of a case study protocol ensures that reliable, transparent and rigorous work has been performed. Rationale for, and discussion on the appropriateness of, the research design are given. Why a case study approach was considered the best choice is also outlined. An introduction to the sources of data used, in addition to the participants who took part, has also been presented. Influenced by the findings of the Systematic Literature Review (SLR) and exploratory studies, the Kobot robot simulator was used during the case study. This allowed for an investigation into the effectiveness of using simulated robots as programming teaching tools. An associated 10-hour workshop was created to support the teaching of introductory programming concepts. An overview of the structure and content of this workshop have been provided and study limitations discussed. In the following chapters further details of the study design, along with the results of the research, are described. In Chapter Six the first case, ‘Trainee Teachers’, is presented. This is followed by the second case, ‘Students’, in Chapter Seven.

Chapter Six

Case One: Trainee Teachers

Further details of the execution and results of one component of the case study, Case One – ‘Trainee Teachers’, are presented in this chapter. Based on the methodology outlined in Chapter Five, two workshops were completed as part of this case. In total, 22 trainee Information Communication Technology (ICT) / Computer Science (CS) teachers took part. All had learned programming concepts in some capacity previously. This allowed an opportunity to compare participants’ prior learning experiences with their experience using the Kobot robot simulator in addition to establishing their views on the effectiveness of the method. The research presented aimed to determine perceptions and opinions of programming through the use of questionnaires. Details of other data collection activities, involving an in-workshop observation log and several programming exercises, are also provided.

6.1 Introduction

This chapter builds on the case study design, presented in Chapter Five, which was devised to achieve the objectives of the thesis. Details of a workshop, developed to determine the effectiveness of the Kebot robot simulator as a tool to support the learning of introductory programming, have previously been given. Several research propositions were also identified. In the preceding chapter it was established how this case study is a two-case case study. High level details related to the research design, such as information about workshop content, were described. Specific details relating to the execution of the research, however, were not provided. As a result, in Section 6.2, information is outlined in regards to: the participants who were involved, the workshop procedure and setting, the data collection instruments and the data analysis strategy.

The involvement of participants was secured after Keele University's PGCE course leader for ICT made contact with the author enquiring about the possibility of these taking a part in the workshop. As most participants had past programming knowledge this allowed an additional opportunity to gain unique feedback, as this previous experience could be contrasted to that using Kebot. The success of the exploratory workshop involving trainee teachers (reported in Chapter Three) also influenced the decision to involve such participants. This is because participants involved in the exploratory studies provided feedback that positively impacted later activities. It was thought that the involvement of similar cohorts would likewise contribute to the research. The work reported in this chapter had the following aims:

- 1) To compare participants' prior programming learning experience to the one using Kebot
- 2) To establish opinions on the effectiveness of the Kebot robot simulator
- 3) To determine general opinions of programming

Collected has been used to address the case study research question and propositions in Chapter Eight. Case One also allowed the creation, testing and validation of programming exercises in advance of studies involving students (reported in Chapter Seven).

6.2 Study Execution

In this section information relating to the execution of Case One of the case study is presented.

6.2.1 Participants

Two groups of participants took part and two separate workshops were held:

Trainee Teacher Workshop One (TTW1) – 17 participants (nine males, eight females) enrolled on an ICT Postgraduate Certificate of Education (PGCE) course at Keele University (2011/2012 Academic Year). All participants were awarded Qualified Teacher Status (QTS) in the months following the workshop and had gained teaching experience by the time of the research. One participant did not attend Day Two of the workshop. This meant that 16 participants completed the workshop in total. The results of TTW1 are presented in Section 6.3.

Trainee Teacher Workshop Two (TTW2) – Five participants (two males, three females) enrolled on an ICT Postgraduate Certificate of Education (PGCE) course at Keele University (2012/2013 Academic Year). As with TTW1, all had gained teaching experience by the time of the research. All five participants attended the full workshop. The results of TTW2 are presented in Section 6.4.

In total 22 participants were involved during Case One. Both cohorts were registered on the same course in name, but during different academic years (TTW1 participants 2011/2012 / TTW2 participants 2012/2013). The full complement enrolled on each respective course took part. The reason why there was a lower number of trainees enrolled during the 2012/2013 academic year, compared to 2011/2012, is unknown.

6.2.2 Workshop Procedure and Setting

TTW1 took place in June 2012 while TTW2 took place in October 2012. TTW1 took place towards the end of the academic year while TTW2 was hosted near the start. This was not considered to be a problem as the PGCE course onto which all participants were enrolled contained no other programming material. Both workshops followed the procedure outlined in Chapter Five and were hosted over two full days in a computer lab at Keele University. Participants were assigned to an individual PC, read an information sheet and completed a consent form. Each was also given a code number. No deviations to the workshop procedure occurred.

6.2.3 Data Collection Strategy

Several data sources were used during Case One of the case study. The procedures for the pre-workshop questionnaire, the post-workshop questionnaire and the researcher's workshop log were the same for both groups. Other instruments that were used (specifically two in-workshop, and post-workshop, programming exercises), however, differed. In Figure 6-1 a diagram can be seen that displays the different data collection instruments used.

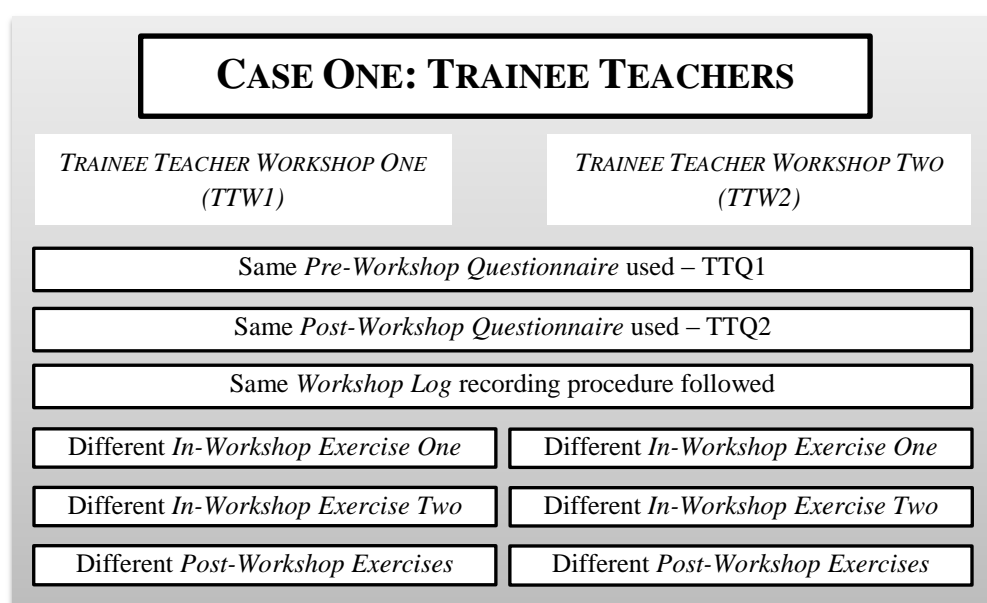


Figure 6-1. Data collection instruments used during Case One: Trainee Teachers

Trainee Teacher Questionnaire One (TTQ1) and Two (TTQ2)

The questionnaires used in TTW1 and TTW2 (TTQ1 and TTQ2) were similar to those used during the exploratory studies (EQ1 and EQ2). Relevant literature was consulted (Oppenheim, 2000) and suggestions for avoiding common problems were followed (de Vaus, 2002). The decision to use modified versions of EQ1 and EQ2 was taken as these had been successfully used during the exploratory studies and had facilitated the collection of a rich set of data. There are, however, differences between the two sets of questionnaires. Firstly, presentation and layout differs. This was due to procedural reasons (as when updating the ethical approval application it was noted how changes were required) and after considering feedback received as a result of the case study protocol review (as this led to the inclusion of additional questions). Both TTQ1 and TTQ2 were trialled with three PhD students to determine if there were any issues with spelling or layout. The PhD supervisor (TK) was also consulted.

TTQ1 and TTQ2 allowed an opportunity to collect a substantial amount of data, from a number of participants, at the same time. These questionnaires were designed to be completed in several minutes and, with them being distributed at the start and end of the workshops, they did not impact upon the running of the sessions. Both questionnaires were designed to be quantitatively, and qualitatively, analysed. Content was devised after considering the objectives of the research and the case study aims. The questionnaires consist of open and closed questions. The use of open questions allowed for the collection of information that could be used to corroborate responses to closed answer questions. Copies of TTQ1 and TTQ2 can be found in Appendix A5.

TTQ1 determines:

- Level of past programming experience
- Programming language most recently learned
- Previous method of learning programming
- Amount of time spent learning programming
- Views on previous method of learning programming

- Enjoyment of past programming experience
- Familiarity of introductory programming concepts
- Issues associated with learning to program
- Stereotypes associated with learning to program
- Views on the current teaching of programming in high schools
- Own confidence in their ability to teach programming in a high school
- Perceived difficulty teaching programming using their existing knowledge
- Gender

TTQ2 determines:

- Enjoyment of their programming experience during the workshop
- Difficulty in completing the programming tasks set during the workshop
- Views on Kobot as a tool to introduce basic programming concepts
- Aspects of Kobot liked
- Aspects of Kobot not liked
- Views on the workshop session in general
- Views on Kobot compared with their previous programming learning experience
- Views on teaching programming in a high school
- Own confidence in their ability to teach programming in a high school
- Perceived difficulty teaching programming having now seen a tool such as Kobot
- Views on whether programming should be part of the National Curriculum in schools
- Opinion on whether they would consider using a robot simulator in their own lessons
- Other comments

Trainee Teacher Workshop Log

A record of events was maintained during the workshops and this supplemented other data collected. The workshop leader kept a personal log of incidents or issues that occurred according to pre-determined criteria. It was believed this enhanced the reliability of other data collected and offered a consistent means for the researcher to record their thoughts. Literature related to participant observation (Robson, 2011), and researcher self-reflection (Marotto *et al.*, 2007), was considered. Self-reflective electronic journals, which detail observations, are often used as a supplementary method to collect data (Marotto *et al.*, 2007; Robson, 2011). The completion of a workshop log offered an efficient and direct means of recording potentially important information. Such an approach, however, was considered only to be a supportive (i.e. an additional) means of data collection. This is because it was recognised the workshop leader would be pre-occupied with delivering the session itself. The recording criteria were as follows:

To Be Documented Immediately During the Workshop

- Issues with environment/equipment (e.g. problem with laboratory, technical failure)
- Issues involving participant involvement (e.g. non-attendance, missing participants)
- Other unidentified potentially critical issues

To Be Documented As Soon As Possible After the Workshop

- Were most participants “on task” during the workshop? If not, why?
- Did participants seem enthusiastic throughout the session? If not, why?
- What aspects of the workshop/programming concepts did participants struggle to grasp?
- Did three or more participants voice concerns about a particular aspect of the workshop?
- Did three or more participants voice concerns about a particular aspect of the simulator?
- Were any concerns raised about the nature of assessment used during the workshop?
- Was a considerable amount of time spent diverging from workshop related activities?
- Are there any other points that need to be noted?

In- and Post-Workshop Programming Exercises

Case One allowed an opportunity to create, test and validate programming exercises in advance of two workshops involving student programmers (reported in Chapter Seven). The programming exercises were not used during TTW1 and TTW2 to assess learning as it was assumed beforehand that these exercises would offer little challenge to participants with quite substantial prior programming experience. It was, however, acknowledged that interesting data might be collected. Various constraints meant that it was not possible to test the programming exercises used during TTW1 in advance. Whilst it was hoped that the exercises would not have to be modified following TTW1, hence enabling the same instruments to be used during all four workshops, it was accepted that there was a possibility that this could happen due to these not being trialled previously in a workshop environment. Indeed, after the completion of TTW1, the need for such modification became apparent. This explains why different exercises were completed during TTW1 and TTW2.

Programming knowledge is normally determined by using (McCracken *et al.*, 2001):

1. *Objective testing* – for example multiple-choice exercises, which provide an effective means of determining knowledge about areas such as language syntax or program behaviour. Objective testing can give instant feedback and is suitable for both formative and summative assessment. Multiple-choice tests have come into favour as a useful evaluation tool on CS courses (Roberts, 2006). The use of selected response instruments (such as multiple-choice) is one of the main strategies for the assessment of learners (Stiggins, 2005).
2. *Performance-based assessment* – involves the evaluation of programming proficiency. Unlike objective testing, performance-based assessment directly determines an individual's ability to create computer programs. Due to the act of programming being a process, programming ability must be assessed by considering code (Daly & Waldron, 2004). Lab-based assessments are effective in determining the programming performance of a large group of people (Bennedsen & Caspersen, 2006). This is because they provide learners with a valuable and appropriate means of testing themselves (Haghighi & Sheard, 2005).

Multiple-choice/constructed response tests (two in-workshop programming tests) and programming exercises (four post-workshop challenges) were developed. It was believed that in-workshop programming exercises would allow understanding of syntax and program behaviour to be tested (McCracken *et al.*, 2001), in addition to enabling in-workshop progress to be monitored. It was considered important to do this given the length of the workshop (which lasted for more than ten hours). The exercises also offered an opportunity to detect problems with material delivered (e.g. if a number of participants incorrectly answered a particular question). The in-workshop exercises complimented the workshop log as these together allow a broad picture of events to be established.

After the workshops a third programming exercise, which involved the completion of four programming tasks, was administered. Code was collected (onto a USB) and later graded. The challenges were distributed on A4 paper. The post-workshop exercises allowed actual programming ability to be considered. The post-workshop exercises were completed under “test conditions” (i.e. individually in silence) to ensure performance could be accurately monitored. For the post-workshop exercises a robotic agent not encountered during the workshop, named Page, was programmed. Page differed from the other robots previously used. This is because Page is equipped with eight proximity sensors (opposed to two) as shown in Figure 6-2.

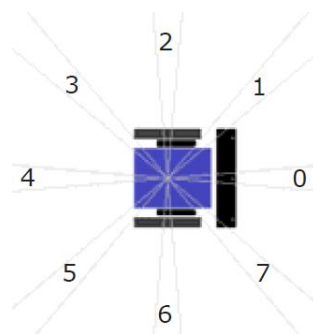


Figure 6-2. Screenshot of the Page robotic agent used during the post-workshop programming exercises.

Page also has three encoder sensors capable of detecting 2D objects drawn on the arena floor. Whilst encoder sensors were introduced during the workshop, the specific control methods required to use each of these was not. As a result, participants were introduced to these proximity and encoder sensors for the first time during the post-workshop exercise. In regards to the student programmers, use of Page enabled deep learning to be established. Deep learning is when a learner

aims towards understanding whereas surface learning is where learners aim to simply reproduce material without actually understanding (Case, 2008). By introducing a robot during the post-workshop exercises, with different abilities to those already encountered, participants with little limited prior knowledge (such as the students discussed in Chapter Seven) had to adapt and use knowledge gained during the workshop to successfully complete a challenge. This is because code from earlier tasks could not be copied due to Page, and related control methods, differing. Details of the programming exercises used during TTW1 and TTW2 are provided in Sections 6.3 and 6.4.

6.2.4 Data Analysis Strategy

In this sub-section the analysis strategy for the pre- and post-workshop questionnaires and the in-workshop log is outlined. The analysis strategy for the programming exercises that were completed is outlined in Sections 6.3 and 6.4 with the results of each study.

Trainee Teacher Questionnaires

Both TTQ1 and TTQ2 have been subject to quantitative and qualitative analysis. This includes the use of descriptive statistics, tabulation, graphics and the categorisation of open questions. It was noted during the exploratory studies how a large amount of data related to subjective opinions. A qualitative analytical approach was considered best suited for the analysis of these. This has led to ‘open’ responses being thematically analysed by determining trends in the replies received and by categorising this information. Quotes have been provided to support identified themes.

Trainee Teacher Workshop Log

The analysis of the workshop logs involved the identification of important events that may have had an impact upon the other data collected. Specific approaches for the analysis of qualitative data, such as thematic coding analysis, were not required as the workshop log data could be analysed in its raw format due to there being a relatively small amount of information. Issues such as problems with participant involvement are presented. A summary of workshop events is provided. Full workshop log transcripts are available in Appendix A6 to ensure transparency.

6.3 Trainee Teacher Workshop One (TTW1)

Further information relating to the programming exercises used during TTW1 is presented in this section. Following this the results of the study are provided.

6.3.1 Data Collection Instruments – Programming Exercises

In-Workshop Programming Exercise One and Two (TTW1-IWE1 & TTW1-IWE2)

Two paper-based in-workshop exercises were administered during TTW1 consisting of 10 multiple-choice questions where one of three potential answers could be selected (although one constructed response question was also used). It was expected that these would take around five minutes to complete. The exercises were devised after considering the programming constructs and learning objectives outlined in Chapter Five. Three PhD CS students, all with Java programming knowledge, were asked to complete these exercises to determine if there were any ambiguous questions or errors. No further validation activities were undertaken as TTW1 offered an opportunity to trial such instruments. Copies of the exercises can be found in Appendix A7.

Post-Workshop Programming Exercises (TTW1-PWE)

Like the in-workshop programming exercises, the post-workshop challenges were also developed after considering the ACM/IEEE Joint Task Force computing curricula guidelines. Completion of these four exercises required around 30 minutes. Code was collected and later graded. A copy of the post-workshop programming exercises (used during TTW1) can be found in Appendix A7.

6.3.2 Data Analysis Strategy – Programming Exercises

In-Workshop Programming Exercise One and Two (TTW1-IWE1 & TTW1-IWE2)

Each in-workshop exercise has a maximum score of ten. Both exercises were initially graded by a sole marker (the author). After a period of six months the same marker re-graded these exercises and the scores previously awarded were examined. This is the test-retest approach (Kottner *et al.*,

2011). There were no differences in marks that were awarded. Use of the test-retest approach was sufficient because the TTW1 workshop was an opportunity to trial the instruments.

Post-Workshop Programming Exercises (TTW1-PWE)

Code collected as a result of the post-workshop programming exercises was graded according to a three-point scale. This scale was developed with novice student programmers in mind rather the trainee teachers. Letter grades have previously been used to rate programming performance (Poindexter, 2003). The three-point marking criteria were as follows:

- A.** Knowledge gained during the workshop has been used to critically solve a new problem. At least 80% of code is correct.
- B.** The new problem has been attempted and solved partially. Between 50% and 80% is correct.
- C.** No or little attempt has been made to solve the new problem. The participant may have simply tried to copy previous code without trying to adapt it. Less than 50% of the code is correct.

This scale was chosen because the criteria are simple to understand and easy to interpret. In addition, the criteria were considered capable of producing manageable analytic data. Three-point marking schemes also share parallels with the ‘traffic-light’ assessment strategy that has been used to consider learner performance (Black, 2004). A sample of data (code collected from six participants) was second-marked by a PhD supervisor (TK) after initially being marked by the author. There was disagreement in several marks. After a discussion, and examination of other participants’ code, marks were agreed. For those without substantial Java experience, analysis of how knowledge progressed during the workshop was possible and separate analysis of data collected from participants with considerable Java experience was undertaken.

6.3.3 TTW1 Trainee Teacher Questionnaire One (TTQ1)

17 participants attended Day One and completed TTQ1.

Past Programming Experience

Participants were asked which programming languages they had used previously and were instructed to select one of three options (Beginner, Competent, Expert) to reflect their self-perceived knowledge of each. All participants had programming experience. The minimum number of languages participants had used was one (four participants) and the maximum number was ten (one participant). The mean number of languages used was 3.6. Table 6-1 displays this information.

<i>Language</i>	Beginner	Competent	Expert	Total Number of Participants With Experience
Java	10	3	0	13
Scratch	4	4	1	9
HTML	5	2	1	8
VB.Net / VB	6	0	0	6
SQL	2	3	0	5
Build Your Own Blocks	2	1	1	4
PhP	2	2	0	4
Javascript	1	1	0	2
Actionscript	0	0	1	1
CSS	0	0	1	1
Logo	0	1	0	1
C++	0	1	0	1
MatLab	1	0	0	1
Alice	1	0	0	1
BBC Basic	0	1	0	1

Table 6-1. Programming languages TTW1 participants had experience using arranged according to participants self-perceived knowledge of each.

As shown in Table 6-2, participants identified the language they most recently learned.

<i>Programming Language</i>	<i>Total Number of Participants</i>
Java	8
Scratch	3
Javascript	1
HTML	1
VB.Net / VB	1
BBC Basic	1
PhP	1
[No Response Received]	1

Table 6-2. Programming language most recently learned by TTW1 participants.

The experience of learning this most recent language was then investigated (Table 6-3).

<i>“How did you learn this language?”</i>					
	<i>Self-Taught</i>	<i>Part of a course or education program</i>		<i>Specify own response</i>	
Number of Responses	3	13		1 ¹	
<i>“How much time did you spend learning this language?”</i>					
	<i>Less Than One Week</i>	<i>One Week to Two Months</i>	<i>Two Months to Four Months</i>	<i>Over Four Months</i>	<i>Unable to Determine</i>
Number of Responses	2	5	7	1	2
<i>“How well do you feel you learnt this language?”</i>					
	<i>Learnt very little</i>	<i>Became familiar with most concepts</i>	<i>Became competent</i>	<i>Became expert</i>	
Number of Responses	2	10	5	0	
<i>“Which of the following best describes your past programming experience?”</i>					
	<i>Didn’t Like</i>	<i>Indifferent</i>		<i>Enjoyed</i>	
Number of Responses	5	7		5	

Table 6-3. TTW1 participants' experience of learning their most recent programming language.

To determine familiarity with programming fundamentals, instructions were provided to select which concepts had been used in previous code. This data can be seen in Table 6-4.

¹ The one open response received specified “Code Academy”

<i>Programming Concept</i>	<i>Total Number of Participants</i>
Arrays	16
Variables	16
Iteration (e.g. while loops, for-loops)	15
Selection (e.g. if, if...else)	15
Strings	14
Object-oriented programming	13
Methods/Functions/Subroutines	11
File Input/output	10
Graphical User Interfaces (GUIs)	5
Recursion	5
Expressions	2
Multithreading	1

Table 6-4. Programming concepts arranged by the number of TTW1 participants who stated that they had used each concept in their past code.

Participants' Views on Programming

TTQ1 also focused on views on programming. Open-ended questions were used to determine issues associated with learning to program. Four categories were established as shown in Table 6-5.

<i>General Categories Identified</i>	<i>Participant Responses</i>
<i>Issues related to views of programming</i>	<p>“Mathematical/logical – if not good at maths then not good at programming”</p> <p>“Competent maths. Dyslexia learning difficulty”</p> <p>“The need to have good logical thinking”</p> <p>“Making it enjoyable and interesting (and) not the stereotypical view of geeky”</p>
<i>Issues with the nature of programming</i>	<p>“Difficult to understand”</p> <p>“Getting to grips with the language can sometimes be difficult”</p> <p>“Complicated”</p> <p>“Attention to detail”</p> <p>“Using correct syntax”</p> <p>“Sometimes not understanding the concepts”</p>
<i>Issues learning to program</i>	<p>“Making it fun. Understanding (every) concept”</p> <p>“Can be difficult to learn initial concepts. Can be difficult to see results”</p>
<i>Miscellaneous issues</i>	<p>“People who already know stuff – a teacher might aim at their level and ignore the others who are just beginning”</p>

Table 6-5. Issues identified by TTW1 participants in regards to the learning of programming.

Participants were then asked what stereotypes they associated with learning to program. Two categories were established after considering responses as shown in Table 6-6. As shown in Table 6-7, participants' opinions on teaching high school students programming were also determined.

<i>General Categories Identified</i>	<i>Responses</i>
<i>The perception of programmers/programming</i>	"Geeky" "Geeks. Swots" "Geeky" "Geeky. Nerdy" "Geeky. Clever" "Nerd. Boring person. Intelligent" "It is for nerds. Got to be very clever to do it. Boring" "That it is difficult, is for nerds and has maths involved" "Highly technical. For geeks" "Have to understand the basic concepts"
<i>Issues with gender</i>	"Male. Anti-social" "Boys will know more about programming and will get ahead"

Table 6-6. Stereotypes identified by TTW1 participants in regards to the learning of programming.

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>			
	Yes	Not Sure	No
Number of Responses	15	1	1

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>			
	Yes	Not Sure	No
Number of Responses	11	4	2

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	Difficult	Neither difficult nor easy	Easy	Not sure
Number of Responses	5	9	1	2

Table 6-7. TTW1 participants' opinions on teaching high school students programming (pre-TTW1).

6.3.4 TTW1 Trainee Teacher Questionnaire Two (TTQ2)

TTQ2 was completed after the TTW1 workshop. 16 participants attended Day Two of the workshop and completed TTQ1. One participant who had attended Day One was absent.

Participants' Workshop Experience and Opinions of the Kebot Robot Simulator

Participants were asked about their workshop enjoyment, the programming tasks set and their views on the effectiveness of Kebot as tool to introduce programming (Table 6-8).

<i>“In regards to your programming experience during the workshop, which of the following best describes your enjoyment?”</i>			
	<i>Enjoyable</i>	<i>Indifferent</i>	<i>Not Enjoyable</i>
Number of Responses	12	3	1
<i>“In regards to the workshop session, which of the following best describes how difficult the programming tasks have been?”</i>			
	<i>Easy</i>	<i>Neither difficult nor easy</i>	<i>Difficult</i>
Number of Responses	2	10	4
<i>“Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	15	0	1
<i>“Would you consider using the robot simulator as a tool to teach programming in your own lessons in the future?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	15	1	0

Table 6-8. TTW1 participants' opinions of their workshop experience and effectiveness of the Kebot simulator.

Two open-ended questions were used to determine aspects of Kebot that were liked and not liked. Instructions were given so up to three aspects for each were specified. Information related to elements of the simulator that were liked is displayed in Table 6-9 while information related to elements disliked is displayed in Table 6-10. General categories have been established after considering responses. No data provided by participants has been omitted.

General Categories Identified	Participant Responses
<i>The visual and interactive nature of simulated robots</i>	<p>“Kebot is easy to use and it is good to see how the code works. Makes it interesting”</p> <p>“Visual display... actually see what the code is doing!”</p> <p>“The ability to get visual results even when looking at simple beginners code”</p> <p>“Looks simple but can program more complex things. Trajectories are good so you can see what your program is doing”</p> <p>“Ease of use. Good use of visual reference for testing”</p> <p>“Visual results. Realistic tasks. (Can) relate to real-life (e.g. Lego Mindstorms etc)”</p> <p>“The moving robot. The maze. The fact that code was already there, you only had to change small parts”</p>
<i>The accessible nature of Kebot</i>	<p>“Set out and ready to go”</p> <p>“Fun. Enjoyable. Easy to understand for beginners”</p> <p>“Hands-on approach. Tasks started easy (and became) more difficult tasks. Simple GUI”</p> <p>“Simplicity, friendly, graphical. Easy to recover when wrong”</p> <p>“Ease of use. Understandable. If...else used quickly and explained well (using the simulator)”</p>
<i>Miscellaneous comments</i>	<p>“Well-structured workshop. Interesting”</p> <p>“Use of appropriate commands (at the top of each robot class) to remind the user what methods they can use”</p> <p>“Interesting topic. Made programming engaging”</p> <p>“Initial part, getting the robot to move”</p>

Table 6-9. Aspects of Kebot liked by TTW1 participants.

General Categories Identified	Participant Responses
<i>Issues with workshop difficulty</i>	<p>“Not differentiated. Difficult. Pace too much”</p> <p>“Gets complicated too quickly”</p> <p>“Progressively difficult. Not enough time for activities”</p> <p>“Can become quite complicated”</p> <p>“The <code>i = 0 < i++</code> etc. Arrays were too difficult for me. The Kebot program does not help with syntax errors”</p>
<i>The visual appearance of Kebot or the interface design</i>	<p>“Not very visually appealing”</p> <p>“Some right-click options not used. Pupils could be distracted”</p> <p>“Can only use one robot at a time!”</p>
<i>Miscellaneous comments</i>	<p>“Could add a note to each robot to specify what is different about each one (e.g. that a particular robot has been designed to execute in a loop) just as a reminder for students”</p>

Table 6-10. Aspects of Kebot disliked by TTW1 participants.

To establish how effective participants perceived elements of the workshop to have been, they were instructed to score on a scale of one (not at all effective) to five (extremely effective) their views on the: Kobot simulator, programming support received, presentation; teaching environment. Responses of 15 participants can be seen in Table 6-11. One participant did not respond.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		1	2	3	4	5
Kobot Simulator	4.4	0	1	1	4	9
Programming Support	4.0	1	1	2	4	7
Workshop Presentation	4.1	0	2	2	3	8
Environment	4.5	0	0	2	4	9

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 6-11. TTW1 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.

As shown in Table 6-12, participants were asked to compare their previous introductory programming learning experience to the one using Kobot. Seven participants completed the workshop whose most recent programming learning experience had been in Java. Table 6-13 further breaks down the information presented in Table 6-12 and responses of these participants alone are displayed.

<i>"If you have previously learnt a programming language, was your previous introduction to basic programming..."</i>					
	<i>Much less effective</i>	<i>Less effective</i>	<i>About the same effectiveness</i>	<i>More effective</i>	<i>Much more effective</i>
Number of Responses	4	5	6	1	0

Table 6-12. Comparison of TTW1 participants' previous introductory programming experience to the one using Kobot.

<i>"If you have previously learnt a programming language, was your previous introduction to basic programming..."</i>					
	<i>Much less effective</i>	<i>Less effective</i>	<i>About the same effectiveness</i>	<i>More effective</i>	<i>Much more effective</i>
Number of Responses	3	1	2	1	0

Table 6-13. Comparison of TTW1 participants' previous introductory programming experience to the one using Kobot (participants who most recently learned Java only).

TTQ2 included questions also used on the TTQ1 questionnaire. These related to the importance of teaching programming and participants' confidence/difficulty teaching programming (see Table 6-14). When comparing any changes from TTQ1 responses note that one less participant took part in Day Two of TTW1 and, as result, TTQ2 was completed by one less respondent.

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>				
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>	
Number of Responses	11	3	2	
Change from TTQ1 Responses	-4	+2	+1	

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>				
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>	
Number of Responses	8	9	1	
Change from TTQ1 Responses	-3	+5	-1	

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Number of Responses	5	8	1	2
Change from TTQ1 Responses	0	-1	0	0

Table 6-14. TTW1 participants' opinions on teaching high school students programming (post-TTW1).

TTQ2 concluded with an opportunity to offer general comments about the robot simulator and the workshop session. Responses were almost entirely positive and included praise of the workshop approach, the nature of the workshop or the manner in which material had been delivered:

“Excellent atmosphere in the classroom. Enjoyed the two-day workshop”

“Very well delivered”

“Really good – I am actually happy I learnt something!! The session pace was good”

“Thank you for a fantastic workshop where I learnt at my own pace”

“I would like to have a copy of Kebot as I believe it is an effective way of teaching Java

“Was really good – needed extension activities”

“Very informative and enjoyable session. [Workshop leader] has shown excellent subject knowledge. Hand-outs proved extremely helpful and will be useful for future reference”

One participant, however, offered comment that perhaps the approach would not help all novice programmers to learn:

“The program is expected to take eight hours to deliver to high school students – unrealistic – very complicated concepts and the pupils I am used to teaching struggled with queries in databases. Programming won’t suit all pupils”

6.3.5 Results - TTW1 Workshop Log (Day One Overview)

No issues were documented during Day One of TTW1 in regards to the research environment or technical failure. Three participants arrived late for the session (no more than 30 minutes) although no programming activities were missed due to the early part of the workshop being dedicated to discussion and completion of consent forms. Participants remained ‘on-task’ during the session and appeared eager to engage with material and to help one another when required. Four participants fell behind at points during the session and extra time was spent discussing aspects of programming with these. This was typically when a programming concept, or aspect of Kobot, was introduced for the first time. General syntax errors appeared to cause most participants a problem at some stage. Kobot appeared to offer an engaging means of introducing programming concepts. Little time was spent diverging from material and participants remained focused. Due to the number of participants in the group occasionally some had to wait several minutes for an answer to a particular question. This is because several participants required help at the same time. At the end of Day One TTW1-IWE1 was distributed before the session was concluded. TTW1-IWE1 was completed by most participants in one or two minutes, much quicker than was anticipated. Several comments were made in regards to issues with the presentation of TTW1-IWE1, specifically problems distinguishing multiple-choice responses from some questions. A complete copy of the Workshop Log for TTW1 Day One is available in Appendix A6.

6.3.6 Results - TTW1 Workshop Log (Day Two Overview)

As with Day One of TTW1, no issues were encountered in regards to the research environment or technical failure. Two participants arrived late for the session while one participant, who had attended Day One, was absent entirely (for reasons unknown). One participant was also only able to partially complete the programming tasks undertaken during the session as a result of a wrist injury. The programming challenges completed during Day Two were intentionally more complex, and less prescriptive, than those undertaken on the first day. It was intended that this would allow for greater experimentation and this was observed during the workshop. The large class size again resulted in several participants requiring assistance at the same time and this led to the delivery of programming material being delayed at points. Participants remained enthusiastic throughout even during times of difficulty that had the potential to frustrate. Kobot performed well and no software issues were encountered. Several participants were observed, at several points, attempting to elicit additional behaviour from their robotic agents. This was particularly the case when these participants were waiting for their colleagues to complete the tasks already set. In terms of concepts that caused issues, arrays and for-loops were in particular troublesome. The nature of both sessions was praised verbally and all participants appeared to be enthused by the Kobot simulator and its approach to introducing Java. Several participants made requests for a copy of the software and the workshop slides for future use. Issues with the programming assessments used were, however, noted. Like TTW1-IWE1, TTW1-IWE2 was completed rapidly by the majority of participants. TTW1-PWE also appeared to be hastily completed by some. Around 25 minutes after TTW1-PWE had been distributed several participants submitted their responses and left the laboratory. This caused disruption and a number of remaining participants appeared to lose concentration before rapidly attempting their remaining tasks. Several questions were also asked in regards to Task Two of TTW1-PWE and the wording of the question, and nature of the task, caused significant confusion. Many participants did not submit code in response to this task due to this issue, or submitted code considered as incorrect. A complete copy of the Workshop Log for TTW1 Day Two is available in Appendix A6.

6.3.7 Analysis of TTW1 Participants' Programming Performance

As the main purpose of implementing the programming exercises during TTW1 was to not to assess learning, but to allow the testing and validation of programming exercises in advance of studies involving students (reported in Chapter Seven), participant performance has not been detailed in the main body of this thesis. Lessons learned are, however, presented below. Consideration of participants' programming performance has been undertaken independently of the overall analysis because the participants were not novices and as the exercises were modified after TTW1. For completeness information relating to the performance of TTW1 participants is presented in Appendix A8.

As the instruments used during TTW1 were later modified due to issues with design and content, collected information cannot be used to draw strong conclusions. This discussion is intentionally brief as a result. Despite being assumed that participants would have little difficulty completing the in- or post-workshop exercises due to their prior programming experience, this turned out not to be the case. TTW1-IWE1 caused few problems although two questions (Q6 and Q7) were incorrectly answered by over half of participants. Indeed, only one participant who had not most recently learned Java answered Q7 correctly. Otherwise, the mean scores between participants who had/had not recently learned Java are considered similar. On TTW1-IWE2 a lower collective mean score was reported compared to TTW1-IWE1. This was anticipated considering that the difficulty of concepts and tasks increased as the workshop progressed. The mean scores of participants, both with and without recent Java experience, were identical. In regards to TTW1-PWE performance on Task Two (in conjunction with observations reported in the TTW1 Workshop Log) demonstrates a notable problem occurred and that the task required substantial modification before later workshops. As also reported in the Workshop Log, Task Four was also completed hastily by some participants and this may have impacted upon grades awarded. Statistical analysis techniques have not been used to analyse data due to issues associated with the design of the TTW1 exercises and as all participants had different levels of prior programming experience.

Whilst firm conclusions cannot be drawn from participants' performance on the programming exercises, overall performance did not match expectations. Several factors may have been responsible for the performance observed:

1. The size of the group – 17 participants took part in TTW1. This is considered to be a large number especially for a workshop leader who, at the time, did not have substantial experience leading such sessions. At various points the workshop needed to be delayed as the same programming concept was explained to multiple participants. It is possible some participants did not receive the help they required or did not request help because of this.
2. Problems with the programming exercises – As already discussed all exercises were modified, for various reasons, in advance of TTW2 and the workshops involving students (Chapter Seven). It is reasonable to assume that problems identified with the design of these exercises had a detrimental impact upon performance.
3. The programming backgrounds of participants – As established during discussions with participants prior to the workshop, it was known that TTW1 participants had programming experience. However, after considering responses to the pre-workshop questionnaire, for some participants this past exposure to programming was not as substantial as originally thought.

6.4 Trainee Teacher Workshop Two (TTW2)

In this section the results of the second trainee teachers' workshop (TTW2) are presented. As discussed below and in Section 6.5, all programming exercises were modified after their use during TTW1. The procedures and instruments for the pre-workshop questionnaire, the post-workshop questionnaire and the researcher's workshop log remained the same as for TTW1.

6.4.1 Data Collection Instruments – Programming Exercises

TTW1 allowed an opportunity to trial the programming exercises in advance of TTW2 and the workshops involving students aged 16 to 18 (Chapter Seven). Like TTW1 participants, all of those involved in TTW2 had prior programming experience. This allowed an opportunity to validate the in- and post-workshop programming exercises used during the workshops involving students.

In-Workshop Programming Exercise One and Two (IWE1 & IWE2)

Two paper-based exercises were distributed during TTW2. Lessons learned as a result of administering the in-workshop programming exercises during TTW1 influenced the design of these instruments. Most questions were newly created. Compared to the TTW1 exercises, the updated exercises used during TTW2 and the student programming workshops were considered to offer:

- Improved presentation – a larger font was used and the layout made clearer
- Greater link to Kebot – as questions were modified so they were scenario-based, it was considered this would promote problem solving and make the exercise more relevant.
- A more rigorous test – due to greater use of constructed response questions. Where multiple-choice questions were still used, potential responses were increased from three to four.

A PhD supervisor (TK), who has introductory programming teaching experience, reviewed the exercises several times before suggesting improvements. Multiple-choice exam papers, used at the end of a first-year Java programming course at Keele University, were also consulted to gain an understanding of how best to structure multiple-choice questions. The programming constructs, and

learning objectives, outlined by the ACM/IEEE Joint Task Force again dictated content. Copies of the in-workshop programming exercises used during TTW2 can be found in Appendix A9.

Post-Workshop Programming Exercises (PWE)

A programming exercise, consisting of four programming tasks, was again administered after the workshop. This required around 35 minutes for completion. As during TTW1, such exercises were based around the Page robotic agent. Whilst three of the four post-workshop tasks shared similarities with those used during TTW1, the exercises differed. This is because the details of, and instructions relating to, these tasks were revised after considering feedback provided during TTW1. One task was completely new. The PhD supervisor (TK) reviewed these instruments also. A copy of the post-workshop programming exercises used during TTW2 can be found in Appendix A9.

6.4.2 Data Analysis Strategy – Programming Exercises

In-Workshop Programming Exercise One and Two (IWE1 & IWE2)

Each in-workshop exercise has a maximum score of ten and all questions are marked either correct or incorrect. All of the in-workshop exercises were second marked by a PhD supervisor (PB) to ensure consistency in the marking process. There were no differences in marks that were awarded.

Post-Workshop Programming Exercises (PWE)

Code collected as a result of the post-workshop programming exercises has been graded according to the same three-point scale used during TTW1. This was because the marking criteria were considered simple to understand, easy to interpret in addition to being capable of producing manageable analytic data. Due to the small sample size all of the collected data was second-marked by a PhD supervisor (TK) after initially being marked by the author. There was no disagreement in the marking of this code.

6.4.3 Results – TTW2 Trainee Teacher Questionnaire One (TTQ1)

TTQ1 was completed before the workshop. Five participants attended Day One of the workshop.

Participants' Past Programming Experience

Programming languages previously used by participants involved in TTW2 are displayed in Table 6-15 along with their self-identified knowledge of each. All participants had programming experience. The minimum number of languages participants had used was two (one participant) and the maximum number was four (three participants). The mean number of languages used was 3.4.

<i>Participant</i>	<i>Language(s)</i>	Beginner	Competent	Expert
TTW2-P1	C#	✓		
	Java	✓		
TTW2-P2	Java	✓		
	Basic		✓	
	Delphi		✓	
	Pascal		✓	
TTW2-P3	JavaScript		✓	
	C#			✓
	HTML			✓
	Visual Basic			✓
TTW2-P4	ASP.Net		✓	
	RPG/400		✓	
	SQL		✓	
	VB Script		✓	
TTW2-P5	JavaScript	✓		
	SQL	✓		
	Visual Basic		✓	

Table 6-15. Programming languages TTW2 participants had experience using along with their self-identified knowledge of each.

Participants were asked to consider their most recent programming learning experience and to state: language learned, the method by which they learned, time spent learning and how well they felt they learnt. Data collected is presented in Table 6-16. To determine familiarity with programming fundamentals, participants were instructed to select which concepts they had used in their previous code (see Table 6-17). Of the 12 programming concepts participants were asked to select, the mean number of concepts previously used was 10.

<i>Participant</i>	Language most recently learned	How did you learn this language? <i>Self-Taught / Part of a course or education program / Specify own response</i>	How much time did you spend learning this language? <i>Less than one week / One week to two months / Two to four months / Over four months</i>	How well do you feel you learnt this language? <i>Learnt very little / Became familiar with most concepts / Became competent / Became expert</i>
TTW2-P1	C#	Self-Taught	<i>Unable to Determine</i>	<i>Unable to Determine</i>
TTW2-P2	Java	Part of a course or education program	Two to Four Months	Became familiar with most programming concepts
TTW2-P3	C#	Self-Taught	Less than one week	Became expert
TTW2-P4	ASP.Net	Part of a course or education program	Less than one week	Learnt very little
TTW2-P5	JavaScript	Part of a course or education program	Two to Four Months	Became familiar with most programming concepts

Table 6-16. Information relating to TTW2 participants' most recent programming learning experience.

<i>Programming Concept</i>	Total Number of Participants
Arrays	5
Variables	5
Iteration (e.g. while loops, for-loops)	5
Strings	5
File Input/output	5
Selection (e.g. if, if...else)	4
Object-oriented programming	4
Methods/Functions/Subroutines	4
Graphical User Interfaces (GUIs)	4
Recursion	4
Expressions	4
Multithreading	1

Table 6-17. Programming concepts arranged by the number of TTW2 participants who stated that they had used each concept in their past code.

Participants were then asked to describe their past enjoyment of programming (Table 6-18).

<i>“Which of the following best describes your past programming experience?”</i>			
	<i>Didn’t Like</i>	<i>Indifferent</i>	<i>Enjoyed</i>
Number of Responses	0	1	4

Table 6-18. TTW2 participants' enjoyment of their past programming experience.

Participants’ Views on Programming

Two participants responded to open-questions associated with issues learning to program:

“Understanding the theory and picking a theory to follow - e.g. object-orientation – before trying to code anything... The overview and the why before the specific and the how”

“People learn at different rates and this can be a problem. Practical teaching is easier to understand rather than just theory (such as lectures)”

The following stereotypes were also noted by these participants:

“More females should study programming”

“Tends towards boys who aren’t sporty”

In Table 6-19 participants opinions on teaching high school students programming are shown.

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	5	0	0

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	2	3	0

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Number of Responses	0	3	0	2

Table 6-19. TTW2 participants’ opinions on teaching high school students programming (pre-TTW2).

6.4.4 Results – TTW2 Trainee Teacher Questionnaire Two (TTQ2)

TTQ2 was completed after the workshop. All five participants who attended Day One of the workshop attended Day Two and completed TTQ2.

Participants' Workshop Experience and Opinions of the Kebot Robot Simulator

Participants were asked about their workshop enjoyment, the programming tasks set and their views on the effectiveness of Kebot as tool to introduce programming (see Table 6-20).

<i>“In regards to your programming experience during the workshop, which of the following best describes your enjoyment?”</i>			
	<i>Enjoyable</i>	<i>Indifferent</i>	<i>Not Enjoyable</i>
Number of Responses	5	0	0
<i>“In regards to the workshop session, which of the following best describes how difficult the programming tasks have been?” (*note only four responses received)</i>			
	<i>Easy</i>	<i>Neither difficult nor easy</i>	<i>Difficult</i>
Number of Responses	2	2	0
<i>“Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	4	1	0
<i>“Would you consider using the robot simulator as a tool to teach programming in your own lessons in the future?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	5	0	0

Table 6-20. TTW2 participants' opinions of their workshop experience and effectiveness of the Kebot simulator.

Two open-ended questions were used to establish what aspects of Kebot were liked and not liked. Instructions were provided so up to three aspects were specified for each. All data collected as a result of these activities is displayed in Table 6-21. Between them TTW2 participants identified 14 aspects of Kebot that they liked and seven aspects that they did not.

<i>Participant</i>	<i>Aspects of Kebot liked</i>	<i>Aspects of Kebot disliked</i>
TTW2-P1	<i>“Relates to ‘real-life’ scenarios”</i> <i>“Learn as you go – both theory and practice”</i> <i>“Broken down into easy understandable chunks”</i>	<i>“n/a - I enjoyed it all!”</i>
TTW2-P2	<i>“Visual expression of code”</i> <i>“Direct application”</i> <i>“Immediate response”</i>	<i>“Does not help when code is wrong”</i> <i>“Instruction codes are not fully explained before application”</i> <i>“No library for checking instruction definitions”</i>
TTW2-P3	<i>“Aesthetically pleasing”</i> <i>“Simple interface”</i> <i>“Logo style commands mean learners are concentrating on the structure rather than the minutiae of coding”</i>	<i>“Imprecision in the movement”</i> <i>“Reloading is a pain (although I appreciate this is a limitation of the language)”</i>
TTW2-P4	<i>“Easy-to-use”</i> <i>“Interactive”</i> <i>“Motivational”</i>	<i>“Only had 2D and 3D drawing tools - simplistic”</i>
TTW2-P5	<i>“Appealing and intriguing”</i> <i>“User friendly in terms of design and structure”</i>	<i>“Sometimes having to re-draw graphics if one forgets to close the program correctly”</i>

Table 6-21. Aspects of Kebot liked and disliked by TTW2 participants.

To establish how effective participants believed elements of the workshop to have been, they were instructed to score on a scale of one (not at all effective) to five (extremely effective) their views on the: Kebot simulator, programming support received, presentation delivered; teaching environment. Responses can be seen in Table 6-22.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		1	2	3	4	5
Kebot Simulator	4.2	0	0	1	2	2
Programming Support	4.0	1	0	0	1	3
Workshop Presentation	4.2	0	0	1	2	2
Environment	4.8	0	0	0	1	4

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 6-22. TTW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.

As documented in Table 6-23, participants were asked to compare their previous introductory programming learning experience to the one using Kebot.

<i>“If you have previously learnt a programming language, was your previous introduction to basic programming...”</i>					
	<i>Much less effective</i>	<i>Less effective</i>	<i>About the same effectiveness</i>	<i>More effective</i>	<i>Much more effective</i>
Number of Responses	1	1	1	2	0

Table 6-23. Comparison of TTW2 participants’ previous programming experience to the one using Kebot.

TTQ2 included questions also used on the TTQ1 questionnaire. These related to the importance of teaching programming and confidence/difficulty teaching programming (see Table 6-24).

<i>“Do you believe that it is important to teach basic programming concepts to all high school students enrolled on an ICT or computing course at some stage during their time in school?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	5	0	0
Change from TTQ1 Responses	0	0	0

<i>“With your current knowledge would you be confident teaching high school students about introductory programming concepts?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	3	2	0
Change from TTQ1 Responses	+1	-1	0

<i>“How difficult do you think it would be to teach elementary programming concepts to high school students using your current knowledge?”</i>				
	<i>Difficult</i>	<i>Neither difficult nor easy</i>	<i>Easy</i>	<i>Not sure</i>
Number of Responses	1	2	2	0
Change from TTQ1 Responses	+1	-1	+2	-2

Table 6-24. TTW2 participants’ opinions on teaching high school students programming (post-TTW2).

Finally, an opportunity was given for general comments. Two were received:

“Brilliant – The better way to teach programming at all learning levels”

“Be nice (although I don’t know how practical) to use the written program to control a real robot. Once it’s been written and tested it would be nice to something physical doing what you told it to do. I’d like a copy of the code so I can try getting a robot through a maze”

6.4.5 Results – TTW2 Workshop Log (Day One Overview)

No issues were documented during Day One of TTW2 in regards to the research environment or technical failure. All five participants arrived on time. Participants remained ‘on-task’ during the session and were eager to participate and use Kobot. The group seemed knowledgeable about programming and it was evident that all had coding experience. All participants completed the programming tasks set and asked questions. The small group size meant that the session progressed well as the same concept did not need to be explained to multiple participants. IWE1 was distributed before the session was concluded. No concerns were raised by participants in regards to this instrument and no problems with it were apparent. A complete copy of the Workshop Log for TTW2 Day One is available in Appendix A6.

6.4.6 Results – TTW2 Workshop Log (Day Two Overview)

As with Day One of TTW2, no issues were encountered in regards to the research environment. All participants attended and were on-time. As with the first day participants were enthusiastic. Greater assistance needed to be offered, due to the increased complexity of the tasks, but participants seemed to enjoy the challenge. Few concepts caused consistent difficulty and participants appeared to make use of their prior knowledge and experience to solve tasks. No issues were noted in regards to assessments used. Participants were extremely eager to inquire about the approach taken during the workshop. Copies of all materials used were requested by the group. No issues were noted with either of the two programming assessments (IWE2 and PWE). On the PWE some tasks participants appeared to be coding robotic behaviour beyond that which was requested. A complete copy of the Workshop Log for TTW2 Day Two is available in Appendix A6.

6.4.7 Analysis of TTW2 Participants' Programming Performance

Similarly to TTW1, the programming performance of TTW2 participants is presented in an appendix (Appendix A10). This is because the main purpose of using the programming exercises during the trainee teacher workshops was to not to assess learning, but to allow the testing and validation of programming exercises used during the studies involving students (reported in Chapter Seven). Interpretation of the results of the programming exercises is presented below.

The programming exercises used during TTW2 were the same as those for the workshops involving students. Whilst the exercises implemented during TTW1 and TTW2 cannot be compared directly, the performance of TTW2 participants is believed to have been better. Despite modifications being made to all exercises post-TTW1, such changes are not considered to have made the tests easier. On the contrary, for the reasons outlined earlier, the assessments used during TTW2 and the student workshops are considered to be more rigorous. The size of the group must be taken into account when considering performance as participants may have received greater assistance from the workshop leader compared to TTW1. As fewer trainees were enrolled during the academic year when TTW2 took place (*n.* 5), fewer participants took part. TTW2 trainees:

- Performed well on IWE1. Q6 and Q7 again caused issues for some participants as with TTW1-IWE1. As Q6 and Q7 on both TTW1-IWE1 and IWE1 are completely different, and share no relationship in terms of concepts being assessed, this is considered to be a coincidence.
- Scored lower overall on IWE2, compared to IWE1, as was expected. The performance of two participants, however, was noticeably poorer compared to their peers (three of whom scored nine). Due to the small sample involved, it is not possible to determine whether this was due to the design of particular questions (as both got the same four questions wrong) or other factors.
- Were all awarded Grade A or B on each of the PWE tasks. Due to participants' past programming experience this is not considered to demonstrate that the exercises were disproportionately easy. It was noted, however, that similar scores during the students' workshop may indicate that the PWE was not capable of determining performance.

6.5 Lessons Learned

In this section a discussion relating to the conduct of Case One of the case study is presented. Results reported in this chapter are used to address the aims of the research, and several research propositions, elsewhere (Chapter Eight). This case involved two cohorts of trainee teachers and the work reported in this chapter had the following aims:

- 1) To compare participants' prior programming learning experience to the one using Kobot
- 2) To establish opinions on the effectiveness of the Kobot robot simulator
- 3) To determine general opinions of programming

Case One allowed an opportunity to test and validate programming exercises (two in- and one post-workshop) designed to determine the knowledge of novice student programmers. Experience was gained implementing such exercises for research purposes and issues with these were identified in advance of later workshops. The programming exercises were not used during TTW1 and TTW2 to assess learning although it was acknowledged that interesting data might be collected. It was assumed before the workshops that the programming exercises would offer little challenge to participants with prior programming experience.

The programming exercises used during TTW1 and TTW2 were different. In the case of the in-workshop exercises these differed due to:

- Feedback provided, and observations noted, during TTW1 establishing the need for modification to the format of the instruments
- There being little link with the Kobot simulator or robotics and, with hindsight, these appeared detached from the workshop content
- These being completed extremely quickly during TTW1 (even considering the past programming experience of participants) which suggested the tests needed to be made more substantial

In the case of the post-workshop exercises these differed due to:

- One of the TTW1-PWE tasks completed (Task Two) noticeably causing significant issues and confusion during TTW1
- It being recognised that modification was required to instructions provided

Following the completion of TTW1, and in conjunction with a PhD supervisor (TK), changes were made to all of the programming exercises in advance of other workshops. The performance of TTW1 participants on these exercises was not taken into account and did not impact upon the decision to modify the instruments used (with the exception of Task Two on TTW1-PWE). During this process an evaluation of existing methods currently used at Keele University, to assess learners' programming performance, was undertaken. This included a review of past examinations. The reliability and validity of the revised programming exercises (IWE1, IWE2, PWE) has been considered in the discussion presented in Chapter Eight.

6.6 Summary

Details of the execution and results of one component of the case study, Case One – ‘Trainee Teachers’, have been presented in this chapter. In total, 22 trainee ICT/CS teachers took part. All had learned programming concepts in some capacity previously. Important qualitative data has been collected and reported. Through the use of pre- and post-workshop questionnaires, and in-workshop observations, this data has been used to address the case study aims and propositions in Chapter Eight. Collection of this information allowed an opportunity to compare trainees’ prior programming learning experience to the one using Kebot. In addition, it was possible to establish their views on the effectiveness of the method. Several programming exercises have also been used as a means of testing before, and validation following, the workshops involving students. The results generated due to these exercises are presented in an appendix. This is because use of the programming exercises during the workshops reported in this chapter was to not to assess learning. In the following chapter details and results of the second part of the case study, which involved students, are presented.

Chapter Seven

Case Two: Students

In this chapter details of the second case, Case Two – ‘Students’, are provided. Two workshops were completed as part of this case based on the methodology outlined in Chapter Five. In total, 23 students aged 16 to 18 years old took some part. All were enrolled on a Further Education (FE) course. Data was collected using questionnaires and programming exercises. Statistical analysis was undertaken to analyse the programming performance of the two student cohorts. In-workshop observations were also made. An additional data source was used as semi-structured interviews, with three in-service teachers, took place. Each of these teachers was familiar with one of the student groups involved. Prior to results being presented details of the participants involved, the workshop setting and the data collection and analysis strategies are given.

7.1 Introduction

This chapter continues to build on a case study design devised to achieve the research objectives of this thesis. Details of a workshop, developed to determine the effectiveness of the Kebot robot simulator as a tool to support the learning of introductory programming concepts, have been given and four research propositions were previously outlined. In the discussion in Chapter Eight, results reported in this (and the previous) chapter are used to address these propositions.

It has been established that the case study is a two-case case study. Details of the first case, involving trainee teachers, have been provided. This chapter describes the second case which involved students aged 16 to 18 years old enrolled on a Further Education (FE) course. High-level information related to the research design (such as workshop content) was given in Chapter Five. Information relating to the execution of this case, however, was not. As a result, in Section 7.2, details of the following are outlined: participants involved, workshop procedures and setting, data collection instruments used and analysis strategy adopted.

By involving FE students it was believed that the results generated would be generalisable to students in the later years of high school (aged between 15 and 16 years old) in addition to students in the early stage of Higher Education (HE) and aged around 18 years. This is because FE students are typically aged between 16 and 18 years old and their programming ability (and opinions of the subject) can, therefore, be reasonably likened with these two groups. There were also practical reasons why FE students were selected over other potential student participants. Whilst FE institutions must adhere to regulations that govern what they teach, during discussions with FE teaching staff and managers it became apparent that FE education providers could afford greater flexibility in regards to what they select for their students to do. Involving FE students over other potential participants was also considered to be a risk averse strategy and one that was most likely to succeed. This is because the involvement of high school students could not be guaranteed, as schools (advanced negotiations were held with three of these) were unwilling to commit their pupils for a workshop spanning over 10 hours. Similarly, neither was it considered practical to

involve Keele University CS students in the project. This was due to a pre-defined course syllabus already being agreed in advance of workshop content being finalised, and as the University's teaching team did not believe it was possible to incorporate the Kebot workshop into an already intensive first-year course.

7.2 Study Execution

In this section information relating to the execution of Case Two of the case study is presented.

7.2.1 Participants

Two groups of students took part and two separate workshops were held:

Student Workshop One (SW1) – 12 participants (seven males, five females) enrolled on an Information and Communication Technology (ICT) FE course. One participant was unable to attend the first day while another was unable to attend the second day. This meant that 10 students completed the full workshop. The results of SW1 are presented in Section 6.3.

Student Workshop Two (SW2) – 11 participants (all male) enrolled on a Computing FE course. One participant was unable to attend the first day while two were unable to attend the second. This meant that eight students completed the full workshop. The results of SW2 are presented in Section 6.4.

In total 23 participants, aged between 16 and 18 years old, took some part in a workshop. 18 participants attended a workshop in full. SW1 and SW2 participants were enrolled at different FE colleges.

An additional data source has also been used:

Teacher Interviews (TI) – Three in-service teachers were interviewed. All were familiar with one of the student groups involved (SW1 or SW2).

7.2.2 Workshop Procedure and Setting

Student Workshop One (SW1) – SW1 took place in early-July 2012. Day One of the workshop was held in the participants' own education institute while Day Two was hosted in a computer lab at Keele University.

Student Workshop Two (SW2) – SW2 took place in mid-July 2012. Both days of the workshop took place in the participants' own educational institution.

Both workshops followed the workshop procedure outlined in Chapter Five. The computer labs used were assessed for their suitability by the workshop leader several weeks in advance and the Kobot simulator was tested on installed machines. Upon entering the laboratories, participants were assigned to an individual PC, read an information sheet and completed a consent form. Each was given a code number. No deviations from the workshop procedure occurred.

7.2.3 Data Collection Strategy

Several data sources were used during Case Two. The procedures and instruments for the collection of data were the same for both SW1 and SW2. The instruments used (shown in Figure 7-1) are described in this sub-section.

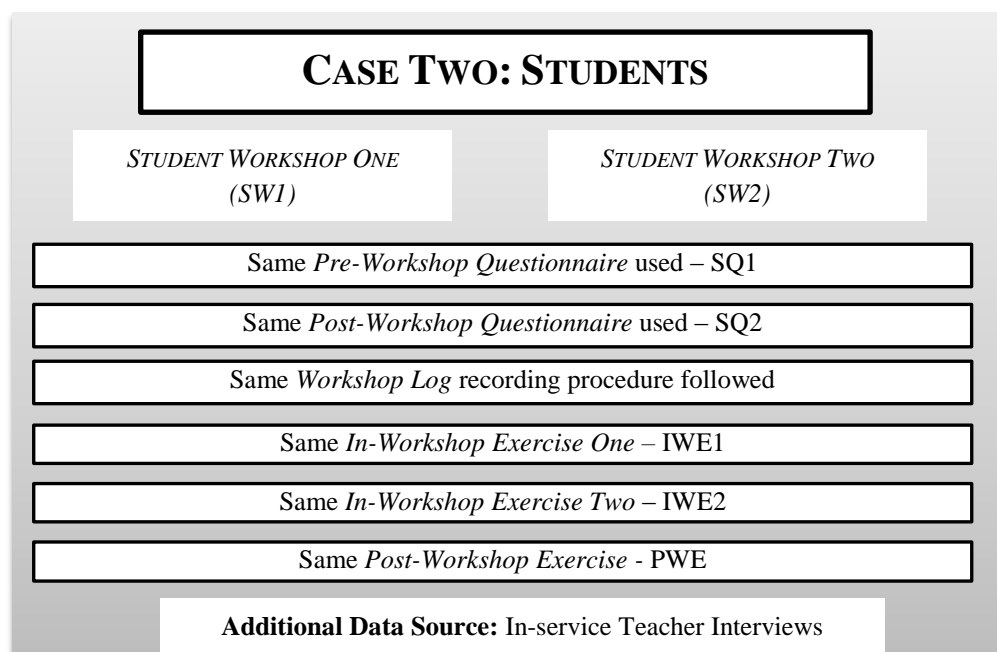


Figure 7-1. Data collection instruments used during Case Two: Students

Student Questionnaire One (SQ1) and Two (SQ2)

The questionnaires used in SW1 and SW2 (SQ1 and SQ2) were similar to those used during Case One. Literature previously identified as useful was considered (Oppenheim, 2000; de Vaus, 2002). Pre- and post-session assessment surveys have previously been completed by research participants to determine their personal opinions of robotic interventions used to support the teaching of programming (Fagin *et al.*, 2001). SQ1 and SQ2 allowed an opportunity to collect a substantial amount of data, from a number of participants, at the same time. These questionnaires were designed to be completed in several minutes and, as they were distributed at the start and end of the workshops, they did not impact upon the running of the sessions. Lessons learned during the exploratory studies influenced presentation and layout. Both questionnaires were designed to be quantitatively, and qualitatively, analysed. Content was devised after considering the objectives of the research and the case study aims. The questionnaires consist of open and closed questions. The use of open questions allowed for the collection of information that could be used to corroborate responses to closed answer questions. A PhD supervisor (TK), and three PhD students, reviewed the questionnaires for errors. Copies of SQ1 and SQ2 can be found in Appendix A11.

SQ1 determines:

- Level of past programming experience
- Programming language most recently learned
- Previous method of learning programming
- Enjoyment of previous programming learning experience
- Consideration of whether previous programming experience was challenging
- Issues associated with learning to program
- Stereotypes associated with learning to program
- Whether participants would consider learning programming independently
- Views on whether programming should be taught in high schools
- Gender

SQ2 determines:

- Enjoyment of their programming experience during the workshop
- Difficulty in completing the programming tasks set during the workshop
- Views on Kobot as a tool to introduce basic programming concepts
- Aspects of Kobot liked
- Aspects of Kobot disliked
- Views on Kobot compared with any previous programming learning experience
- Views on whether Kobot had improved perceptions of programming
- Consideration of whether Kobot helped to dispel stereotypes associated with programming
- Consideration of whether participants would consider learning programming independently
- Views on whether programming should be taught in high schools
- Views on the workshop in general

Student Workshop Logs

A record of workshop events was maintained to supplement other collected data. As outlined previously, self-reflective journals can be used as a supplementary means of data collection (Marotto *et al.*, 2007; Robson, 2011). The workshop logs offered a systematic and consistent way of recording potentially important information and they help to present a clear image of what occurred during the sessions. It was intended for the logs to be descriptive (i.e. an account of events) and reflective. It was believed that the logs would help in the identification of outliers, exceptional cases etc. The recording criteria were as follows:

To Be Documented Immediately During the Workshop

- Issues with environment/equipment (e.g. problem with laboratory, technical failure)
- Issues involving participant involvement (e.g. non-attendance, missing participants)
- Other unidentified potentially critical issues

To Be Documented As Soon As Possible After the Workshop

- Were most participants “on task” during the workshop? If not, why?
- Did participants seem enthusiastic throughout the session? If not, why?
- What aspects of the workshop/programming concepts did participants struggle to grasp?
- Did three or more participants voice concerns about a particular aspect of the workshop?
- Did three or more participants voice concerns about a particular aspect of the simulator?
- Were any concerns raised about the nature of assessment used during the workshop?
- Was a considerable amount of time spent diverging from workshop related activities?
- Are there any other points that need to be noted?

In-Workshop Programming Exercise One and Two (IWE1 & IWE2)

Two multiple-choice/constructed response tests were used during SW1 and SW2. Designed to evaluate understanding of syntax and program behaviour, as suggested by previous work (McCracken *et al.*, 2001), the same exercises were used as during Trainee Teacher Workshop Two (TTW2). The in-workshop exercises allowed an unobtrusive means of monitoring programming progress. The exercises were completed individually under “test conditions” (i.e. individually in silence). Copies of both IWE1 and IWE2 can be found in Appendix A9.

Post-Workshop Programming Exercise (PWE)

A programming exercise, consisting of four programming tasks, was administered. Pseudo code was used to assist participants because past research highlighted the benefit of such an approach (Grandell, 2006). The PWE required around 35-40 minutes. Designed to evaluate programming proficiency, the PWE is an example of performance-based assessment (McCracken *et al.*, 2001) and was also distributed during TTW2. The PWE was completed under test conditions. A robotic agent not encountered during the workshop, named Page, was programmed. Page differed from the other robots used. This is because Page is equipped with eight proximity sensors (opposed to two) and has three encoder sensors capable of detecting 2D objects drawn on the arena floor. Whilst encoder sensors were introduced during the workshop, the specific control methods required to use these sensors was not. As a result, participants were introduced to such sensors when completing the PWE. Use of the Page robot during the post-workshop assessment tasks enabled deep learning to be established. Deep learning is when a learner aims towards understanding whereas surface learning is where learners aim to simply reproduce material without actually understanding it (Case, 2008). By introducing a robot, with different abilities to those already encountered, participants had no option but to adapt and use their workshop knowledge if they were to successfully complete each task. This is because code from earlier exercises or examples could not be copied due to Page, and associated methods, differing. A copy of the PWE can be found in Appendix A9. See Chapter Six for further information on all the programming exercises used.

Teacher Interviews

The views of three in-service FE teachers, on using Kebot as a means of introducing programming concepts, were collected using semi-structured interviews. These are where a researcher sets up a general structure by deciding in advance the main questions to be asked (Drever, 1995). Semi-structured interviews can lead to insightful findings (Hove & Bente, 2005). Themes covered included the suitability and effectiveness of Kebot as a teaching resource. Interviews were recorded and transcribed. These transcriptions were compared with the original recordings six months after completion to ensure reliability. All interviewees involved were familiar with Kebot and had held discussions with their students about the workshop. Interviews lasted for 15 to 20 minutes and were conducted one-to-one. All took place the week following SW1 or SW2 and involved discussion on the following questions:

- In regards to the participants, would you say they were typical of a similar demographic?
- How do you think the workshop session was received by participants?
- In terms of the workshop, do you think that it helped to save time (i.e. would ‘traditional’ methods of teaching the same programming concepts take longer)?
- Compared to traditional methods, do you think participants who take part in this workshop would be more encouraged?
- How many hours would it normally take to cover the same range of material, in a similar amount of depth, using ‘traditional’ methods?
- What modifications would you suggest in order to improve the workshop?
- Do you believe that a robot simulator is an effective tool for supporting the learning of introductory programming?
- Do you believe that a robot simulator improves novices’ perceptions of the subject?
- What are the obstacles for educators using the robot simulator?
- In terms of the research project, can you think of any advice or anything to look for when analysing collected data? What approach would you take when investigating such a topic?

7.2.4 Data Analysis Strategy

In this sub-section the analysis strategy for SW1 and SW2 is outlined.

Student Questionnaire One and Two (SQ1 & SQ2)

Both SQ1 and SQ2 have been subject to quantitative and qualitative analysis. This includes the use of descriptive statistics, tabulation and graphics. A qualitative analytical approach has been used to consider data related to participants' subjective opinions. This has resulted in responses to 'open' questions being analysed to determine trends. Such information has been categorised into groups and quotes have been provided to support identified themes.

Student Workshop Logs

Analysis of the workshop logs involved the identification of important events that may have had an impact upon other data collected. Specific approaches for the analysis of qualitative data, such as thematic coding, were not required as the log data could be analysed in its raw format due to there being a manageable amount of information. Issues such as problems with participant involvement are presented. A summary of workshop events is provided. The full workshop log transcripts have been made available in Appendix A12 to ensure transparency.

In-Workshop Programming Exercise One and Two (IWE1 & IWE2)

Each in-workshop exercise has a maximum score of ten and all questions are marked either correct or incorrect. All of the in-workshop exercises were second marked by a PhD supervisor (PB) to ensure consistency. Three mistakes were noted, with the original scores awarded, and these were corrected. An overall score of six or greater is considered to indicate satisfactory learning. Why such a threshold was selected is discussed in Chapter Eight along with general observations on all programming exercises used. Descriptive statistics have been used to analyse scores while figures and tables present data. As SW1 and SW2 students were enrolled at different institutions, statistical tests have been used to determine whether there was a significant difference in performance.

Post-Workshop Programming Exercise (PWE)

Collected code has been graded according to a three-point scale. The use of letter grades was chosen and these have previously been used to rate programming performance (Lister & Leaney, 2003; Poindexter, 2003). Three-point marking schemes share similarities with the traffic-light assessment strategy that has been used to consider learner performance (Black, 2004). The marking scheme used during SW1 and SW2 is based on the one used during Case One:

- A.** Code shows evidence of deep learning as knowledge gained during the workshop has been used to critically solve a new problem. At least 80% of code is correct.
- B.** Code shows some evidence of deep learning as the new problem has been attempted and successfully solved in part. Between 50% and 80% of code is correct.
- C.** Code shows little evidence of deep learning as no or little attempt has been made to solve the new problem. The participant may have simply tried to copy previous code without adapting it. Less than 50% of code is correct.

As discussed further in Chapter Eight, it was decided that if three-quarters of participants were awarded an A or B for a task then this would provide evidence of learning. A sample of data (code from eight participants) was second marked (by TK) after being marked by the author. There were no differences in scores awarded. Six months later all exercises were also re-examined when preparing this thesis. These measures ensure the consistency of the marking process. Statistical tests have been used to determine whether there was a difference in performance between SW1 and SW2 groups. Data collected from participants with no self-taught programming experience are also presented. This allows for the performance of participants with no programming experience to be considered separately from those who have had greater exposure to the subject. SPSS¹ was used for statistical analysis. The analysis that has been performed has been replicated, and verified, by a researcher (TK) with greater statistical experience.

¹ IBM Corp. Released 2010. IBM SPSS Statistics for Windows, Version 19.0. Armonk, NY: IBM Corp.

Teacher Interviews

Thematic coding has been undertaken to analyse data collected during the three teacher interviews.

Thematic coding is an iterative process that involves:

- Organising information before bringing meaning to it (Rossman & Rallis, 1998)
- The identification of passages of text that exemplify the same idea. Text that represents the same thing is coded to the same name (Gibbs, 2007)
- The grouping of initial codes into a smaller number of themes (Robson, 2011)

Advantages of thematic coding analysis include (Braun & Clarke, 2006): flexibility; accessible results; identification of key features in a large body of data; identification of similarities and differences across a data set; generation of unexpected insights. Thematic coding is also accessible to researchers with limited experience while guidelines outlined by Robson (Robson, 2011) dictated how the coding was carried out. These guidelines identify five main stages: 1) Familiarisation with data; 2) Generating initial codes; 3) Identification of themes; 4) Construction of thematic networks; 5) Integration and interpretation. All coded interview transcripts have been made available in Appendix A13. These have been anonymised and contain no information that can be related to participants. The initial coding was re-examined after a period of six months in order to ensure consistency. This involved the author re-reading transcripts and comparing these with the defined code categories. No changes were made. As only three interviews were undertaken further steps to ensure validation of identified codes were deemed unnecessary.

7.3 Results: Student Workshop One (SW1)

7.3.1 Student Questionnaire One (SQ1)

SQ1 was completed before the workshop by 11 participants.

Past Programming Experience

Two participants had some previous programming experience and details of this are provided in Table 7-1 and Table 7-2. The remaining nine participants had not encountered programming prior to the workshop.

<i>Participant</i>	<i>Language(s)</i>	Beginner	Competent	Expert
SW1-P2	CSS	✓		
	HTML		✓	
	jQuery		✓	
	PhP	✓		
SW1-P5	C#	✓		
	Visual Basic		✓	

Table 7-1. Programming languages SW1 participants had experience using along with their self-identified knowledge of each.

<i>Participant</i>	Language most recently learned	Previously, how did you mainly learn to program?	Which of the following best describes your past programming experience?	Did you find programming challenging?
		<i>Self-Taught / Part of a course or education program / Specify own response</i>	<i>Didn't Like / Indifferent / Enjoyed</i>	<i>Challenging / Indifferent / Trivial</i>
SW1-P2	jQuery	Self-Taught	Enjoyed	Enjoyed
SW1-P5	Visual Basic	Self-Taught	Enjoyed	Indifferent

Table 7-2. Information relating to SW1 participants past programming learning experience.

Opinions of Programming

No participant responded to open questions intended to determine issues or stereotypes associated with programming. Responses to questions associated with views on the learning of programming, and whether the subject should be taught in high schools, have been documented in Table 7-3.

<i>“Would you consider learning to program in your spare time if you were given appropriate support?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	10	1	0
<i>“Do you believe that programming should be an important part of the National Curriculum in Schools?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	9	2	0

Table 7-3. Opinions of SW1 participants on the learning of programming and the teaching of programming in high schools (pre-SW1).

7.3.2 Student Questionnaire Two (SQ2)

SQ2 was completed after the workshop by 11 participants. One of these participants did not attend Day One and this data has been omitted.

Experience Using Kebot

Opinions on the workshop, and of their experience using Kebot, are presented in Table 7-4.

<i>“In regards to your programming experience during the workshop, which of the following best describes your enjoyment?”</i>			
	<i>Enjoyable</i>	<i>Indifferent</i>	<i>Not Enjoyable</i>
Number of Responses	7	3	0
<i>“In regards to the workshop session, which of the following best describes how difficult the programming tasks have been?”</i>			
	<i>Easy</i>	<i>Neither difficult nor easy</i>	<i>Difficult</i>
Number of Responses	1	8	1
<i>“Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	9	1	0

Table 7-4. SW1 participants’ opinions on their workshop experience and use of Kebot.

To establish how effective participants believed elements of the workshop to have been, they were instructed to score on a scale of one (not at all effective) to five (extremely effective) their views on the: Kobot simulator, programming support received, presentation delivered; teaching environment. Responses can be seen in Table 7-5.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		1	2	3	4	5
Kobot Simulator	4.3	0	0	0	7	3
Programming Support	4.6	0	0	0	4	6
Workshop Presentation	4.4	0	0	1	4	5
Environment	4.4	0	0	1	4	5

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 7-5. SW1 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.

The two participants with self-taught programming experience (SW1-P2 and SW1-P5) stated that their previous introduction to programming was “*More effective*” and “*About the same effectiveness*” respectively. As documented in Table 7-6, all participants were asked to consider whether Kobot had helped to improve their perceptions of programming and whether the software had helped to dispel any negative stereotypes.

<i>“Has the robot simulator helped to improve your perception of programming?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	9	0	1
<i>“Has the robot simulator helped to dispel any negative stereotypes you had about programming before the session?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	4	6	1

Table 7-6. SW1 participants' views on the effect of Kobot upon their opinions of programming.

Two open questions were used to establish what aspects of Kebot were liked and not liked. Instructions were provided so up to three aspects were specified for each. All data is displayed in Table 7-7. SW1 participants identified 27 aspects of Kebot that they liked and 10 that they did not.

<i>Participant</i>	<i>Aspects of Kebot liked</i>	<i>Aspects of Kebot disliked</i>
SW1-P1	<i>“Easy interface that clearly showed what buttons to press”</i> <i>“List of code at the top of each coding screen to help remember the instructions”</i> <i>“Easy to program robot with code”</i>	
SW1-P2	<i>“Easy to use”</i> <i>“Easy preview and simulation of programming code - type in one window, robot performs in the other”</i> <i>“Good scenarios and arenas”</i>	<i>“Help & Selection: gives hints on what can be added to the code e.g. if - then it gives you possible options that can be entered. Prompts”</i>
SW1-P3	<i>“Easy to work providing you have an instructor”</i> <i>“New experience”</i> <i>“Messing about with the robot in the arena”</i>	<i>“Braces”</i> <i>“Could be difficult to understand”</i> <i>“Difficult to start”</i>
SW1-P4	<i>“Different environments”</i> <i>“Good software functions”</i> <i>“Highlights errors”</i>	
SW1-P5	<i>“The ability to see the results.”</i> <i>“How each robot could do different tasks”</i> <i>“The challenges in each arena”</i>	<i>“Hard to draw on (the arena)”</i> <i>“The arena would pop-up saying the robot had left the arena when it hadn't”</i>
SW1-P6	<i>“Efficient and interesting”</i>	<i>“It was kind of confusing at times”</i>
SW1-P8	<i>“Interesting”</i> <i>“Easy to get used to”</i> <i>“Effective”</i>	<i>“Could be confusing”</i>
SW1-P9	<i>“New experience”</i> <i>“Plenty of information”</i>	<i>“Hard at the start. Very challenging to change certain values at the start of the course”</i>
SW1-P10	<i>“Very interactive.”</i> <i>“Fun and enjoyable”</i> <i>“Well explained”</i>	
SW1-P14	<i>“Interesting”</i> <i>“Challenging”</i> <i>“New”</i>	<i>“Difficult at times”</i>

Table 7-7. Aspects of Kebot liked and disliked by SW1 participants.

General Opinions of Programming and the Workshop

Two questions used on SQ1 were also used on SQ2. These questions were associated with views on learning programming and whether the subject should be taught in high schools. Collected data is presented in Table 7-8.

<i>“Would you consider learning to program in your spare time if you were given appropriate support?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	6	4	0

<i>“Do you believe that programming should be an important part of the National Curriculum in Schools?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	7	3	0

Table 7-8. Opinions of SW1 participants on the learning of programming and the teaching of programming in high schools (post-SW1).

An opportunity to offer comments about the session was offered and responses were:

[SW1-P2] “Good simulator that combines with the programmer window well”

[SW1-P3] “Overall I enjoyed this a lot. It was a new experience, was enjoyable as well as teaching me something new”

[SW1-P4] “Good. Quite enjoyable. Helped my understanding”

[SW1-P5] “I enjoyed the session and would love to continue programming”

[SW1-P6] “Very enjoyable”

[SW1-P8] “Enjoyable!”

[SW1-P9] “Good and fun software which is effective for first time learners”

[SW1-P10] “It was a fun experience and more people should be offered the chance to learn about programming”

[SW1-P14] “Interesting to participate in”

7.3.3 SW1 Workshop Log – Day One Overview

The workshop took place on a hot day and this led to the laboratory being warm. Some participants struggled to see the workshop slides at points due to the size of the lab. All participants declined an offer to receive paper copies of the workshop slides. This may be due to participants not being used to such materials. All participants arrived promptly (bar SW1-P14 who arrived 45 minutes late and required assistance to catch up). One participant (SW1-P8) had to leave the session slightly early due to other commitments and did not complete IWE1. On the whole participants remained “on task”. Three participants in particular appeared very engaged (SW1-P1, SW1-P2, SW1-P5) and advanced beyond the rest of the group. Three participants (SW1-P7, SW1-P8, SW1-P10) commented that they felt bemused by programming concepts at times. If...Else statements, the location of braces and some Java operators caused the majority a problem at some point. Participants reported encountering few other issues. It is considered that this might be down to an unwillingness to ask questions as some participants seemed to struggle at times. Potentially a different approach to teaching, different teacher or fear of embarrassment in front of peers may have been responsible. Considering most participants had no prior programming experience progress was satisfactory. It is considered, however, that the intensive nature of the workshop could have impacted upon participant performance. This is because participants’ typical lessons normally last no more than 55 minutes which is in contrast to the workshop even considering regular breaks were given. Some participants appeared reluctant to experiment with their code and wanted to create a “complete” solution in advance of running the simulator. This may have been due to a worry of “damaging” the software used. IWE1 was completed in silence and this took around five minutes. A complete copy of the Workshop Log for SW1 (Day One) is available in Appendix A12.

7.3.4 SW1 Workshop Log – Day Two Overview

There were no issues with the research environment or equipment. One participant who attended Day One was not able to attend (SW1-P7) while one participant attended Day Two who had not attended Day One (SW1-P15). Participant enjoyment during Day Two appeared to be higher. This may have been due to the different laboratory setting and as PCs were closer together. There were few disruptions and the workshop appeared to be insightful and enjoyable. No participants struggled throughout although SW1-P10 said she felt overwhelmed at points. Two participants (SW1-P6 and SW1-P8) displayed a willingness to experiment and attempted to perfect code. This is in contrast to others who seemed content to put in the minimum required effort to complete a challenge (SW1-P9 and SW1-P14). One participant demonstrated advanced concept knowledge and effortlessly completed tasks (SW1-P5). Remaining participants remained engaged and had varying degrees of success. For Loops and Array syntax caused significant confusion and the introduction offered during the workshop did not appear to be sufficient. Students also struggled with the format of Nested If statements. Some aspects of Kobot, related to drawing tools, caused frustration (although this is not considered to be a failing of the software). It was commented by participants that IWE2 was much more challenging compared to IWE1. The PWE tasks seemed to be enjoyed and were completed individually in silence. Task Four, however, appeared to be hastily completed by most participants. Few appeared to make a genuine attempt to solve the problem and seemed content to have completed Task One to Three. Indeed, only SW1-1, SW1-5 and SW1-6 made any meaningful attempt to complete Task Four. Finally, compared to Day One, participants appeared much more willing to ask questions and were happier to experiment with their code and Kobot. A complete copy of the Workshop Log for SW1 (Day Two) is available in Appendix A12.

7.3.5 In-Workshop Programming Exercise One (IWE1)

Table 7-9 displays participants' performance on IWE1. '1' indicates a correct answer and '0' an incorrect answer. 10 participants completed this exercise. The highest score was the maximum ten (one participant), the lowest was four (one participant), the mean is 6.9/10 and the standard deviation is 1.79. Figure 7-2 presents the combined scores of participants on each question.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
SW1-P1	1	1	1	0	1	1	0	0	1	1	7
SW1-P2	1	1	1	1	1	1	0	0	1	1	8
SW1-P3	1	1	1	0	0	1	1	0	1	0	6
SW1-P4	1	1	1	0	1	0	1	0	1	0	6
SW1-P5	1	1	1	1	1	1	1	1	1	1	10
SW1-P6	1	1	1	0	1	0	1	1	1	0	7
SW1-P7	1	1	1	0	1	0	1	1	1	0	7
SW1-P9	1	1	1	0	1	0	1	0	0	0	5
SW1-P10	1	1	1	1	1	1	1	1	1	0	9
SW1-P14	0	1	1	0	1	0	1	0	0	0	4
Totals	9	10	10	3	9	5	8	4	8	3	/

Table 7-9. Breakdown of SW1 participants' performance on IWE1.

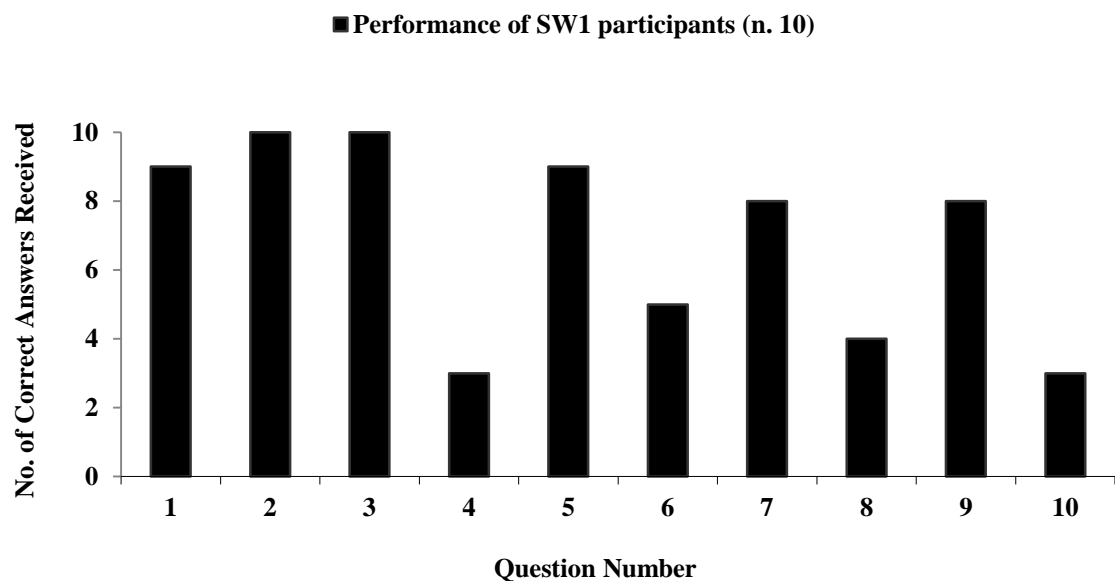


Figure 7-2. Combined scores of all SW1 participants on IWE1.

7.3.6 In-Workshop Programming Exercise Two (IWE2)

Table 7-10 displays participants' performance on IWE2. Again a '1' indicates a correct answer and '0' an incorrect answer. The responses of 10 participants are considered. The highest score was nine (one participant), the lowest was two (two participants), the mean is 5.6/10 and the standard deviation is 2.46. Figure 7-3 presents the combined scores of participants.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
SW1-P1	1	0	1	1	1	1	1	1	1	0	8
SW1-P2	1	0	1	1	1	1	1	1	1	0	8
SW1-P3	0	0	0	0	0	1	1	0	0	0	2
SW1-P4	1	0	1	0	1	1	0	1	0	0	5
SW1-P5	1	1	1	1	1	1	1	1	1	0	9
SW1-P6	1	1	1	1	0	1	1	1	0	0	7
SW1-P8	0	0	1	1	0	1	1	1	0	0	5
SW1-P9	1	0	0	0	0	0	1	0	0	0	2
SW1-P10	0	0	1	1	1	1	1	1	0	0	6
SW1-P14	1	0	0	0	1	1	1	0	0	0	4
Totals	7	2	7	6	6	9	9	7	3	0	/

Table 7-10. Breakdown of SW1 participants' performance on IWE2.

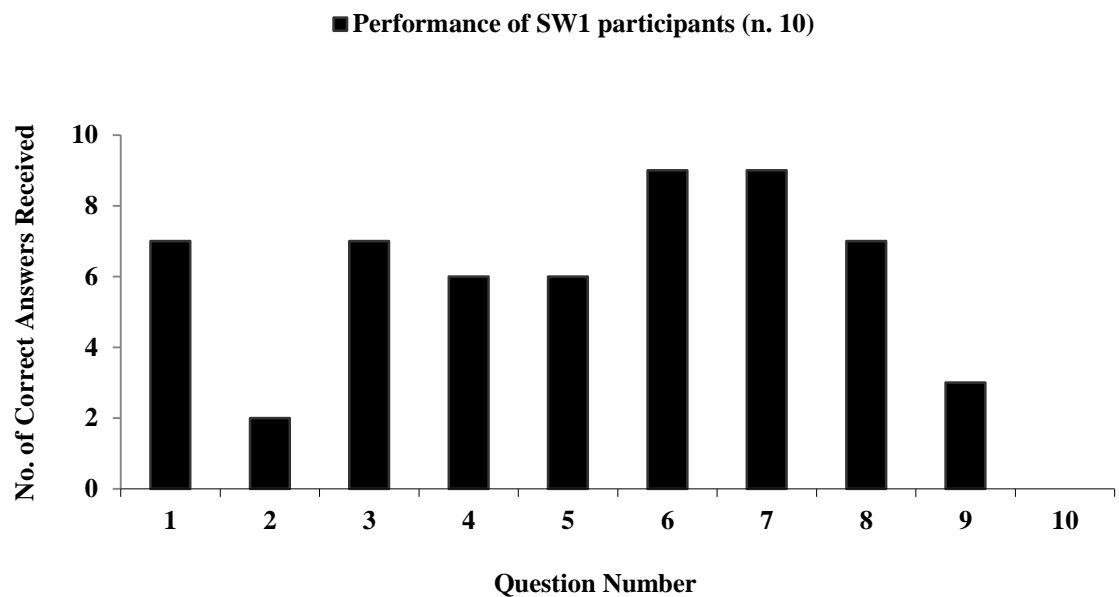


Figure 7-3. Combined scores of all SW1 participants on IWE2.

7.3.7 Combined Performance on IWE1 and IWE2

Participants overall in-workshop score, after combining IWE1 and IWE2 totals, is displayed in Table 7-11. One participant (SW1-P8) had to leave the session slightly early and did not complete IWE1. It has not been possible, therefore, to include this participant's responses in the aggregation.

<i>Participant Number</i>	Overall In-Workshop Score (IWE1 and IWE2 total scores combined)
SW1-1	15
SW1-2	16
SW1-3	8
SW1-4	11
SW1-5	19
SW1-6	14
SW1-8	<i>Participant did not submit IWE1</i>
SW1-9	7
SW1-10	15
SW1-14	8

Table 7-11. Combined means scores of SW1 participants on the in-workshop exercises.

7.3.8 Post-Workshop Exercise (PWE)

PWE consisted of four separate programming tasks and all participants submitted their code after around 35 minutes. As outlined in Section 7.2.4, each participant was awarded a grade A, B or C for their performance on each task. Table 7-12 displays a breakdown of scores awarded.

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
SW1-1	B	A	A	C
SW1-2	B	A	C	C
SW1-3	A	B	B	C
SW1-4	B	B	C	C
SW1-5	A	A	A	A
SW1-6	B	B	A	C
SW1-8	B	A	A	C
SW1-9	B	B	B	C
SW1-10	A	B	C	C
SW1-14	B	A	C	C
Total (n. 10)	A – 3	A – 5	A – 4	A – 1
	B – 7	B – 5	B – 2	B – 0
	C – 0	C – 0	C – 4	C – 9

Table 7-12. Breakdown of SW1 participants' performance on the PWE.

7.4 Results: Student Workshop Two (SW2)

7.4.1 SW2 Questionnaire One (SQ1)

SQ1 was completed before the workshop by 10 participants.

Past Programming Experience

As documented in Teacher Interview 1 (discussed in Section 7.7), all ten SW2 participants had previously been introduced to elements of Visual Basic (VB) several months before the workshop. For six of these this was their only exposure to programming. Four participants (SW2-P10, SW2-P11, SW2-P12, SW2-P14), however, had also attempted to learn programming by themselves. See Table 7-13 and Table 7-14 for full details of this information.

<i>Participant</i>	<i>Language(s)</i>	Beginner	Competent	Expert
SW2-P9	Visual Basic	✓		
SW2-P10	C#		✓	
	PhP		✓	
	Visual Basic		✓	
SW2-P11	C++	✓		
	Java		✓	
	PhP		✓	
	Visual Basic		✓	
SW2-P12	C++	✓		
	C#	✓		
	Java	✓		
	Visual Basic		✓	
SW2-P14	C++		✓	
	C#	✓		
	Visual Basic		✓	
SW2-P15	Visual Basic		✓	
SW2-P16	Visual Basic		✓	
SW2-P17	Visual Basic	✓		
SW2-P18	Visual Basic	✓		
SW2-P19	Visual Basic	✓		

Table 7-13. Programming languages SW2 participants had experience using along with their self-identified knowledge of each.

<i>Participant</i>	Language most recently learned	Previously, how did you mainly learn to program?	Which of the following best describes your past programming experience?	Did you find programming challenging?
		<i>Self-Taught / Part of a course or education program / Specify own response</i>	<i>Didn't Like / Indifferent / Enjoyed</i>	<i>Challenging / Indifferent / Trivial</i>
SW2-P9	Visual Basic	Part of a course or education program	Enjoyed	Indifferent
SW2-P10	C#	Self-Taught	Enjoyed	Indifferent
SW2-P11	Java	Self-Taught	Enjoyed	Indifferent
SW2-P12	Java	Self-Taught	Enjoyed	Indifferent
SW2-P14	C++	Self-Taught	Enjoyed	Indifferent
SW2-P15	Visual Basic	Part of a course or education program	Enjoyed	Indifferent
SW2-P16	Visual Basic	Part of a course or education program	Enjoyed	Indifferent
SW2-P17	Visual Basic	Part of a course or education program	Enjoyed	Challenging
SW2-P18	Visual Basic	Part of a course or education program	Enjoyed	Indifferent
SW2-P19	Visual Basic	Part of a course or education program	Enjoyed	Indifferent

Table 7-14. Information relating to SW2 participants past programming learning experience.

Opinions of Programming

Four participants identified issues that they associated with the learning of programming:

[SW2-P9] “It can be difficult if only little/no support is given”

[SW2-P10] “Some languages have a lack of example code and support”

[SW2-P12] “Examples used to learn from not always helpful”

[SW2-P14] “Stress”

Five participants identified stereotypes that they associated with learning to program:

[SW2-P10] “You have to read a lot of text”

[SW2-P11] “Textbooks”

[SW2-P12] “Large amount of reading”

[SW2-P18] “Nerds (gamers), male, quite geeky and only talk about games and memes”

[SW2-P19] “Geeky people (not a bad thing!)”

Responses to questions associated with views on learning programming, and whether the subject should be taught in high schools, have been documented in Table 7-15.

<i>“Would you consider learning to program in your spare time if you were given appropriate support?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	9	1	0
<i>“Do you believe that programming should be an important part of the National Curriculum in Schools?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	5	4	1

Table 7-15. Opinions of SW2 participants on the learning of programming and the teaching of programming in high schools (pre-SW2).

7.4.2 SW2 Questionnaire Two (SQ2)

SQ2 was completed after the workshop. Nine students completed SQ2. One of these participants, however, did not attend Day One. Data collected from this participant has been omitted due to this.

Experience Using Kobot

Participants' opinions of their workshop experience, and of their experience using Kobot, are presented in Table 7-16. Enjoyment, difficulty and views on effectiveness have been considered.

<i>“In regards to your programming experience during the workshop, which of the following best describes your enjoyment?”</i>			
	<i>Enjoyable</i>	<i>Indifferent</i>	<i>Not Enjoyable</i>
Number of Responses	7	0	1
<i>“In regards to the workshop session, which of the following best describes how difficult the programming tasks have been?”</i>			
	<i>Easy</i>	<i>Neither difficult nor easy</i>	<i>Difficult</i>
Number of Responses	3	4	1
<i>“Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	8	0	0

Table 7-16. SW2 participants' opinions on their workshop experience and use of Kobot.

Views were also collected on the: Kobot simulator, programming support received, presentation delivered and teaching environment. Responses can be seen in Table 7-17.

Workshop Component	Mean Score (maximum of 5)	Score Breakdown (by no. of responses) ^a				
		<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>
Kobot Simulator	4.5	0	0	0	4	4
Programming Support	4.25	0	0	1	4	3
Workshop Presentation	4.13	0	0	1	5	2
Environment	3.75	0	0	2	6	0

^a n.b. 1 (not at all effective) to 5 (extremely effective)

Table 7-17. SW2 participants' opinions on the effectiveness of elements of the workshop. Arranged by the number of participants who selected each option.

Two open questions were used to establish what aspects of Kebot were liked and not liked. Instructions were provided so up to three aspects were specified for each. All collected data is displayed in Table 7-18. SW2 participants identified 14 aspects of Kebot that they liked and eight that they did not.

<i>Participant</i>	<i>Aspects of Kebot liked</i>	<i>Aspects of Kebot disliked</i>
SW2-P9	<i>“That we could visualise what our programming was doing”</i>	
SW2-P10	<i>“Can show previous trajectories”</i>	<i>“Have to use right click and void main option to run simulator”</i>
SW2-P11	<i>“Previous trajectories”</i>	<i>“Clicking void main every time you run it”</i>
SW2-P12	<i>“You have a representation of what you spent your time doing”</i> <i>“[You can] watch what the robot does/how it responds”</i> <i>“The trajectories and other functions such as that”</i>	<i>“More functionality to the simulator”</i>
SW2-P16	<i>“Shows physical consequences of logical and arithmetic functions”</i> <i>“Shows code only does what you tell it”</i>	<i>“A little slow for people already familiar with programming”</i>
SW2-P17	<i>“Putting us into an interactive working environment made the simple things seem more interesting”</i> <i>“The practicality was more engaging”</i> <i>“Robots are awesome”</i>	<i>“I hate maths - felt a bit too complicated for a taster”</i>
SW2-P18	<i>“Easier than full-on programming and more enjoyable”</i> <i>“Can see working with robots”</i>	<i>“Long time needed lots of patience”</i>
SW2-P19	<i>“Programming a maze avoider”</i>	<i>“Only one robot at a time”</i> <i>“Not enough sensors”</i>

Table 7-18. Aspects of Kebot liked and disliked by SW2 participants.

As documented in Table 7-19, participants were asked to consider whether Kebot had helped to improve their perceptions of programming and whether the software had helped to dispel any negative stereotypes. They were also asked to contrast their previous programming experience to the one using Kebot.

<i>“Has the robot simulator helped to improve your perception of programming?”</i>					
	<i>Yes</i>		<i>Not Sure</i>		<i>No</i>
Number of Responses	5		2		1
<i>“Has the robot simulator helped to dispel any negative stereotypes you had about programming before the session?”</i>					
	<i>Yes</i>		<i>Not Sure</i>		<i>No</i>
Number of Responses	2		4		2
<i>“If you have previously learnt a programming language, was your previous introduction to basic programming...”</i>					
	<i>Much less effective</i>	<i>Less effective</i>	<i>About the same effectiveness</i>	<i>More effective</i>	<i>Much more effective</i>
Number of Responses	0	6	0	2	0

Table 7-19. SW2 participants’ views on the effect of Kebot upon their opinions of programming.

General Opinions of Programming and the Workshop

Two questions used on SQ1 were used on SQ2. These were associated with views on learning programming and whether the subject should be taught in high schools (see Table 7-20).

<i>“Would you consider learning to program in your spare time if you were given appropriate support?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	8	0	0
<i>“Do you believe that programming should be an important part of the National Curriculum in Schools?”</i>			
	<i>Yes</i>	<i>Not Sure</i>	<i>No</i>
Number of Responses	5	2	1

Table 7-20. Opinions of SW2 participants on the learning of programming and the teaching of programming in high schools (post-SW2).

SQ2 was concluded with an opportunity to offer general comments about Kobot or the session:

[SW2-P10] “Software good for an introduction to programming, very easy to use. Syntax colour coding good help to distinguish sections better”

[SW2-P11] “The robot simulator is good for novices as it shows them actually producing something on the screen they can see”

[SW2-P12] “Sessions are enjoyable and using the simulator as well as the BlueJ software made the whole process easier and more enjoyable. Maybe some more challenging tasks at the end”

[SW2-P16] “Seems effective for younger teenagers (13 - 16) but not so tailored for those who have experience. Good introduction to programming”

[SW2-P19] “Good fun”

7.4.3 SW2 Workshop Log – Day One Overview

No issues occurred during the workshop in regards to the workshop environment. Participants remained attentive throughout. This prompted the member of FE staff present to remark how participants must have enjoyed the experience. The limits of Kebot were pushed by a number of participants looking to elicit further behaviour from the software. In some cases extra variables were declared and the internet searched for other syntax. Whilst questions were asked, in regards to the software used and programming code, upon explanation further help was not normally required. IWE1 was enthusiastically completed. Some participants stayed after the session had officially ended to carry on working with Kebot. A complete copy of the Workshop Log for SW2 (Day One) is available in Appendix A12.

7.4.4 SW2 Workshop Log – Day Two Overview

SW2-P14 and SW2-P15 did not attend Day Two. SW2-P22 attended Day Two but not Day One. The increased complexity of Day Two appeared to be welcomed. The only participant to consistently struggle was SW2-P17. Few programming concepts caused repeated problems. No concerns were raised in regards to the workshop itself although, at points, a number of participants completed assigned tasks and were left waiting for their peers to catch up. Both IWE2 and the PWE were completed with enthusiasm and the on-location member of FE staff was again surprised by the level of concentration demonstrated. With the exception of one participant (SW2-P17) all participants were observed to make a solid attempt at Task Four, although not all managed to elicit the desired behaviour from their robotic agents due to the use of concepts which were advised against during the workshop. Several participants worked through their lunch break on Kebot. Some expressed after the session how they would like for Kebot to form part of their coursework project the following academic year. A complete copy of the Workshop Log for SW2 (Day Two) is available in Appendix A12.

7.4.5 In-Workshop Programming Exercise One (IWE1)

Table 7-21 displays participants' performance on IWE1. '1' indicates a correct answer and '0' an incorrect answer. 10 participants completed this exercise. The highest score was the maximum ten (four participants), the lowest was six (one participant), the mean is 9/10 and the standard deviation is 1.25. Figure 7-4 presents the combined scores of participants on each question.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
SW2-P9	1	1	1	1	1	1	1	1	1	1	10
SW2-P10	1	1	1	0	1	1	1	1	1	1	9
SW2-P11	1	1	1	1	1	1	1	1	1	1	10
SW2-P12	1	1	1	1	1	1	0	1	1	1	9
SW2-P14	1	1	1	1	1	1	0	1	1	1	9
SW2-P15	1	1	1	1	1	1	1	1	1	1	10
SW2-P16	1	1	1	1	1	1	1	1	0	0	8
SW2-P17	0	1	1	0	1	0	1	0	1	1	6
SW2-P18	1	1	1	1	1	1	1	1	1	1	10
SW2-P19	1	1	1	1	1	0	1	1	1	1	9
Totals	9	10	10	8	10	8	8	9	9	9	/

Table 7-21. Breakdown of SW2 participants' performance on IWE1.

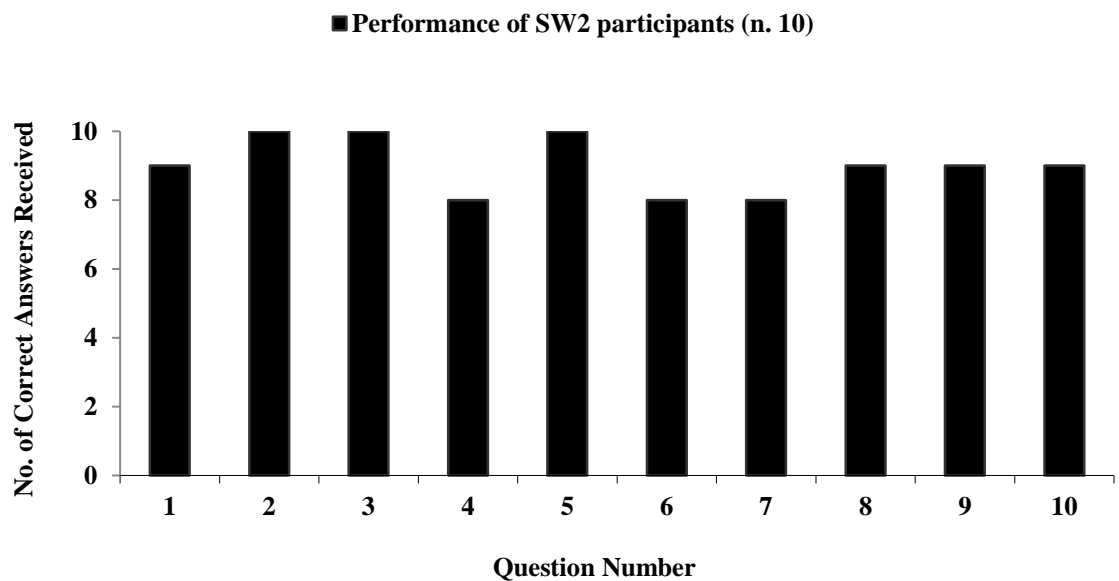


Figure 7-4. Combined scores of all SW2 participants on IWE1.

7.4.6 In-Workshop Programming Exercise Two (IWE2)

Table 7-22 displays participants' performance on IWE2. Again a '1' indicates a correct answer and '0' an incorrect answer. The responses of eight participants are considered. The highest score was nine (two participants), the lowest was three (one participant), the mean is 6.75/10 and the standard deviation is 2.25. Figure 7-5 presents the combined scores of participants.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
SW2-P9	0	0	1	1	1	1	1	1	0	0	6
SW2-P10	1	1	1	1	1	0	1	1	1	1	9
SW2-P11	1	1	1	1	1	1	1	1	1	0	9
SW2-P12	1	1	1	1	1	1	1	0	1	0	8
SW2-P16	0	0	1	0	0	1	1	0	1	0	4
SW2-P17	0	0	0	0	1	0	1	1	0	0	3
SW2-P18	1	1	1	1	1	1	1	0	1	0	8
SW2-P19	0	0	0	1	1	1	1	1	1	1	7
Totals	4	4	6	6	7	6	8	5	6	2	/

Table 7-22. Breakdown of SW2 participants' performance on IWE2.

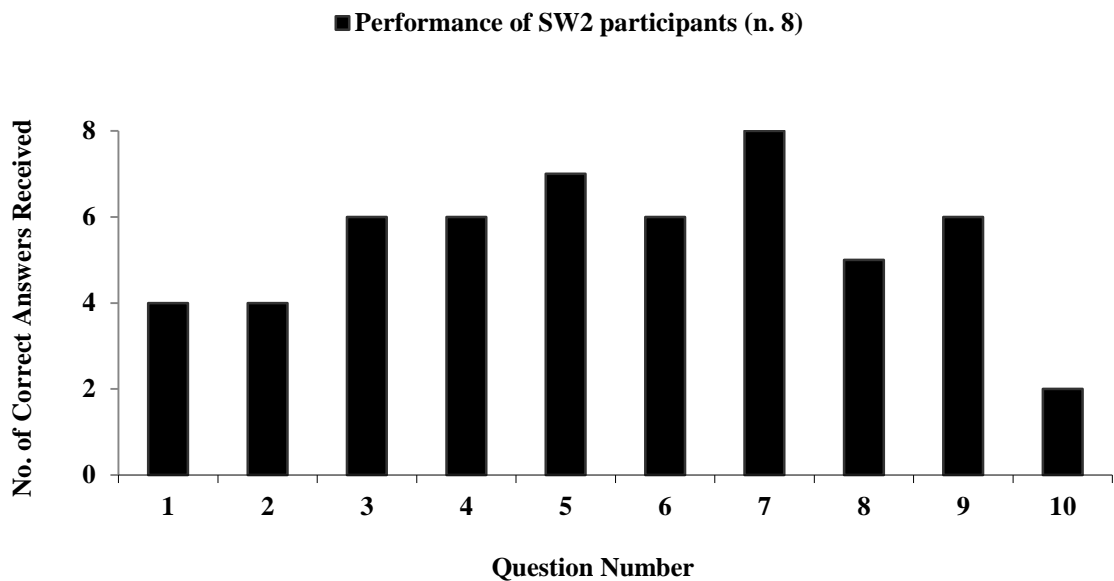


Figure 7-5. Combined scores of all SW2 participants on IWE2.

7.4.7 Combined Performance on IWE1 and IWE2

Participants overall in-workshop score, after combining their IWE1 and IWE2 total scores, is displayed in Table 7-23.

<i>Participant Number</i>	Overall In-Workshop Score (IWE1 and IWE2 total scores combined)
SW2-P9	16
SW2-P10	18
SW2-P11	19
SW2-P12	17
SW2-P16	12
SW2-P17	9
SW2-P18	18
SW2-P19	16

Table 7-23. Combined means scores of SW2 participants on the in-workshop exercises.

7.4.8 Post-Workshop Exercise (PWE)

As outlined in Section 7.2.4, each participant was awarded grade A, B or C for their performance on the PWE tasks. Four separate programming exercises were completed and code was submitted after around 35 minutes. Table 7-24 displays a breakdown of scores.

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
SW2-P9	B	A	A	C
SW2-P10	A	B	A	B
SW2-P11	B	A	A	B
SW2-P12	A	A	B	B
SW2-P16	B	B	B	C
SW2-P17	C	B	C	C
SW2-P18	A	A	A	B
SW2-P19	A	A	A	B
Total (n. 8)	A – 4	A – 5	A – 5	A – 0
	B – 3	B – 3	B – 2	B – 5
	C – 1	C – 0	C – 1	C – 3

Table 7-24. Breakdown of SW2 participants' performance on the PWE.

7.5 Combined Analysis of Programming Performance

In this section programming exercise data is compared and contrasted. Statistical analysis has been undertaken.

7.5.1 In-Workshop Programming Exercise One (IWE1)

Figure 7-6 displays the performance of SW1 and SW2 participants on IWE1.

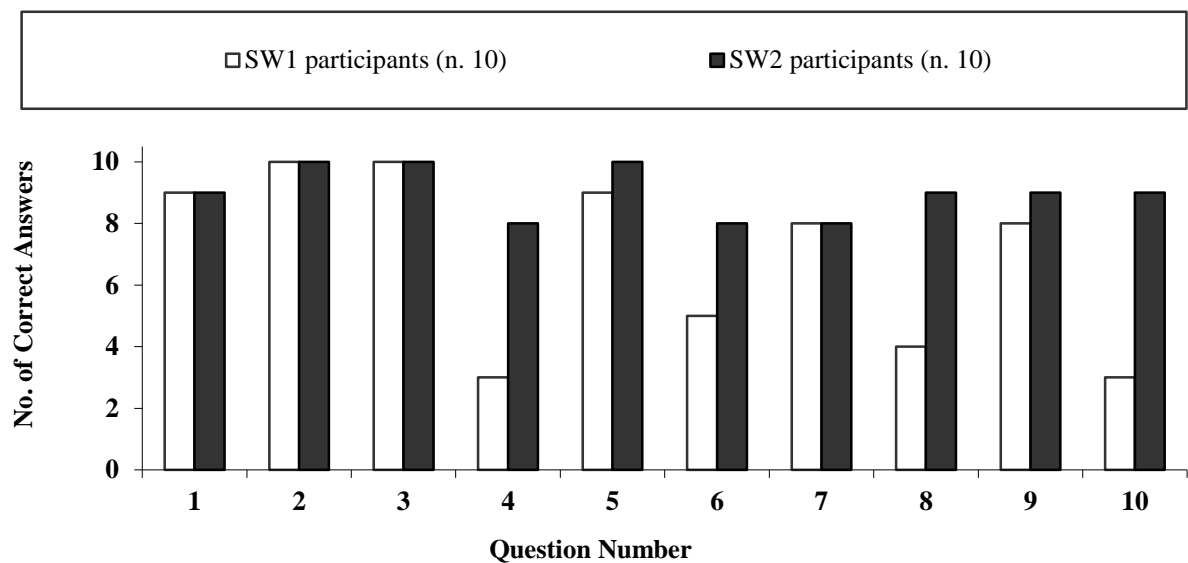


Figure 7-6. SW1 and SW2 participant performance on IWE1.

The independent *t*-test allows the means of two groups to be compared and has previously been used to consider student performance (Alavi, 1994). As SW1 and SW2 participants are enrolled at different FE institutions, an attempt was made to use the *t*-test to determine whether there was a statistically significant difference in overall performance on IWE1. A pre-requisite of the *t*-test is that data must be normally distributed. Scores for SW1 were normally distributed as assessed by the Shapiro-Wilks test ($p > .05$). However, scores for SW2 were not normally distributed as the p value ($p = .007$) was not greater than the chosen alpha level ($\alpha = .05$). The Mann-Whitney *U* test was instead used as this is the non-parametric alternative of the *t*-test. It was observed that $p = .011$ ($U = 17.5$). It can be concluded, therefore, that there is a statistically significant difference between the two groups in terms of overall scores on IWE1.

7.5.2 In-Workshop Programming Exercise Two (IWE2)

Figure 7-7 displays the performance of SW1 and SW2 participants on IWE2. As there were fewer SW2 participants the mean scores of the SW2 group have been presented to allow for a direct comparison.

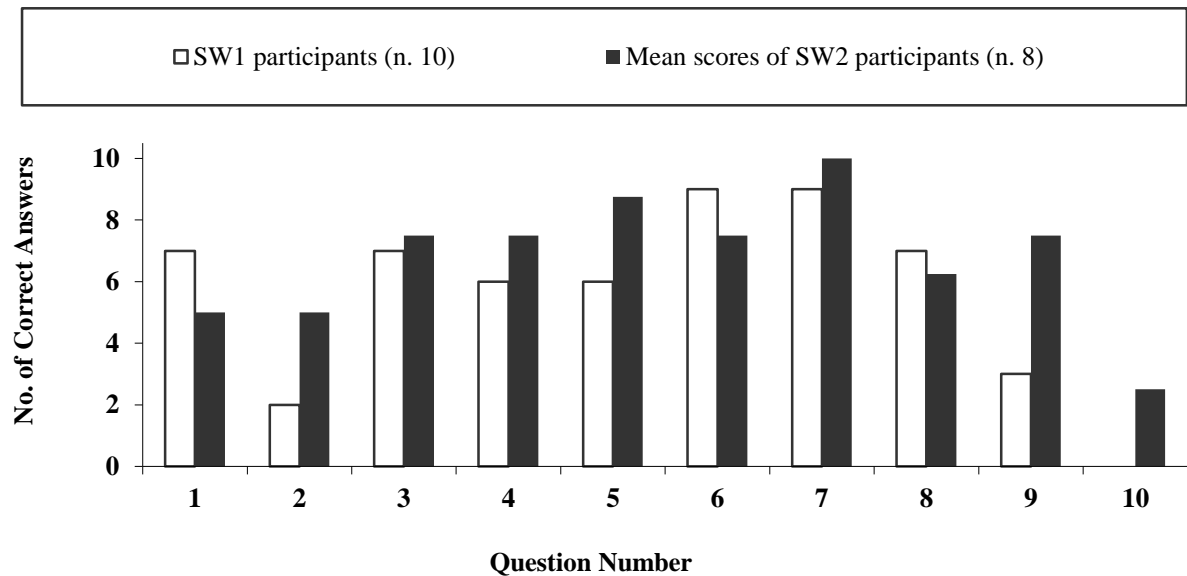


Figure 7-7. SW1 and SW2 participant performance on IWE2.

Scores for both SW1 and SW2 groups were normally distributed as assessed by the Shapiro-Wilks test ($p > .05$). There was also a homogeneity of variances for SW1 and SW2 scores, as assessed by Levene's Test for Equality of Variances ($p = .749$). This allowed the independent t -test to be used to determine whether there was a statistically significant difference in the mean total scores, between SW1 and SW2 participants, on IWE2. The t -test was selected, over the Mann-Whitney U test, because of the greater power of parametric tests (Siegel, 1957). No statistically significant difference was found as $t(16) = 1.02$, $p = .322$.

7.5.3 Post-Workshop Programming Exercise (PWE)

Table 7-25 displays a breakdown of scores for participants. Data collected from the 13 participants (eight from SW1, five from SW2) with no self-taught programming experience is also presented.

<i>Participants</i>	Task 1			Task 2			Task 3			Task 4		
	<i>a</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>c</i>
Group A (<i>n.</i> 10)	3	7	0	5	5	0	4	2	4	1	0	9
Group B (<i>n.</i> 8)	4	3	1	5	3	0	5	2	1	0	5	3
Total (<i>n.</i> 18)	7 39%	10 55%	1 6%	10 56%	8 44%	0 0%	9 50%	4 22%	5 28%	1 6%	5 28%	12 66%
No self-taught experience (<i>n.</i> 13)	4 31%	8 61%	1 8%	6 46%	7 54%	0 0%	6 46%	3 23%	4 31%	0 0%	2 15%	11 85%

Table 7-25. Breakdown of SW1 and SW2 performance on the PWE.

Due to their different backgrounds statistical analysis was undertaken to investigate whether the groups performed differently on the PWE. Letter grades (as used to mark tasks) are examples of ordinal data (Stewart & White, 1976). The *t*-test could not, therefore, be applied as it is only suitable for analysis of interval or ratio data. Instead, the Mann-Whitney *U* test was selected as it is suitable for analysing ordinal data. No significant difference was found between the groups: Task One ($U = 35.5$, $p = .696$), Task Two ($U = 35$, $p = .696$), Task Three ($U = 28$, $p = .315$) and Task Four ($U = 21.5$, $p = .101$). Figure 7-8 displays the combined performance of participants.

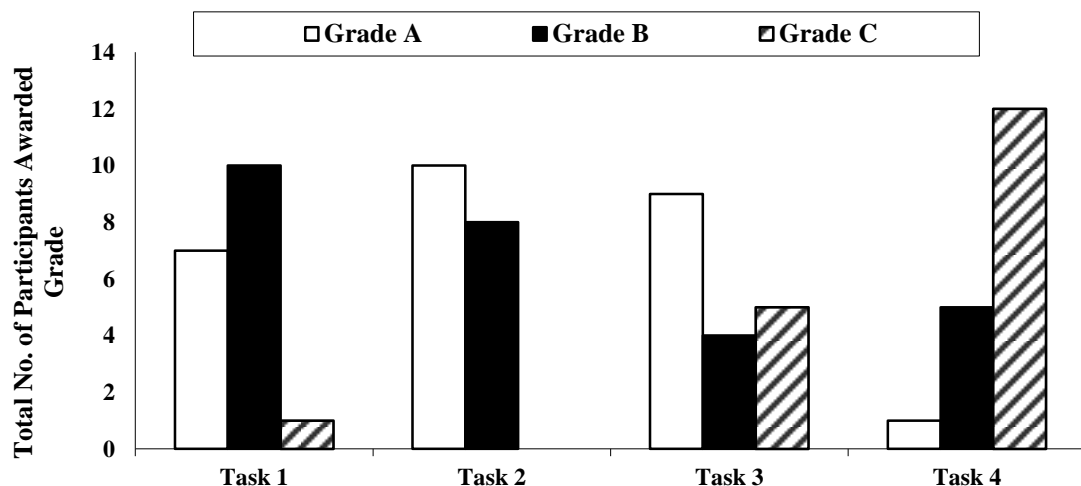


Figure 7-8. Combined performance of SW1 and SW2 participants on the PWE.

7.6 Reliability and Validity of Programming Exercises

A reliable assessment is one that would produce similar results when used by participants of similar ability in the same circumstances; a valid assessment is one that measures what it professes to measure (Crisp & Palmer, 2007). In this sub-section the reliability of the post-workshop programming exercises are considered.

7.6.1 Reliability of the Post-Workshop Programming Exercises (PWE)

Item difficulty is the proportion of participants who answer a test item correctly. P-values are used to denote the proportion of participants who get an item correct (Varma, 2006). A p-value is obtained by dividing the percentage of correct answers by the number of responses received (Smith *et al.*, 2008). Extreme p-values (e.g. 0.0 or 1.0) restrict the reliability of test scores (Matlock-Hetzel, 1997) and may indicate that a question does not discriminate performance. The grading system that was used allows the number of participants who can be considered to have completed a task correctly (Grade A), partially (Grade B) and incorrectly (Grade C) to be examined. As no statistical difference was found between SW1 and SW2 groups on the PWE, the performance of both groups has been considered jointly. P-values have been calculated based on the number of Grade A's awarded:

- Task One: a p-value of 0.39 for SW1 and SW2 groups and 0.31 for participants with no self-taught experience
- Task Two: a p-value of 0.56 for SW1 and SW2 groups and 0.46 for participants with no self-taught experience
- Task Three: a p-value of 0.50 for SW1 and SW2 groups and 0.46 for participants with no self-taught experience
- Task Four: a p-value of 0.06 for SW1 and SW2 groups and 0.0 for participants with no self-taught experience

Rejection of p-values less than 0.20 and greater than 0.90 has been previously proposed (Escudero *et al.*, 2000). Other work suggests an alternative threshold of 0.30 and 0.85 (Sadesky & Pope, 2011). With the exception of Task Four, the p-values reported all fall within these ranges. This indicates that Tasks One to Three were not disproportionately easy and can be considered to discriminate performance. The low p-value of Task Four, however, suggests that the design of this question may be flawed or that other factors may have impacted on participants' responses. See Chapter Eight for a discussion of such issues.

7.6.2 Validity of the Programming Tests

As discussed in Chapter Four, the programming constructs introduced were based on those identified by the ACM/IEEE Joint Task Force. Table 7-26 and Table 7-27 demonstrate how the IWE1, IWE2 and PWE assessments relate to these concepts. All three exercises were designed to address the learning objectives identified by the ACM/IEEE. As a result, the condition of content validity is considered to have been satisfied as the exercises are of direct relevance to the topic under consideration. When examining these tables note there was overlap between many of the concepts and, in regards to each question, only the most pertinent programming fundamentals have been identified. The implications of using these exercises are discussed in Chapter Eight.

<i>Programming Fundamental</i>	In-Workshop Test One (IWE1)										In-Workshop Test Two (IWE2)									
	<i>Q1</i>	<i>Q2</i>	<i>Q3</i>	<i>Q4</i>	<i>Q5</i>	<i>Q6</i>	<i>Q7</i>	<i>Q8</i>	<i>Q9</i>	<i>Q10</i>	<i>Q1</i>	<i>Q2</i>	<i>Q3</i>	<i>Q4</i>	<i>Q5</i>	<i>Q6</i>	<i>Q7</i>	<i>Q8</i>	<i>Q9</i>	<i>Q10</i>
Basic syntax and semantics							✓						✓							
Variables		✓	✓	✓					✓			✓								
Data Types					✓								✓							
Expressions				✓		✓				✓		✓								
Assignment				✓								✓								
Conditional Structures						✓	✓		✓								✓	✓	✓	
Iterative Structures													✓	✓						
Functions (Methods)	✓															✓	✓			
Parameter Passing	✓																			
Structured Decomposition							✓											✓	✓	
Simple I/O										✓								✓	✓	

Table 7-26. Breakdown of the relationship between the in-workshop exercises and the programming fundamentals taught.

<i>Programming Fundamental</i>	<i>Task One</i>	<i>Task Two</i>	<i>Task Three</i>	<i>Task Four</i>
Basic syntax and semantics	✓	✓	✓	✓
Variables				✓
Data Types				✓
Expressions	✓	✓		
Assignment				✓
Conditional Structures	✓	✓		✓
Iterative Structures			✓	✓
Functions (Methods)				✓
Parameter Passing	✓	✓		
Structured Decomposition	✓	✓	✓	✓
Simple I/O	✓	✓		✓

Table 7-27. Breakdown of the relationship between the post-workshop exercise and the programming fundamentals taught.

7.7 Additional Data Source: Teacher Interviews

In this section a summary of interview data is presented. Full interview transcripts are available in Appendix A13. The teachers interviewed have been assigned codes T1, T2 and T3. T1 was familiar with student cohort SW2 while T2 and T3 were familiar with student cohort SW1. The T1 interview took place over the telephone while T2 and T3 were conducted in person. Two main themes were identified during the coding (see Figure 7-9). A third theme, ‘Miscellaneous’, was also established.

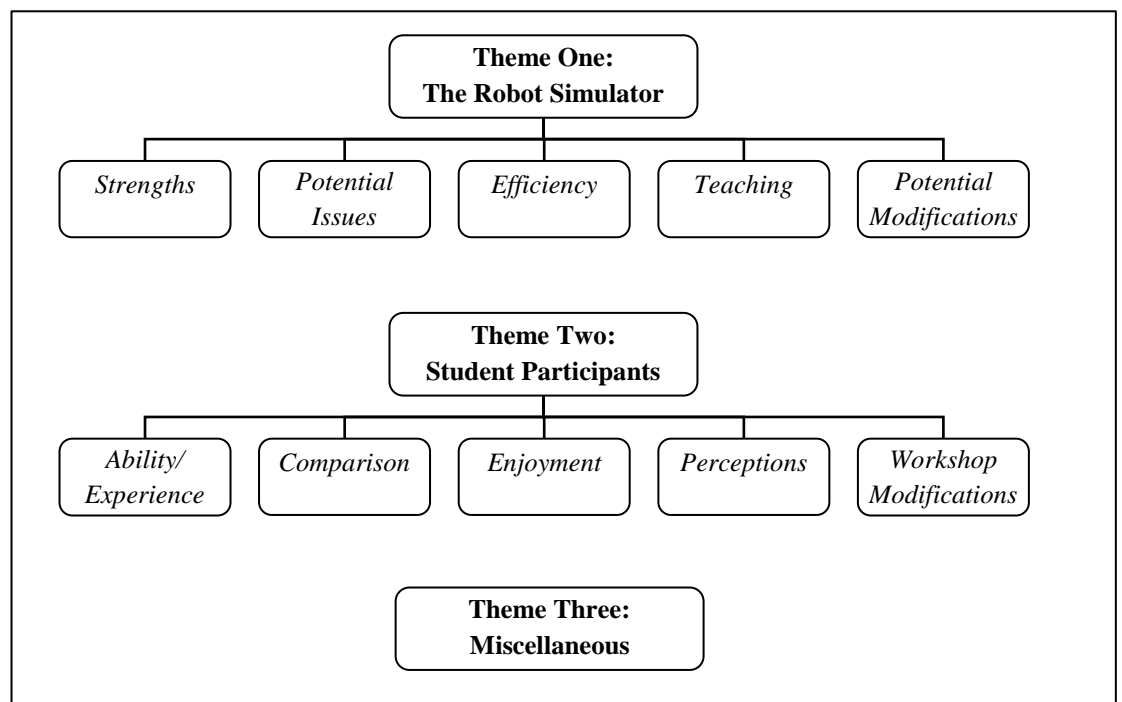


Figure 7-9. Themes identified during the teacher interviews.

Both T1 and T2 had some programming knowledge. T1 gained industrial experience before entering into a teaching career while T2 had a degree in computing. T3 had minimal computing experience and holds a Business Studies degree. All three interviewees were active teachers responsible for delivering either ICT or CS modules at the time of the study.

7.7.1 Theme One – The Kebot Robot Simulator

Strengths

All three teachers believed Kebot to be an effective introductory programming teaching tool that has a number of strengths. The nature of the simulator, in particular how it is visual and simple, were highlighted:

T1: The way it worked I think is a very good idea... (students) are not having to worry about the nitty gritty stuff, it just works... it's a much more effective way of doing it actually... they can see something happening as a result of what they are doing.

T2: I think it's great and it really puts things into a visual context so (students) can understand the idea of a robot moving, stopping and turning... you have to put it in a grounding that they are going to understand.

T3: ... I think it was great.

Potential Issues

Only T2 identified a potential issue with Kebot:

T2: ...because you have pre-written a lot of the methods it does make it look easier to them than it perhaps would be if they were starting from nothing as obviously a lot of the methods are just built in and they are just calling the methods... If you go onto explain the methods later, like you did in the session, then I think that (way) is fine so they get to see about what actually goes into it.

Efficiency

T1 believed Kebot offered a more efficient means of introducing programming compared to traditional methods. T1 stated they would use the simulator in their own lessons due to this:

T1: Since I was planning on pinching it and using it in September I would say definitely [response to being asked whether 'traditional' methods of teaching the same programming concepts would take longer]... if we were doing it the 'traditional way' you would be talking about double (the amount of

time), at least. In terms of efficiency if you were to run that workshop in that way and then follow that up for the ones who are struggling a bit we'd see a greater gain in productivity I think very quickly.

T2 also felt Kebot likely offers an efficient means of teaching programming, especially for lower ability students:

T2: It gives some of the concepts a more concrete outcome... I think the time would be saved by the fact that the concepts would probably sink in first time. You could do the same concepts and you could program a "Hello World!" or even just some straightforward maths, or whatever, in the same amount of time but you would probably have to go over it a couple of times without there being some kind of visual representation of what was going on... With a high ability group it probably wouldn't make an awful lot of difference (in terms of saving time). But with a medium to low ability group it probably makes a difference.

Teaching

T1 identifies the "what if scenario" as being an issue which may prevent some teachers from using educational software such as a robot simulator. T1 also stated how teachers would be capable of using Kebot in their lessons, provided they received some sort of briefing beforehand:

T1: There's always the "what if scenario"... if it goes wrong, what happens? But that happens with any sort of software... the fact that you can just restart (the simulator) takes away some of the problem. I would think that you could actually use (the simulator) with a variety of people because it's self-contained and as long as they have prepped up what is going to happen and they have got the answers there I don't think there would be too much of a difficulty.

T1: If (teachers) could work through the workshop, and they were confident with that, then they could take that programming on. I would be quite happy to do that and I think that others would be as well because you have got the security of "this is what happens" and as you can experiment with it and as it isn't going to go horribly, horribly wrong.

T2 discussed how, in regards to teachers using a robot simulator, limited programming knowledge and an inability to be creative with the software could be barriers:

T2: (The) majority of teachers in the UK who are teaching ICT are probably not from a programming background, the majority will be from something else and they will have moved into ICT. So you have got a subject knowledge barrier. And the second (issue is) setting the challenges. Some people would look at a robot and think it can (only) move forward, backwards and turn round... and can't think of anything else to do with it. So being creative (and for example) getting (the robot) to draw a spiral, to draw a figure of eight, to draw an n-sided polygon that's where I think people would struggle.

T3 describes how Kobot itself helps to break down anxieties about teaching programming:

T3: I've got a Business Studies degree but it's still something I am interested in. It's (about) breaking down the fear barrier for the others and I think (Kobot has) done that and it is doing that... I think it helps to break down the barriers of the unknown and the fear.

Simulator Modifications

T1 was of the opinion that in its current form Kobot required little modification:

T1: I don't think I would really do very much with the software. It works. I'd be inclined to leave it alone. I see a lot of stuff that is 'improved' but is not necessarily an improvement. The idea of (the robot simulator) is to be a tool to get kids thinking about designing and building something and it does that job. I'd be happy to use (the simulator) in a classroom without further modification as it does its job. I'd leave it alone.

T3 also shared a similar opinion:

T3: Perfect. It's perfect.

7.7.2 Theme Two – Student Participants

Ability/Experience

T1 remarked how their student cohort (SW2) were capable, motivated and had some prior introductory programming experience having previously encountered elements of Visual Basic:

T1: (The students) would be the sort of top end, the interested ones. They have done VB (before). Around 15-16 hours in the first term and then (some students) in the second term (in the form of projects).

Both T2 and T3 taught in the same High School/Sixth Form at the time of the interview. Both agreed that their student cohort were a mixed ability group:

T2: They were a mixed ability group so they are really typical... You are talking of students from Grade A right down to Grade E.

T3: ... they are not the brightest we have got, but they are the most hard working... they want to graft.

T2 discussed the issue of gender and it's potential impact on students' performance using Kebot:

T2: There was an err on the side of ability, hitting the goals was stronger on the boys side. A couple of the girls managed to do that as well. But I was quite impressed with the fact that the girls were having a go. They seemed to do well and seemed to generally enjoy it.

Generalisability

When asked to consider a typical student cohort, all of the teachers were broadly in agreement that a similar number of students would significantly struggle or not take much of an interest, regardless of the programming intervention being used:

T1: One or two (out of a group of 10) would find it difficult I would have thought.

T2: (Out of a group of ten) ... a couple very high fliers, the majority middle of the road, a couple would struggle and then you might find a couple who just can't do it at all and don't try.

T3: If we say an average group of 15 I would say about three would be probably disengaged.

Enjoyment

T1 believed that Kebot and workshop was well received and the experience enjoyed:

T1: I got a bit of flak from them saying, “Why can’t you do it like this!”. The fact that I’ve seen something, and that they have all been coming back talking about it saying positive things about it (is good)... They were very pleased with it. They were happy with it... As I have got two of the guys (who took part in the workshop) talking about doing this for a project I’d imagine yes (they did enjoy using the simulator)!

T3: All of them really, really enjoyed it... and when I spoke to them about it afterwards back in lesson they were all very positive.

T2 insinuated that whilst a small number of students may have found some aspects of their experience a little dry, on the whole the workshop was enjoyed:

T2: The majority seemed to enjoy it. As you would expect, probably one or two found it a bit dry but the vast majority seemed to enjoy it.

Perceptions

T1 suggested how a robot simulator could help to improve novice’s perceptions of subject, specifically because it enables students to picture real-world applications of programming:

T1: ... If you put it into a bigger context... then you can immediately see how this would be useful or whatever... It was a productive exercise and not a chore... here (the students) were actually actively engaged in doing something and would of kept on going whether we (the staff) were there or not I would have imagined. Which is really where they need to be, if they are actively engaged (that would be good) for OFSTED.

T2 was less certain whether the simulator helped to improve students perceptions of programming, although this may be in part to do with their past exposure to the subject:

T2: They wouldn't have come (into the session) thinking it was going to be boring because their only experience (of programming) so far would have been geared towards the exciting. So they will have done game making or possibly animation. They will have also designed websites using HTML. Most of them will have used raw HTML rather than something like Dream Weaver.

T3 commented that they felt Kebot improved students' perceptions of the subject.

Comparison

In regards to whether students would be more encouraged when taught using Kebot compared to traditional teaching methods, T3 believed that programming virtual robots was a positive:

T3: I think it definitely encourages, yes... What we think they like about your (workshop) is the fact that it is a robotic simulator and you can hook them in with robots and show them what robots can do.

Likewise, T1 also felt using a robot simulator helped to encourage students but that questions did remain about whether such an approach would be suitable for all students:

T1: I think they would (be more encouraged), although I think the difficulty is when they have not done anything like this before. You get this huge difference in pace from the ones who are struggling to type the thing up properly to the others who are just away, who have just understood it, and who are gone. With traditional mechanisms, (like) "Hello, World!" which is relatively simple, the moment you move beyond that that's when (the participants) start to spread out so some will move fairly slowly and some much more quickly and that creates little problems.

T2 felt it was too early to offer an opinion on the matter.

Workshop Modifications

All three teachers offered views on how the workshop may be modified in order to improve students' learning experience. T1 suggested how a summary booklet containing concepts introduced might be beneficial:

T1: ... what you really might want is something that summarises (each) particular concept so they have got something they can take away. Almost like a little summary booklet... Something which they can actually take away at the end (so participants) can refer back.

T1 commented how they felt the workshop was almost complete and just needed to be extended:

T1: You have got all of the right bits, I think, its just sort of putting it together into a longer running package like we (the college) would do. I am perfectly happy to experiment with that and see where that goes from our end.

T2 believed that some sort of hand-out would be beneficial:

T2: ... hand-outs would be useful because then they can always jot notes on as well. Whenever we use hand-outs we encourage the students to jot notes... I noticed in the session a lot of students were only using one ampersand (in their code) because, even though it's been explained verbally and has been on the slides, they haven't done anything with that. Even circling it (would help) otherwise it is gone.

T2 felt that it was important that when using a simulator that students attempted every programming command:

T2: From a teaching point of view you would probably want to get them to attempt every instruction. So rather than give them a concept verbally and then move onto another concept verbally and then say let's try them both you would want to say, "Here's one concept, give it a go". Even if it seems trivial, try it anyway so they have all had a go then move onto the next concept because students aren't very good at retaining what you have just told them unless they actually do it.

Finally, T2 believed that 'extension activities' were important and that they would help to further engage students with the robot simulator and the programming material:

T2: I think extension activities (are) the key. So, on each task... there will be some who take five minutes which is your allocated time (but) there will be some who do (a task) in one minute and won't know what to do next... If you say (to the students)... if you finish (a task) can you extend it... so the main task is covered by everybody and those who do it quickly have got the opportunity to just try a little bit of a modification. I would tag it on the end of your slides just at the bottom. What we call it in education is

extension activities... The idea is if you can't it's no great shame because you have done the main bit that we asked you to do. From an educational point of view that's very important because you just don't know (students) ability.

T3 offered a similar view, namely further activities for students who have finished a task:

T3: The first day (of the workshop) could have been a little bit pacier. They are used to pace. We are taught as teachers that there has to be pace. So if there are some that are sitting around because they have done what you need them to do but you are waiting for the other ones who aren't as quick to process you need to have something for the others. You need to say (to them) "they are still finishing that but why don't you try this?"

T3 also commented, however, that while the workshop was quite rigid in the early stages, they felt that it had to be due to the nature of the subject and tool being used:

T3: Probably it was quite prescriptive in terms of you need to put this in but it has to be at the beginning when they are learning something new.

7.7.3 Theme Three – Miscellaneous

Other points were noted during the three teacher interviews that have not been classified. These are considered in this sub-section.

Other Considerations

Both T1 and T3 suggested analysis of the gender of participants who use Kebot could be a potential source of valuable information:

T1: It would be worth looking at the responses from the gender perspective... because it has a big pay off when it gets to our end. We have 76 A-Level (Computing) students and three are girls. They make that choice somewhere in Year 10 (aged 14) and they don't see it again. Certainly, on that basis, I would look at that... Gender is the big thing for me... Even on an open evening, during a taster session, you don't get

many girls. They look at it and think “I am off!”. It’s very deeply engrained in Schools that Computing is not for (girls) or whatever.

T3: Gender is the one that sticks out.

T3 also suggested a range of other factors that may be useful to consider when analysing data:

T3: I don’t know if you have the data to do it but we have to look for is things like kids who are on free school meals, kids that are looked after, kids who have got English as a second language because their processing skills are different. Where they live (may be one)... Ages of (students), what term of the year (could be something to look at)... If you have got a child born on the 31st August they are 12 months younger than (everybody else).

T2, meanwhile, suggested that use of an approach frequently adopted in teaching could be helpful when considering the programming performance of participants:

T2: One of the useful things we use in teaching a lot is colour coding, traffic lighting. So if there was a way for instance of very visually flagging up, from a distance, that a student achieves a task, you know the screen goes green or something like that, you can immediately look around the room and go, “Right, excellent, everybody except four of them has managed to get that”... looking at it from a teaching point of view, (it would enable you) to get instant feedback.

Another consideration noted by T2 relates to the attitude of students in regards to programming, specifically that some may initially possess unrealistic ambitions:

T2: What they probably think (coming into the session), like everybody does is, “I am going to learn to program, great, I’ll be writing Angry Birds by the weekend and will get myself rich”.

7.8 Summary

Details of the execution and results of the second case, Case Two – ‘Students’, have been presented in this chapter. In total, 23 students aged 16 to 18 years old took part. All of these participants were enrolled on a Further Education (FE) course at the time of the workshop. Qualitative and quantitative data has been collected and reported. Data collected through pre- and post-workshop questionnaires, in-workshop observations and three programming exercises (used to assess programming performance) have been used to address the case study aims and propositions in Chapter Eight. An additional data source, specifically semi-structured interviews with three in-service teachers, has also been used. Each of these teachers was familiar with one of the student cohorts involved and this data had allowed for a greater insight. In the discussion section that follows the results reported in Chapter Six and Seven allow the effectiveness of the Kobot robot simulator, as a tool to introduce programming concepts to novices, to be evaluated. Recommendations for future research are also provided.

Chapter Eight

Discussion

In this chapter the results generated from the multiple sources of data used during the case study are brought together. Data reported in Chapter Six (Trainee Teachers) and Chapter Seven (Students) are used to address four propositions and answer the thesis research question. Evidence suggests that a robot simulator is an effective tool for supporting the learning of introductory programming concepts. This conclusion must be considered with caution, however, as a number of factors should be taken into account when interpreting results. The flexibility of the case study methodology has helped to facilitate the collection and interpretation of a large quantity of qualitative and quantitative data. Steps taken to ensure the rigour of the case study are provided. The research findings are examined critically and compared with previous work. Threats to validity and rival explanations are also discussed. Recommendations based on the lessons learned are presented as are suggestions for future research. It is intended that these will help to guide the future development and use of robot simulators in an introductory programming context.

8.1 Introduction

The overall aim of this thesis was to investigate whether simulated robots are effective tools to support the learning of introductory programming. To achieve this aim a Systematic Literature Review (SLR) was completed and exploratory empirical research was conducted. These activities provide the justification for this work. The Kebot robot simulator and an associated 10 hour programming workshop were then developed. Student programmers, in addition to pre- and in-service high school teachers, have taken part in empirical research designed to evaluate the use of Kebot. The case study methodology has been used and the following research question was asked:

Is a robot simulator an effective tool for supporting the learning of introductory programming?

Effectiveness has been determined after considering participants' opinions, attitudes and motivation using a robot simulator in addition to an analysis of students' programming performance. In Chapter One it was established that a simulator would be considered effective if all of the following are satisfied:

- 1) Participants enjoy learning programming in such a manner,
- 2) Participants believe the approach to be valuable and useful,
- 3) Participants successfully learn introductory programming concepts.

Analysis of collected data indicates that all three of these factors can be considered satisfied and that this demonstrates that Kebot is an effective tool for supporting the learning of introductory programming. In this chapter the four case study propositions are considered:

P1 A robot simulator is an effective tool for supporting the learning of introductory programming

P2 A robot simulator improves novices' perceptions of programming

P3 A robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience

P4 A robot simulator improves trainee ICT/Computer Science (CS) teachers' confidence in their ability to teach introductory programming

A triangulation strategy is used and this helps to ensure reliable conclusions (Runeson *et al.*, 2012). A chain of evidence has also been maintained as this ensures reliable and traceable case study research (Verner *et al.*, 2009). In Figure 8-1 the chain of evidence, based on one discussed by Yin (Yin, 2009), can be seen:

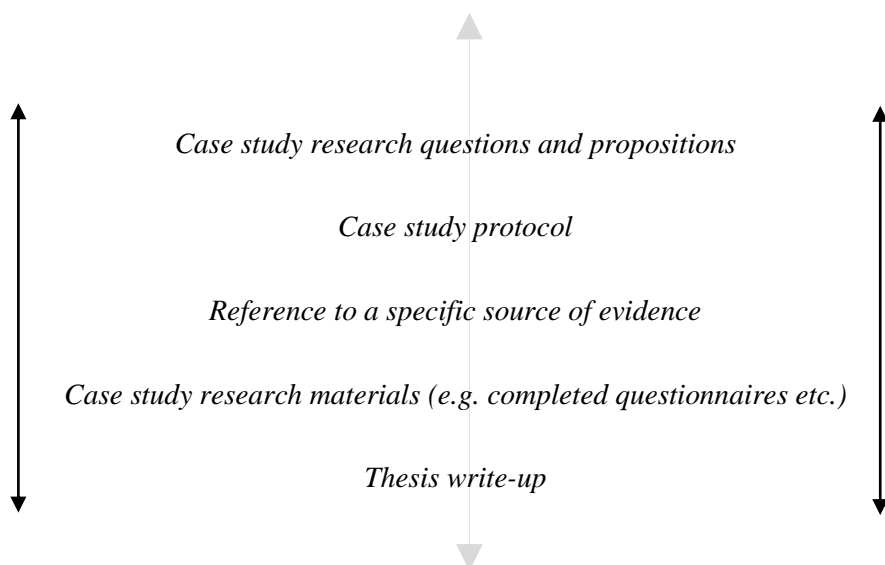


Figure 8-1. The case study chain of evidence.

The chain of evidence involves: the final report making reference to relevant materials (e.g. specific documents, interviews or observation data); evidence being accessible and data collection procedures described (e.g. in appendices); circumstances being consistent with procedures outlined in the protocol; a link to the research question being maintained throughout. As is represented by the arrowheads in the diagram above, the chain of evidence is a two-way process that allows a link between the initial research question/propositions and final write-up to be established. In Chapter Five other validation activities are outlined and a case study design checklist, suggested by previous research (Runeson & Höst, 2009), is provided in Appendix A15. Finally, in Section 8.6, rival explanations and threats to validity are analysed.

8.2 The Case Study Propositions

In this section the four case study propositions are considered. For each, the three main sources of data (student workshops, trainee teacher workshops and in-service teacher interviews) have been drawn on. In-workshop observations have also been used.

8.2.1 Proposition One: *A robot simulator is an effective tool for supporting the learning of introductory programming*

Student Participants

Post-workshop questionnaire results indicate that SW1 and SW2 participants believed Kebot to be an effective introductory programming teaching tool. No participant stated the simulator was ineffective, only one of the 18 students did not enjoy their experience using the software while it also scored highly when rated on a five-point scale. The visual nature of Kebot and the fact that robots are engaging were among the attributes liked. This was in addition to the simulator being accessible, interesting and innovative.

An underlying assumption of this research was that if the robot simulator could be shown to motivate participants then this would offer some evidence of effectiveness. This is because there is a link between learning and enjoyment (Lumby, 2011). Increased enjoyment also increases motivation which enhances levels of learner effort, persistence, performance and cognitive processing (Jerez *et al.*, 2012; Ring *et al.*, 2008). Approaches to programming that are “fun” are also likely to be more effective (Fagin *et al.*, 2001). Almost all students viewing Kebot positively (as only one of the 18 students completing the workshop believed their programming experience was not enjoyable while no students believed the simulator was not effective), therefore, is considered to be partially indicative of effectiveness. This is supported by the fact that both groups of students identified a greater number of positive aspects than negative ones when asked to identify what they did and did not like about the simulator (when the responses of both groups are combined 41 positive, and 18 negative, aspects were outlined). To establish the effectiveness of

programming interventions the effect on learner attitude and confidence can also be considered (Moskal *et al.*, 2004) in addition to whether an appetite to learn further programming skills is demonstrated (Cliburn, 2006b). As no student stated that they would not consider learning to program in their own time after the workshop this, at the very least, indicates that the use of Kebot did not dissuade students from the prospect of studying programming. This is in contrast to the “tedious and dull” traditional method of introducing programming that involves the production of calculator-type programs that provide simple numeric output (Guzdial & Soloway, 2002).

It is known that the tried and tested method of teaching programming can be boring. Exercises involving manipulations of student marks, stock levels or bank account details are unlikely to excite novices (Jenkins, 2002). This is especially true for first-time learners. The fact that robots have a visual representation, and that output is more entertaining than textual output, is an advantage of simulated robots (Becker, 2001). Furthermore, learning is improved and better enjoyed when students work on tasks that interest them (Jenkins, 2002). Data suggests this was the case during SW1 and SW2. When first learning to program, a significant amount of effort is required for very modest return (Perkins *et al.*, 1988). Use of the robot simulator was believed to have allowed students to quickly elicit interesting behaviour from basic programs. Due to all these factors, and as demonstrated in the post-workshop questionnaire responses, Kebot was found to interest and engage participants and it is fair to assume that this can improve levels of learner effort and performance (Jerez *et al.*, 2012; Ring *et al.*, 2008). As collected evidence indicates that a robot simulator is interesting and enjoyable to use, this helps to address a common issue that teachers and designers of educational technology face – namely, how can students be sufficiently engaged and motivated (du Boulay, 2000). The simulator could also enable ‘unintended’ learning (where programming skills are unwittingly acquired through ‘tinkering’ and ‘play’) as some students may focus on the activities completed and not on the task of learning programming (Petre & Blackwell, 2007).

If the effectiveness of Kebot has partially been determined by examining learner attitudes and motivation, the impact upon programming performance must likewise be considered. Three

separate exercises were used to assess programming performance. As determined by the pre-workshop questionnaire, the programming backgrounds of students were varied. Two SW1 participants had previous self-taught experience while nine had not encountered programming prior to the workshop. The SW1 group was described in the associated teacher interviews as being typical and of mixed ability. SW2 participants, however, were described as motivated and had been introduced to rudimentary elements of Visual Basic several months before. For six participants this was their only exposure to programming while four had attempted to learn programming by themselves. To investigate differences between SW1 and SW2 statistical techniques have been used.

For the in-workshop exercises it was decided that a mean score greater than six (out of 10) would demonstrate an acceptable level of performance and indicate learning. This score was used because it was considered unlikely that it could be achieved by guesswork. UK high school, FE and HE institutes traditionally adopt a pass threshold of 40% which is below this threshold. Existing assessments (specifically past exams used on the Programming I First Year module at Keele University) were considered for use during the study. While such instruments influenced the design and layout of the programming exercises later developed, it was not possible to use these as they were designed to assess the performance of students who had completed an undergraduate module (involving 22 hours of lectures, 22 hours of practical sessions, four one-hour tutorial classes, 36 hours practical/tutorial preparation and 66 hours of private study) and not a two-day workshop.

In regards to In-Workshop Exercise One (IWE1), most participants performed well with scores of six or above (although two SW1 participants were awarded scores below six). A statistical difference between groups was, however, found with SW2 participants performing significantly better. Potentially these participants may have adapted their previous, if limited, knowledge of Visual Basic to fit the tasks set during the early part of the workshop. The group was also more likely to be familiar with computing technology given the course on which SW2 participants were enrolled. As such, less time may have been expended understanding rudimentary aspects of programming (e.g. how to compile a program), and more time may have been spent focusing on the

workshop material. In relation to In-Workshop Exercise Two (IWE2), overall scores were lower than on IWE1. A decrease in scores was expected due to the increased complexity of the test and concepts assessed. Unlike IWE1 there was no statistically significant difference between groups. This may provide evidence that any advantage that SW2 participants had due to increased familiarity with programming and/or computer technology was reduced during the workshop.

The performance of SW1 participants on both IWE1 and IWE2 is broadly in line with what Teacher Two (T2) stated would be typical, "... a couple of very high fliers, the majority middle of the road, a couple would struggle". Of the nine SW1 participants completing IWE1 and IWE2, one scored 19, five scored between 12 and 18 and three received an overall score of 12 or less. The same is true for the eight SW2 participants as one scored 19, five scored between 12 and 18 and two received an overall score of 12 or less. It is also interesting to note that the lowest scores for both groups were observed on questions relating to variables, expressions and assignment (Q4 on IWE1 and Q2 on IWE2). Whilst it was not the intention of this work to investigate which concepts a robot simulator helped students to learn, this is an area that future research should further explore.

The Post-Workshop Programming Exercises (PWE) was devised to determine whether deep learning has occurred. The Page robot was introduced during these tasks and this has different abilities to the other robots encountered. Participants had no option but to adapt their knowledge if they were to successfully complete each of the four tasks. This is because code from exercises completed earlier in the workshop could not be simply copied due to Page, and associated control methods, differing. For both groups, performance on Tasks One, Two and Three of the PWE is judged to demonstrate learning. As previously outlined in Chapter Seven, the p-values for these tasks all fall within an acceptable range and this indicates that the exercises were not disproportionately easy and do discriminate performance. Tasks One and Two assessed several programming concepts (including basic syntax, expressions, conditional structures, parameter passing, structured decomposition and simple I/O) while it was intended that learners would use an iterative construct when devising a solution to Task Three.

Performance on Task Four, however, differed considerably. Task Four was the most substantial PWE challenge and involved participants using a number of programming concepts. From discussions with participants, which were confirmed by the in-workshop observations, the nature of the exercise itself (i.e. it being poorly designed) is not considered to be responsible for the high number of low grades awarded. Several SW1 participants commented that they were content to have attempted (and in most cases engineered solutions to) Tasks One, Two and Three and did not feel inclined to tackle the final challenge. Analysis of code supports this view as only three SW1 participants made a meaningful attempt to solve the problem. Some participants also appeared less willing to work when the first of their peers left the laboratory having submitted their solutions. Most SW2 participants, despite attempting the task to some extent, only devised a partial solution (which restricted the maximum awardable grade to a B). Several participants commented that they were satisfied to submit this basic attempt without modification. No statistical significant difference was observed between SW1 and SW2 participants on any of the PWE tasks.

The timing of the post-workshop exercises, in addition to how long they took to complete, is considered to be potentially responsible for performance on Task Four. This is because the PWE was administered at the very end of the second workshop day when fatigue may have been an issue. The fact that students were not used to such intensive sessions, with several assessments, over a short period of time may have been an issue. An alternative explanation may be because participants were under no obligation to complete the tasks. As the assessments were not part of any formal assessment there is no guarantee that effort on the final (or indeed any) task was to the best of participants' ability. As Task Four constituted the most substantial programming challenge, and less pseudo code was used to assist participants compared to the other tasks, this could indicate how the workshop failed to adequately prepare participants. This may be supported by the fact that on the post-workshop questionnaire only two participants responded that the tasks completed during the workshop were difficult. All of these factors mean it is unclear whether it was the design of the task, the nature of the task or other factors that were responsible for the performance observed on Task Four. As it will be outlined shortly, however, as a cohort of trainee teachers all

developed a solution to Task Four indicates that factors other than design may have been responsible for the students' performance.

It was appreciated from the outset that the nature of the programming exercises (i.e. because they were developed for the purposes of this work and as the reliability of these instruments has not been independently verified) meant that it would not be possible to make strong claims about the simulator's effectiveness based on these alone. Indeed, it was appreciated that other sources of evidence would have to be considered when formulating conclusions and this is one reason why a case study was undertaken. Bearing these cautionary points in mind, however, the programming exercises are considered to show:

- For IWE1, that the performance of both SW1 and SW2 groups demonstrates a good level of learning
- For IWE2, that the performance of both SW1 and SW2 groups demonstrates an acceptable level of student learning given the increased difficulty of the exercise compared to IWE1
- For the majority of the PWE (specifically Tasks One, Two and Three), that the performance of both SW1 and SW2 groups demonstrates that participants were able to complete satisfactorily several programming challenges unassisted by the end of the workshop. Moreover, a discernible difference cannot be noted when the performance of participants with no self-taught experience is considered separately from those with prior programming experience. As it is fair to assume that participants with self-taught experience may be more motivated than those without, the fact that that PWE data for these students is similar offers additional evidence that the approach was effective. Participant performance on Task Four, however, does raise questions and caution should be used when drawing conclusions from this data.

Trainee Teachers

All of the trainees had some degree of programming knowledge, classroom teaching experience and pedagogic knowledge as a result of their PGCE course. The pre-workshop questionnaire results demonstrate how many trainees had previously used the concepts that were introduced during the workshop and some had spent a considerable amount of time studying programming. As with the students, the overwhelming majority of trainees believe Kebot offers an effective and enjoyable means of introducing basic programming to novices. Of the 21 trainees who completed the workshop none responded that they would not use Kebot in their own future lessons. Aspects of Kebot that were liked related mainly to the visual and interactive nature of simulated robots and the accessibility of the approach. Disliked factors included issues with the visual appearance of Kebot (specifically that only a top down perspective is offered). The fact that positive points outnumbered negative ones by a ratio of around two to one further highlights that the approach was viewed positively. Several participants requested a copy of Kebot and the workshop presentation for their own future use. The author also received emails, from several trainees in the weeks following the sessions, asking questions about the robot simulator and workshop.

Whilst programming exercises were used during the trainees' workshops, these instruments were not intended to assess learning or performance. It was assumed prior to the workshop that the programming exercises would offer little challenge to the trainees due to all having some programming experience. The trainee teacher workshops, therefore, presented an opportunity to create, test and validate programming exercises in advance of the sessions involving students. It was surprising that a number of TTW1 participants struggled with the programming exercises considering that 13 had previously used Java concepts including arrays, variables, iteration and selection in their past code. All of these concepts were also used during the workshop. The number of participants in the group (as 17 trainees took part – the largest group involved in the case study – the workshop leader may not have been in a position to offer adequate support to individuals who were struggling), and recognised problems with the exercises, may have contributed to this. Significant changes were made to all of the exercises because of the issues identified during TTW1.

These revised instruments were used during TTW2 and the programming performance of TTW2 participants was more in line with what was expected before the workshop. The performance of TTW2 trainees on the revised programming exercises, which were also used during both student workshops, indicates that PWE Task Four is not poorly designed and that other factors (which have been previously outlined) may have been responsible for the performance of students on this exercise. This is because all TTW2 participants engineered a solution to PWE Task Four.

In-Service Teachers

Three interviews were held with current teachers to determine whether they believed the simulator to be effective. Responses indicate that this was the case. The accessibility of the approach and the appeal of robots were highlighted as positives. The simplicity of the simulator, and the fact that it was ready for instant use, were also commented on. All teachers also believed that the simulator offers an enjoyable means of learning. As already discussed, this suggests that the simulator is effective for supporting the learning of programming as this can motivate learning. Similarly to the student cohort, few issues with the simulator were identified. One teacher suggested that they would be comfortable using Kebot without modification. Caution was raised by another, however, who warned that it is important not to make the task of programming appear too easy by hiding too much from the user. All interviewees felt that the nature of the simulator and how it breaks down barriers which may be encountered when teaching programming demonstrates effectiveness. Whilst it is not feasible to devise a learning environment to suit all students (Jenkins & Davy, 2008), interview data demonstrates a consensus that use of a robot simulator is suitable for a broad range of learners.

8.2.2 Proposition Two: *A robot simulator improves novices' perceptions of programming*

Student Participants

Responses to the pre-workshop questionnaire suggest that students thought highly of programming and were motivated to learn about it. This is because almost all stated they would consider learning programming and as few issues or stereotypes were identified (indeed, none of the SW1 participants recorded any issues or stereotypes they associated with programming). In addition, most believed it important to teach programming in high schools while perceptions were good among those who had previously programmed. It should be noted that due to the age of students few are likely to have been involved in a research study before. Given that the pre-workshop questionnaire was the first task completed during the workshop, there is a possibility that respondents may have exaggerated in their replies or been reluctant to answer with complete honesty. However, after examining post-workshop questionnaire responses, this is considered unlikely. This is because a large proportion of participants stated that they were unsure whether the simulator helped to dispel, or that they didn't know of any, programming stereotypes.

In regards to whether participants would consider learning programming in their own time, only a minor variation between pre- and post-workshop responses can be noted. This can be interpreted as a slight negative change as four participants who before the workshop stated they would consider learning programming later said that they were unsure (although it should be stressed that no one responded that they would not consider learning to program). The reasons for this variation are unknown but it is not considered substantial enough to suggest that the simulator had an adverse effect on perceptions, especially considering the majority of other qualitative data was positive.

Despite these points, however, the students themselves believed Kebot did help to improve their perceptions of programming. Indeed, fourteen Case Two participants believed their perceptions of the subject were improved while only two did not. It was established in Chapter One how research has partially determined effectiveness of educational programming interventions by evaluating the

ability to improve learner attitudes (Moskal *et al.*, 2004). The fact that a large proportion of students responded that the simulator improved their perceptions is considered, therefore, to be additional evidence of effectiveness.

In-Service Teachers

One teacher discussed how newcomers to programming would probably expect a lot from their early programming interactions, as any previous exposure to the subject would have likely been geared towards the exciting, and that it would be a challenge to manage these expectations. One of the strengths of using a simulator is that it enables learners to elicit interesting visual behaviour from relatively simple programs. The other educators interviewed commented that they felt the software helped to improve perceptions, and encouraged learning, due to it relating programming to a real-world application. The way in which the simulator engages learners was also mentioned as being a positive factor in terms of ensuring positive school inspection reports.

8.2.3 Proposition Three: *A robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience*

Student Participants

The effectiveness of other programming interventions have been determined by assessing learners' preferences, specifically whether they prefer the investigated approach over alternatives (Cliburn, 2006b). All SW2 participants had been introduced to fundamentals aspects of Visual Basic six months before their workshop. During discussions with SW2 participants and their teacher it was established that this was very much in the 'traditional' mould of learning programming (i.e. a text-based approach to general syntax, statements etc). This introduction to Visual Basic lasted for a similar amount of time as the Kobot workshop, and this has allowed participants' experience to be contrasted to the one using the simulator. Whilst it is important to remember that the number of SW2 participants was relatively small (*n.* 8), six believed their previous introduction to

programming was less effective than using the simulator. As this experience was self-taught and/or limited, however, strong conclusions should not be drawn from this.

The results of the research suggest that, when compared to ‘traditional’ methods of teaching programming, a robot simulator may improve learning by enhancing students’ motivation and interest. Open-responses from students on the post-workshop questionnaire indicate that Kebot was interesting and easy to work with in addition to being fun and enjoyable. Because of these factors it is believed that a robot simulator offers advantages over the ‘traditional’ method of teaching programming. This is due to such approaches often being dull and as a robot simulator allows early tasks to offer more than just text-based output. The use of a robot simulator also helps to ensure that knowledge is actively constructed as the risk of ‘passive’ learning is minimised.

Trainee Teachers

The trainees had all learned programming before the workshops. This allows a unique measure of effectiveness to be established as participants’ prior learning experience can be compared to the one using Kebot. Asking trainees to compare their previous learning experience to the one using Kebot allows a baseline of sorts to be established. The expert review of the case study protocol highlighted the potential benefit of such an approach.

For TTW1, only one of 17 participants believed that their previous introduction to programming was more effective than the one using Kebot. As the majority of these participants completed the Programming I undergraduate module offered by Keele University as a pre-requisite for admission onto their PGCE course (*n.* 13), this indicates how the robot simulator offers a more effective introduction to programming than conventional means of teaching (as is the case with Programming I). As the recommended coverage time of the Programming I module is 150 hours, this may also indicate that a robot simulator offers a more efficient means of introducing the subject. Efficiency of the approach, however, was not one of the focuses of this work. For TTW2, when participants were asked the same question, opinions were slightly more mixed. This is

because two trainees responded that their previous introductory programming experience was more effective than that using Kobot (although the three other trainees did not believe this to be the case).

In-Service Teachers

The interviewed teachers mostly agreed that a robot simulator offers a more effective means of introducing the subject compared to more widely used alternatives (as was the case with the majority of students and trainee teachers who had pre-workshop programming experience). How a robot simulator provides a “hook”, which serves to entice novices, appears to be responsible for these observations. One of the teachers interviewed also commented that they were planning on using Kobot in their own lessons the following academic year as they believed the approach would save double the amount of time compared to standard methods of introducing programming.

8.2.4 Proposition Four: *A robot simulator improves trainee ICT/Computer Science (CS) teachers’ confidence in their ability to teach introductory programming*

Trainee Teachers

The trainees were asked before and after the workshop how confident they would be using their knowledge to teach introductory programming concepts to high school students. The majority of TTW1 participants responded prior to the workshop that they would be confident. Following the workshop, however, the opinions of a number changed and more were unsure. This increased uncertainty may be due to a realisation that introductory programming concepts can be difficult to understand, even using a tool such as a robot simulator (which is supported by the fact that a number of TTW1 participants struggled at points). It is unclear whether this was the case, however, as responses on whether it would be difficult to teach programming remained unchanged.

Feedback provided by the TTW2 group differed slightly and was more in-line with what was anticipated. This is because participants’ confidence teaching programming was found to increase.

It was expected that this would be the case as it was thought that exposure to the robot simulator would highlight to trainees how they may be able to use such a tool themselves. This change in confidence was only slight, however, as only one participant reported an increase. Perceptions of perceived difficulty teaching programming likewise changed with two participants responding that such a task would be easy following the workshop. The smaller number of participants who took part in TTW2 means that it is not possible to draw strong conclusions from this however.

In-Service Teachers

The way in which a robot simulator may break down anxieties teachers have, due to it helping overcome the “what if scenario” of something going wrong when teaching programming, was highlighted as a positive by two of the teachers interviewed. This is because a robotic simulation environment can offer a self-contained teaching solution. Whilst not specifically stated as such, it is reasonable to assume that such a factor could help to improve the confidence of educators in their ability to teach programming.

8.3 General Discussion of Findings

In the remainder of this chapter the findings of the research are further considered and other factors discussed including: a comparison of findings to those of previously completed work; consideration of contradicting evidence and unexpected findings; reflection on the research that has been performed; recommendations and suggestions for future work. Following this rival explanations and threats to validity are outlined.

8.3.1 Comparison of the research findings to previously completed work

A critical examination of the research findings, in the context of the previous state of the subject, has been undertaken. The SLR established the potential to investigate the effectiveness of simulated robots. A limited number of studies ($n = 7$) were found and this body of research was considered to be incomplete because the quality of identified studies was judged to be poor. This is due to a large percentage of identified studies lacking vital research features and inadequately reporting results. The supplementary search of the literature, completed two and a half years after the SLR, also confirmed the conclusions of the SLR as does other recently published work (Alemany & Cervera, 2012, Flot *et al.*, 2012).

During the SLR it was observed that a number of studies were related to Karel. It has been discussed how Karel adopt robots as a metaphor and is more akin to an on-screen cursor which can only face in one of four directions and move one ‘block’ at a time (Untch, 1990). The research reported in this thesis advances knowledge as a simulator has been modelled on, and replicates the behaviour of, a ‘true’ real-world robot (i.e. one that has unrestricted movement and rotational freedom). This is not to say, however, that Kebot is ‘better’ than Karel but only that Kebot offers a different approach than other simulators that have been used previously. In addition, whilst the quality of existing studies related to the use of simulated robots was judged to be poor, it is interesting to note that none reported simulated robots to be ineffective (and when only the studies awarded a quality score of five or greater are included all report simulated robots to be effective).

The findings presented in this thesis are considered to support those of previous studies. The fact that the research reported does not challenge the existing perspective that simulated robots are effective is considered to partially validate the research completed. This does not mean, however, that this work is anything other than extremely valuable. The SLR established a significant need to investigate the use of simulated robots, and this research has helped to address a gap in knowledge.

The Robot Virtual Worlds (RVW) project taking place at Carnegie Mellon University (CMU), and involving the University of Pittsburgh (PITT), was identified during the latter stages of this research. RVW aims to determine whether students learn programming better using simulated, rather than physical, robots. The SLR reported in Chapter Two has been cited in a magazine article reporting the activities of CMU and PITT (Flot *et al.*, 2012). Early results released by the RVW team indicate how both physical and simulated robots aid students' learning of programming concepts (Liu *et al.*, 2013). In addition, it is reported that simulated robots may allow faster and more efficient learning. Whilst the scope of this thesis and that of the RVW project are different, the preliminary findings disseminated by the RVW team indicate how a simulator can support the learning of introductory programming. This further corroborates the outcome of this thesis that a robot simulator is an effective introductory programming learning tool.

Whilst Kebot is not considered to offer a game-based approach in the traditional sense, parallels can be drawn between robot simulators and computer games (Alemany & Cervera, 2012). Like games, Kebot provides visual feedback (as users can see when a particular strategy for solving a task is working) and as a variety of challenges can be completed using the tool (Brooks-Young, 2010). Before introducing technology which shares similarities to computer games, how long the approach can be used before it gets boring, in addition to learner access to resources, should be considered (Klopfer *et al.*, 2009). The fact that Kebot received positive feedback following 10 hours of use demonstrates how user interest was maintained during the workshops. The nature of simulated software (i.e. how it can be installed on multiple machines and thus be used by multiple learners simultaneously) also helps to overcome limitations often associated with physical technology (such as restricted resources). James Paul Gee (Gee, 2003) describes a comprehensive

set of principles, collated after analysing the association between video games and learning, which can be used to further demonstrate the effectiveness of a robot simulator environment for supporting the introduction of programming concepts (see Table 8-1). The principles identified are all considered to have been satisfied after drawing on the experience using Kebot.

Learning Principle	Comments
Active, Critical Learning The environment should be set up to encourage active, not passive, learning	<i>The nature of a robot simulator ensures users cannot be “passive”. This is because learning material is delivered through interactive activities and not traditional lectures etc.</i>
Psychosocial Moratorium Learners can take risks in a space where real-world consequences are lowered	<i>Unlike physical robots (with issues such as mechanical failure), a simulator offers a safe environment for experimentation as the software can easily be reloaded if difficulties are encountered</i>
Self-knowledge Learners should not only learn about the domain but also about their current and potential capabilities	<i>A robot simulator not only facilitates the learning of new programming concepts but may also encourage interest to study programming further (as demonstrated in post-workshop questionnaire responses)</i>
Amplification of Input For a little input, learners get a lot of output	<i>Interesting visual behaviour can be elicited from agents using simple programming statements. Participant questionnaire responses demonstrate the importance of visual feedback</i>
Practice Learners spend a lot of time practicing in an environment that it is not boring	<i>Feedback demonstrates how the approach is interesting and that almost all participants enjoyed using the simulator to complete programming challenges</i>
Multiple Routes There are multiple ways to make progress	<i>The nature of robots, in addition to programming itself, ensures that most challenges completed using a simulator can be solved in a number of ways</i>
Multimodal Meaning and knowledge are built up through various interactions and not just text	<i>The text-based component of a simulator is solely for inputting code. The simulated environment itself is visual in nature and allows for interaction with a variety of objects</i>
Subset From the start, learning takes place in a (simplified) subset of the real domain	<i>The fact that the Kebot simulator is modelled on a physical robot, opposed to simpler interventions that use a grid-based world, ensures learning takes place in a manner that is comparable to a ‘real-world’ experience</i>
Incremental Learning situations are ordered in the early stages so that earlier cases lead to generalisations that are fruitful for later cases	<i>A simulator can support scaffolded learning and this is actively encouraged due to the nature of the approach. This is because each programming concept can be broken down into smaller elements before a solution is attempted using the software</i>
Discovery There is ample opportunity for experimentation and discovery	<i>A robot simulator encourages experimentation as users are motivated to solve and/or perfect their solutions to challenges. Tasks can be solved in multiple ways</i>

Table 8-1. Relevant learning principles identified by Gee (Gee, 2003) which can be used to further demonstrate the effectiveness of a robot simulator to support the learning of programming.

Finally, the use of a robot simulator to support the learning of introductory programming concepts is considered to fit well with the constructivism learning theory. This is because a robot simulator allows “learning-by-doing”. Whilst participants received advice and guidance during the workshop, this was not in the form of direct instruction. Instead, participants actively constructed knowledge through their interactions and experiences completing tasks (Papert, 1980; Piaget, 1967). As links between constructivism and robotics (Alimisis *et al.*, 2007), programming (Wulf, 2005) and CS-education in general (Ben-Ari, 2001) have been previously established, it is not unreasonable to suggest that a robot simulator is a constructivist tool. Comparisons can also be drawn between a robot simulator and enquiry/problem-based learning theories as participants, through their interactions with Kobot, are considered to have taken an active role in their acquisition of knowledge (Drummond, 2009).

8.3.2 Consideration of contradictory evidence and unexpected findings

In this sub-section evidence that contradicts the conclusion that a robot simulator is an effective tool is discussed. Each source of data has been analysed to determine if any evidence supports the null hypothesis that using simulated robots is not an effective way of teaching programming. Unexpected findings have also been outlined.

Collected questionnaire data does not suggest that the robot simulator was ineffective. Negative feedback was received but this related mainly to the nature of programming (e.g. understanding Java syntax rules) or software limitations (such as only been able to use one robot at a time). Kobot, the programming support offered by the workshop leader and the associated workshop presentation each gained an overall score greater than four (where a score of one represents not at all effective and a score of five extremely effective) when all responses are considered. Additionally it was unexpected that participants’ views of programming before the workshop would be mostly positive. Given that a number of negative stereotypes are normally associated with the subject, it was not expected that this would be the case. The fact that the majority of participants believe programming should be an important part of the National Curriculum is also interesting

considering that, several months after the conclusion of the workshops, it was announced that computing (including programming) would become a compulsory subject that must be taught in schools from 2014¹. The fact that the confidence of TTW1 trainees' teaching programming did not improve after the workshop was also a surprise. Whilst evidence is limited, this may indicate that Kebot had a negative impact upon trainees' confidence. Potentially, this may be due to a realisation by some participants that they were not yet equipped to teach programming themselves.

The most significant anomaly on the programming exercises was on Task Four of the PWE. Indeed, the performance of both student cohorts (in particular SW1) is considered to be poor. It is unclear why this was the case and whether the design of the instruments was responsible. It is not considered that performance on Task Four indicates that the simulator is ineffective. As already discussed, however, Task Four offered the most substantial post-workshop challenge and students' performance could indicate that they were ill-equipped to tackle a problem which used several programming concepts. Participant performance on Task Four cannot support the overall conclusion that a robot simulator is effective. Several other sources of data suggest, however, that Kebot effectively supported the learning of programming concepts while performance on the other post-workshop programming tasks is considered to be good.

Finally, the teacher interviews and in-workshop observations offer little evidence that the simulator was ineffective. Whilst suggested improvements were made, and these have influenced recommendations for future research, feedback provided was largely positive. It was interesting to note that the teachers interviewed believed that a simulator could help to break down the anxieties that many educators might have in regards to using such software. This is because, in advance of the workshops, it was thought that the simulator may be viewed as adding on an additional layer of complexity to the already difficult task of teaching programming (although this does not appear to have been the case).

¹ Coughlan, S. (4th February 2013). Computer science part of English Baccalaureate. *BBC News*. <http://www.bbc.co.uk/news/education-21261442> (Accessed 9th October 2013)

8.4 Reflection on the Research Undertaken

In the first part of this sub-section the researcher's experience performing a SLR and case study is discussed. Following this further reflection on the work performed is provided.

Experience using the SLR methodology

Woodall and Brereton (Woodall & Brereton, 2006) considered the experiences of a PhD student when conducting a SLR and concluded that, "*Conducting a systematic review is a time consuming process... (and) that there is a need for a lighter systematic review process... which addresses the needs of a single researcher*". It is possible to agree with this view as the SLR presented took around nine months to complete. Such a period of time has been reported as being average for PhD students (Riaz *et al.*, 2010). As the full SLR guidelines (Kitchenham and Charters, 2007) were followed, however, it is suggested that a 'lighter' systematic review process is not a necessity for every PhD student and that a full-scale review can be completed by a relative newcomer to research. Several observations have been made:

- A thorough understanding of the nature and scale of the task is required before undertaking a SLR. Meetings with a senior researcher (e.g. a PhD supervisor) help as they ensure the study stays on track and is completed within a pre-agreed timeframe
- Woodall and Brereton's (2006) advice to adopt a stepwise refinement approach when developing the protocol helps as it allows advice to be gained from a more experienced researcher while ensuring that the process to be followed is suitably rigorous and appropriate
- Short search strings can be used instead of one long one and this can make searches more manageable (although duplicated entries need to be dealt with)
- Suitable inclusion criteria can drastically reduce the number of papers accepted and thereby has a direct correlation with effort exerted (an important consideration for sole researchers)

- A test/re-test approach on a sample of papers helps to verify the results of data extraction and quality assessment (which can help build a new researcher's confidence).

Experience using the case study methodology

The case study methodology has helped to ensure valid and transparent empirical research. Multiple sources of evidence have been used. Before this the case study protocol was assessed by an expert and underwent peer-review. Additional steps, such as a consideration of rival explanations, also ensure acceptable and important findings. From the perspective of a PhD researcher undertaking a large-scale empirical study for the first time, the methodology provided a rigorous, but adaptable, means of investigating a real-world phenomenon. The development of a protocol ensured that problems were identified and considered. The protocol also gave the opportunity for feedback from experienced researchers which in turn increased confidence in the design of the study. Moreover, the protocol increased the likelihood that the research would be completed satisfactorily at the first attempt. This is a critical point for sole-researchers (such as PhD students), especially when human participants are involved, as time and opportunities for such research are often limited.

Reflection on other aspects of the research

While the thesis contributes to knowledge and the findings have scientific value, there is still scope for a critical reflection in regards to aspects of the work that was performed. One factor that warrants discussion is the fact that it was not possible to use the same programming exercises during all four workshops, as it was established during TTW1 that these instruments were flawed. Due to this, the performance of all groups could not be compared and contrasted which may have been helpful. As the programming exercises were developed specifically for the purposes of this work, it was also not possible to compare participants' performance to alternative groups unconnected with the project. The fact that the two student groups had different levels of prior programming experience should also be discussed. Whilst this is considered to be a strength of the research, as it demonstrates how a robot simulator has value when used by participants from

multiple backgrounds, it slightly limits the observations that can be made as the number of ‘true’ novices who took part is reduced. Potentially a screening questionnaire could have been distributed to participants in advance of the study to establish the true nature of their experience before the workshop. Even if it was determined that some participants had prior programming experience, however, it would have likely made little difference to the decision whether or not to involve them in the research due to the difficulty attracting partner institutions to participate in the first place.

Whilst it is firmly believed that the decision to deliver the workshop over two days was the correct one, given that feedback provided during the exploratory studies suggested this would be best for both students and partner institutions, time and resource permitting it would have been interesting to run the workshop with a comparable ability group over several sessions (i.e. one two-hour session per week over several weeks) to see if this led to different results. It may have also been enlightening to follow-up with participants several weeks after each workshop. Data collected during such a follow-up could have been used to verify the overall findings of the research and participants’ opinions and programming ability could have been measured in retrospect. It was not feasible to undertake such activities, due to the timing of the initial workshops, as these took place in the weeks preceding the annual summer break. As a result, feedback could not be gained immediately after the research (as participants were on vacation) while upon participants’ return to their education institute it was the beginning of a new term (and staff and student priorities understandably lay elsewhere).

In regards to the content of the workshop, the use of more prescriptive methods to control the simulated robots during earlier parts should be discussed. Initial tasks involved using commands such as `forward` and `left` and not less restrictive methods as was the case later. The decision to introduce the simulator in such a manner minimised the risk that participants would be overwhelmed by their initial encounter with Kebot and programming. Whilst some early tasks may be viewed as reminiscent of those that are completed using Logo or Karel, it is believed that it was clear to participants from the outset that Kebot is very different compared to these approaches. This is because 2D and 3D objects, and the more advanced functionality of Kebot (such as unrestricted

movement and rotational freedom), were demonstrated in the early stages of the workshop. Moreover, the fact that Kebot was designed to replicate the movement of real-world robots means that it does not offer a ‘perfect’ simulation anyhow. Even during the completion of tasks that involved the use of methods such as `left`, the angle at which a robotic agent turns is rarely a precise 90 degrees (as it is difficult to exhibit such exact behaviour using a robot which has four wheels as is the case with those simulated in Kebot). The fact that Kebot does not offer a ‘perfect’ simulation was noted by participants early in the workshop. On reflection, the decision to adopt a more limited approach during the early stages is considered to be the correct one as it allowed users to get to grips with the software in addition to the already difficult task of learning programming. As a result, such a decision would not be changed. It would have been interesting, however, to have investigated whether participants preferred using more constrained ‘Logo-type’ commands or the less restricted control methods used later in the workshop. It is the author’s opinion that the use of less restricted means for controlling the simulated agents was favoured by users. This view is supported by comments made during one of the teacher interviews (T3) when it was stated, “*The first day could have been a little bit pacier... I personally enjoyed the second day better because it was (more about) the robots*”.

Finally, whilst it is only possible to infer a judgement on the scalability of the approach, and such an opinion cannot be stated as fact, it is the author’s belief that a robot simulator would be an effective tool if used during a longer course (dependent on the development of associated supporting materials, extension activities and project tasks). The educational context would be the main factor responsible for determining how long the software could be used for. In a high school or FE setting, during one or two hour lessons once or twice a week, it would likely be possible to use a simulator comparable to Kebot over a full academic term (of around 12 weeks). In a university setting this time would probably be shorter, due to the increased pace of teaching, but it is thought that a simulator could be used for the first three to four weeks of a programming course.

8.5 Guidance for Related Future Work

In this section a set of recommendations, and specific suggestions, for future work are provided. Whilst one robot simulator, Kebot, has been used during the research it is believed that the findings are generalisable to other robot simulators. Kebot is considered to be a ‘generic’ robot simulator as it has no features that are exclusive to the platform. Viewed in its simplest form the main features of Kebot allow:

- Simulated agents to rotate through 360 degrees
- Simulated agents to move forwards and backwards
- For visual arena backgrounds to be loaded (or ‘drawn’) onto the arena floor
- For the introduction of simple ‘2D’ (over which an agent can traverse) and ‘3D’ objects (which cannot be traversed) in the manner of a traditional ‘Paint’ application
- For interaction with the environment through sensors that can detect the presence of 2D and 3D objects

As the main features of Kebot do not extend beyond those outlined above, the generalisability of the software is enhanced as it is considered that these elements would form a staple part of any newly developed comparable simulator. It was intentional that Kebot would remain, as far as was possible, simple and neutral to counter the suggestion that the results were not applicable to other simulator software. Whilst Kebot is believed to be a simple implementation of a robot simulator, it is important to note that it can be considered as advanced compared to some previously developed simulators such as Karel. Additional issues associated with the generalisability of results are presented in Section 8.6 ‘Threats to Validity’. The recommendations that have been made to support the future development and use of robot simulators, in an introductory programming context, are presented in Table 8-2.

Recommendation	Description
Recommendation One: Maintain a focus on programming	Simulators allow an opportunity for learners to focus on the task of programming. One advantage of simulators is that they can overcome issues with physical robot technology that distract learners (e.g. problems with mechanical failure, storage etc). A simulated environment should, therefore, aim to maintain a focus on programming by taking advantage of these reduced issues. This can be achieved by ensuring associated activities promote the learning of programming concepts and do not overly focus on the simulator technology itself.
Recommendation Two: Appreciate the importance of visualisation and interactivity	Participant responses highlight the importance of visualisation and interactivity when learning programming using a robot simulator. It is important that visual feedback provided by a simulated environment, therefore, reflects the user's programs and offers a simulation that allows code and on-screen activities to be clearly traced. Indication of when a simulator is running should also be provided.
Recommendation Three: Encourage motivation through accessibility	Learner motivation should be the main factor considered when developing a simulated environment. A robot simulator has been found to have significant motivating appeal, for both novices and educators alike, and it is fair to assume that this would translate to an increased level of effort. A simulator should seek to encourage this motivation by ensuring that it remains accessible to new users.
Recommendation Four: Support the needs of diverse users	A simulated environment should be suitable for use by people of varying abilities. To ensure that users are not overwhelmed by the features of a simulator, nor left frustrated by a lack of pace, the software (and associated tasks) should allow complexity to be gradually increased by the users themselves. This may be achieved through an "add-on" system where new software or task features can be incrementally introduced by learners. The importance of "extension activities" was highlighted in each of the teacher interviews. These activities would allow a learner to continually work within, but at the limits of, their programming ability.
Recommendation Five: Beware unnecessary complexity	The importance of ensuring that a simulator does not add an additional layer of complexity that distracts learners was established during the exploratory research. Whilst there may be a temptation to appease today's "Nintendo Generation" who have grown up with high powered and complex software, it remains important that simulated environments use clear and simple GUIs containing only frequently used features. In regards to the programming interface itself, the names of relevant control methods and variables should be identifiable and not abstract. Unnecessary code, related to the functioning of the simulator, should also be concealed.
Recommendation Six: Understand the needs of all learners involved	Depending on the situation in which a simulator is being used, there may be more than one group of learners involved. This is especially true if a simulator is intended for use in a high school or similar environment. This is because teachers responsible for introducing the approach (in addition to technical support staff) may have little or no prior programming experience and should, therefore, be viewed as learners themselves. A simulator should, where possible, aim to offer support to such individuals. A comprehensive requirements gathering exercise should be undertaken prior to the development of a simulator.

Table 8-2. Recommendations to support the future development and use of robot simulators in an introductory programming context.

It is recommended that further research is undertaken to consider the application of the Kebot robot simulator, and associated 10 hour workshop, in alternative learning settings. This would allow an opportunity to provide further evidence to support the findings reported in this thesis or to contradict them. The simulator and workshop are prepared for use, in other research/educational settings, by different researcher(s) and/or educator(s). Indeed, there is the potential for Kebot, and materials used during the project (including data collection instruments), to be applied in multiple educational settings simultaneously. This would allow for comparisons to be drawn and could lead to further confirmation and validation of the findings reported.

The development and use of alternative means for data collection is another possibility. As discussed in the next sub-section, the construct validity of data collection instruments is one possible threat to the reliability of the research results. This is because, due to the nature of the work, it was required that all data collection instruments were newly developed and there is a risk that some of the instruments used do not accurately measure what it was intended they would measure. The development of alternative instruments, in particular the programming exercises completed by students, would allow for an additional insight into how a robot simulator supports the learning of programming. This may also help to overcome questions related to participant performance on some of the programming tasks (in particular the final task of the post-workshop exercise) and could provide further confirmatory evidence in regards to effectiveness. The use of alternative means for the collection of qualitative data is another option.

How a robot simulator could be used during a more extensive programming course is an additional avenue that future research may explore. For this project it was not possible for the workshop length to be greater than two full days. This was due to ethical issues (as the majority of participants were enrolled on a full-time education course and involvement in the research could not be allowed to distract from other commitments) and practical issues (given that a more substantial workshop would have required the development of additional materials which would have taken much longer). It would be interesting to determine whether such a tool was able to support the learning of more advanced concepts, especially considering that concepts learned later

in programming courses are often harder for novices to understand (Petre *et al.*, 2003). In light of the revised National Curriculum and the introduction of Computing as a compulsory high school subject in UK schools from September 2014, it is believed that high schools may have an increased interest in the approach reported. To build on this work it is recommended that contact should be initiated with local high schools, as these may be a good source for potential research partners.

Whilst in its present state Kebot offers a complete robot simulator solution, there is also the potential to modify the software. This could enable additional avenues for research. Both Kebot, and the associated workshop, are freely available online and modification is encouraged. It is also believed that the potential exists to conduct a more specific investigation from the outset, as opposed to the exploratory one reported in this thesis. Indeed, it would be interesting to explore whether learners who have used a robot simulator learn programming concepts more accurately. The extent to which a robot simulator adds an additional layer of complexity for learners and educators alike could also be explored. The degree to which a robot simulator, modelled on a real-world robot, offers an more engaging approach compared to more restricted environments (such as Logo) may additionally be considered. Finally, more research is needed to better understand the effect that a robot simulator has once implementation ends (i.e. do learners who have been exposed to a robot simulator go on to show greater aptitude during a more extensive programming course).

In regards to this work, from the author's perspective, future activities will involve investigating ways in which this thesis can represent the beginning of a project and not the end. Specifically, how Kebot can be promoted as an introductory programming resource for use by teachers of introductory programming (whether in high school, Further Education or early-University settings) is being considered. The fact that a number of educators have expressed an interest in using the Kebot simulator and workshop themselves highlights how there is real interest in the approach. Efforts will now be made to ensure that the resources that have been developed during this project are made available to educators and researchers looking to build on the activities reported in this thesis.

8.6 Rival Explanations and Threats to Validity

In the remainder of this chapter rival explanations, which may be accountable for the results of the study, are presented. Threats to validity are also outlined.

8.6.1 Rival Explanations

Reporting that a study sought out, considered and did not find evidence to support a number of plausible rival explanations enhances the credibility of case study research and helps to counter the suggestion that the results are shaped by any predispositions or bias. This search for rivals involves looking for alternative themes, divergent patterns and opposing explanations (Patton, 2001). A search for rivals was embedded in the data collection and analysis stages as outlined below. Yin lists many potential rivals (Yin, 2009). Several of these were considered to be of relevance:

- **Null Hypothesis** (i.e. observations are the result of chance circumstance only). *How it was addressed:* Workshops replicated. Multiple sources of evidence used to support findings.
- **Novelty of Simulator** (i.e. the novelty of the simulator encourages participants to say that they have learned more than they actually have – interest in the learning mechanism has been confused with actual learning). *How it was addressed:* By the scoring process which distinguishes between deep and surface learning on the post-workshop exercises. Analysis of performance on the two in-workshop tests also helps to provide a measure of progress.
- **Experimenter Expectation Effect/Confirmation Bias** (i.e. the scoring of the programming exercises is influenced by the experimenter's expectation that the simulator is effective). *How it was addressed:* By adhering to pre-determined marking schedules and by subjecting programming exercise data to second marking (by two PhD supervisors).
- **“Good Subject Effect”** (i.e. when participants mark their subjective opinions strongly in favour of the simulator to aid the research project and not because it helped them to learn). *How it was addressed:* By asking participants to identify up to three aspects they liked/disliked about the simulator – it was considered that participants were more likely to be truthful when

identifying positive and negatives than simply answering a question on whether the simulator helped them to learn. If far more positives than negatives were reported it was considered that this would corroborate positive answers to questions related to the effectiveness of the workshop and robot simulator. This also helped to overcome the issue that participants would be inclined to respond positively, when asked about their experiences using Kebot, as they have no other reference frame.

- **Implementation Rival** (i.e. the nature of the workshop sessions and not the robot simulator accounts for the results). *How it was addressed:* By asking participants to rate (on a five point scale) the effectiveness of the simulator in addition to the effectiveness of the workshop in general. If substantially more participants rated the workshop as effective, and the simulator as ineffective, it was considered that the nature of the workshop itself may have accounted for the results of the study.
- **History Effects** (i.e. when research is conducted over several weeks or months environmental or others changes may impact on participants' behaviour). *How it was addressed:* The case study consists of two cases. Two workshops were conducted within each case. To ensure outside factors did not influence the behaviour of one group but not the other the workshops were held as close to one another, in terms of timing, as was possible. A relatively short amount of time also elapsed between the pre- and post-test occasions (i.e. distribution of the pre- and post-workshop questionnaires)
- **Practice Effects** (i.e. when participants are exposed to repeat measures of similar data collection instruments their performance on the second and subsequent tests may differ from what it would otherwise be). *How it was addressed:* By ensuring that the in-workshop exercises (IWE1 and IWE2) were substantially different from one another so that participants' experience completing IWE1 would not have an influence on the second test. This was done by ensuring that completely different questions were used (although a similar presentational format was followed). Likewise, the post-workshop exercises differed entirely from the in-workshop tests as programming tasks were completed and participant code was collected.

8.6.2 Threats to Validity

In this sub-section threats to validity are described. Threats to validity can be classified as follows (Runeson & Höst, 2009):

- Construct validity – The extent to which a study measures what the researcher has in mind (i.e. research questions measure what they intend to).
- Internal validity – When a researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor.
- External validity – The extent to which it is possible to generalise findings and to which they are of interest to people outside the investigated case.
- Reliability – The extent to which the data and the analysis are dependent on the specific researchers.

Each of these threats will now be considered.

Construct Validity

Construct validity requires a researcher to use the correct measures for the concepts being studied. For this study, threats to construct validity are associated with the methods for collecting data (specifically the pre- and post-workshop questionnaires, in- and post-workshop programming exercises, in-workshop observations and teacher interviews). These threats are amplified due to the data collection instruments being developed for the purposes of the research. In this sub-section construct, and general, threats to validity are considered in the context of the data collection strategy. Multiple sources of data were used during the study and this helps to negate the potential issues with construct validity.

The main threat to the construct validity of the *questionnaires* is that some included items may not measure what was intended would be measured or that questions were biased and influenced responses. Steps taken to minimise these risks included the use of open-questions (which enables cross-verification of responses to closed-questions), the checking of the questionnaire items by a

PhD supervisor (to determine whether the inclusion of each item was reasonable) in addition to consideration of relevant literature related to questionnaire design. It should also be noted that the questionnaires shared significant similarities to those used during the exploratory studies (reported in Chapter Three). As this was the case, and as no problems with the questionnaires were reported during this exploratory work, the risk that they are significantly flawed is considered to be small.

The *programming exercises* that were undertaken (whether before or after the workshop) aimed to determine student programming progress. As with the questionnaires, there is a threat that these exercises do not measure the construct of interest (i.e. the effectiveness of the simulator). Indeed, whilst the assessments were developed in conjunction with a researcher who possesses greater programming teaching experience, there is a possibility that the test items do not measure the intended construct due to one or more flaws.

An additional threat is related to the generalisability, and strength of, conclusions that can be drawn considering that pre-tests were not completed by participants. As pre-testing was not undertaken the assumptions that can be made, when drawing on the results of the programming exercises alone, are limited. This is because no quantitative gain in performance can be demonstrated. Given that the nature of case study research involves drawing on multiple sources of information to make conclusions, however, the fact that these exercises alone cannot be used to demonstrate the effectiveness of the intervention is not considered to represent a problem.

Instrument decay is where participant performance on an assessment deteriorates due to fatigue (or lack of interest). Given that students were used to short lessons (of between 50 minutes and one hour), and not intensive sessions (spanning several hours and multiple days), there is a risk that performance on the assessments diminished as the workshop progressed. This is particularly true in regards to the post-workshop exercises (as previously outlined earlier in this discussion) given that these were completed at the very end of the session. The design of the programming exercises also represents a potential threat and flawed questions on the programming exercises could have led to skewed results while the use of pseudo-code on the post-workshop exercises may have

inadvertently helped or hindered performance. There is also a risk that participants may have assisted, or copied from, one another. Whilst this danger is considered to be small, given that the workshop leader ensured completion of the exercises independently in silence, it is a possibility.

The scoring criteria used may also have had an impact. For the in-workshop exercises a score of six or greater was considered to indicate satisfactory learning while for the post-workshop exercises a three-point grading system was used consisting of Grade A, B or C (with Grade B representing satisfactory but not full completion of a task). There is a possibility that these thresholds are not suitable for determining performance and that these boundaries should be higher (or lower). The risk that the results of these exercises has been misinterpreted and incorrect scores awarded, however, is considered to be low given that marking schedules were used and responses were subject to second marking by two PhD supervisors.

Whilst the exercises were not trialled with students in advance, early versions of all three assessments were used during the first workshop involving trainee teachers (TTW1 – Chapter Six). This allowed an opportunity for testing and validation before the workshops involving students. A number of issues were identified following TTW1 and changes were made, after consulting a PhD supervisor, in advance of later workshops. Whilst it is not possible to guarantee that these modifications eliminated all problems with the exercises used during the student workshops, potential risks are considered to have been reduced.

Workshop logs (based on *in-workshop observations* by the workshop leader) were maintained to supplement data collected by other means. Despite the development of a pre-determined recording criteria, which was intended would offer a systematic and consistent means of recording observations, potential issues with construct validity have been identified. To ensure the underlying construct of interest was actually measured, the recording criteria were influenced by the research propositions and question. As the in-workshop observations are an example of when a researcher's subjective opinion is relied on, there is a chance that critical points may not have been recorded as they were considered unimportant. Alternatively, information may have been unconsciously

distorted when being recorded. This risk is increased due to the fact that the observer was also responsible for leading the workshop. It should be noted that the in-workshop observations were made primarily to supplement other collected data and to help in the identification of outliers, exceptional cases etc. The observations are not, therefore, considered to be one of the primary sources of data.

Issues with construct, and other, threats to validity are likewise associated with the semi-structured *interviews* that were completed. A set of interview questions were developed, based on the research propositions, to ensure collected data would address the research aims. Despite this, there is a threat that responses did not allow the construct to be addressed due to questions being misunderstood or replies being ambiguous. This risk is increased as it was not possible to discuss the final interview transcripts with participants due to the timing of the interviews (as they took place shortly before the annual summer holidays) and other factors (such as one participant being appointed at a different teaching institute following the interviews). The small number of teachers who took part must also be considered as this represents a threat to generalisability. Participants may have also received unintended prompts when being asked questions by the researcher while the confidence of the interviewee may have impacted responses.

Internal validity

Internal validity reflects the extent to which claims made and conclusions reached are warranted. The research methods used, the way in which the study was designed in addition to various other factors must be considered. Attempts to ensure internal validity were partially addressed by embedding a search for rival explanations in the data collection and analysis stages (as discussed previously). It is not possible, however, to ensure that research accounts for all potential risks and the remaining threats to internal validity are outlined in this sub-section.

The Design of the Study

When considering the internal validity of the study, several factors have to be explored for causality. The BlueJ IDE was selected for use during the workshop as this has been used on introductory programming courses at Keele University for several years (and, as such, the author and University staff had good knowledge of the software). As BlueJ is a Java IDE designed to support the learning of programming there is a possibility that it may have influenced the results of the research. The likelihood of such a threat, however, is considered to be small. This is because BlueJ was used for its code editing functions and no aspect of the workshop focused on the use of more advanced software features. To elaborate, the Unified Modelling Language (UML) aspects of BlueJ were not employed nor were features such as the Code Pad or other visualisation and interaction tools. It is believed, therefore, that the potential impact that BlueJ had upon the findings is minimal and that similar results would have been observed had a comparable IDE been used.

The workshop format itself presents a threat to the internal validity. The order in which items were introduced may have impacted upon findings reported. It is possible that if programming fundamentals were introduced in an alternative manner then observed results may differ. For instance, it is possible that some participants may have lost interest early on during the session because concepts were introduced too slowly/rapidly. Alternatively the structure of the session (specifically the fact that the workshop involved the introduction of concepts before a task phase and opportunity to reflect and reinforce knowledge) may have affected the performance of some of those involved. These issues are compounded due to the workshop being devised from scratch. The fact that the workshop format was influenced by the findings reported in the literature, a trial workshop was held in advance of the study with novice programmers and a more senior researcher (with introductory programming teaching experience) reviewed the workshop format all help to reduce such risks.

Support offered to participants during the workshop is another possible threat. There is a risk that the support offered was inconsistent (i.e. one or several participants received more help from the

workshop leader than others). Likewise, one set of participants may have offered each other a greater degree of assistance when completing the programming exercises. Each workshop followed the same format and the same supporting slides and workshop schedule were used during all sessions. Whilst this does not guarantee that precisely the same amount of time was spent focusing on a particular fundamental, such measures do help to ensure that significant deviations did not occur between workshops. The fact that there was no way to exercise complete control over the support participants received is one reason why the case study methodology was selected.

Sample attrition (i.e. drop-out) is an additional threat to internal validity. There is a risk that participants who dropped out did so due to dissatisfaction with the Kebot simulator or other aspects of the research. By withdrawing from the study before its conclusion the post-workshop questionnaire and programming exercises were not completed. This may have resulted in important information being omitted. Potentially, the inclusion of this missing data may have led to different study findings and conclusions. Such a risk is considered to be small, however, given that a small number of participants withdrew from the research.

Psychological Factors Relating to Participants

Rosnow and Rosenthal (Rosnow and Rosenthal, 1997) discuss factors that can impact upon the results of research involving human subjects. Several of these are of relevance. In regards to the questionnaires and teacher interviews participants may have responded in the manner in which they expected they should answer. This is known as the “Good subject effect” and involves participants responding in a way that they feel assists the researcher/research project. Another factor is that the novelty of the robot simulator may have also led participants to confuse interest in the software with actual learning. Measures that were taken to actively investigate both of these phenomena during the research have been outlined in Section 8.6.1 – Rival Explanations. The ‘Hawthorne Effect’ could similarly have distorted collected data. This is where participants act differently than they would otherwise because they are aware that they are taking part in a research project. The

Hawthorne Effect could have led to increased levels of participant effort during the workshops, in particular on the programming exercises.

An additional risk is that participants volunteered to take part in the research. It is not unreasonable to assume that those who are prepared to give up a substantial amount of time may be inherently more motivated than those who did not. The timing of the sessions involving students should also be considered because both workshops took place less than two weeks before the beginning of the annual summer holidays. It is possible that this could have led to increased (as participants had recently completed exams and could better focus on the task at hand) or decreased (as participants were unprepared to exert maximum effort completing a workshop that did not contribute towards any formal academic qualification) participant effort. For most students the fact that a new, complicated subject was also being learned could also have had an impact and the limitations of threshold concepts and cognitive load theory (as discussed previously in Chapter Five) could have played a part.

Factors Relating to the Researcher

In any project where the researcher takes an active role bias may inadvertently be introduced. For this work subconscious cues from the workshop leader may have sufficiently influenced participants to alter their behaviour during the workshop sessions or influenced their responses, and performance, on data collection exercises. The search for rival explanations that was undertaken minimises the risk that any pre-held expectations, on the part of the workshop leader, had any significant impact upon the outcome of the study. Third-party checking (by the members of the PhD supervisory team) also reduces any potential bias.

The personal qualities of the workshop leader must also be considered, in particular the fact that only limited introductory programming teaching experience was held prior to the workshop and as the author is not a teaching professional. Due to this there is a threat that the tutoring of subjects may have differed substantially from somebody with greater teaching/programming experience. The potential enthusiasm of a researcher who has been heavily involved in the preparation of a

study is an additional issue. Significant time and resources had been spent developing the robot simulator and workshop, in addition to the preparation and organisation of the sessions involving participants, and this could theoretically have led to the workshop leader ‘willing’ the approach to work to vindicate their efforts. For this research, to remove this bias, it would have been desirable to enlist the assistance of a neutral person to run some or all of the workshop sessions. This would have reduced the risk of any inexperience or potential enthusiasm (or cynicism) held by the author impacting upon results. Due to the scale of the project, however, it was not possible to do this. This is because multiple workshops were held over several days and as no other person was available to lead such sessions. A final issue to consider is the potential impact of teacher expectancy. This is where a teacher expects a particular group of participants to perform more or less well than an alternative group academically. This could have led to one group of participants receiving additional cues during the workshop or to modified behaviour on the part of the workshop leader. Despite such issues being a possibility the potential impact of such factors significantly affecting the results of case study are considered to be small. This is partly because a pre-determined workshop schedule was followed and as the associated slides that were used did not allow for significant deviation.

External validity

External validity is concerned with the generalisability of findings and the extent to which these are relevant to other groups of participants in alternative situations. Two main factors must be considered for threats to external validity, the Kobot simulator and the participants who took part.

The Kobot Robot Simulator

With the exception of the exploratory research (reported in Chapter Three) one robot simulator, Kobot, was used during the project. Kobot is viewed to be a ‘generic’ robot simulator as the main features of the software (e.g. that it allows agents to move through 360 degrees, forwards and backwards, and as environmental interaction can be achieved through sensors) would be features of

other comparable simulators. Whilst it is not believed to be the case, there is a risk that some features of Kebot are not neutral enough to allow sufficient generalisation to other robot simulators.

The fact that Kebot was used during a 10-hour workshop may also threaten the generalisability of findings to a more substantial programming course. This is because a judgement as to the scalability of the approach can only be inferred and not stated as fact. There is also a risk also that the positive findings reported may not have been observed if student exposure to the simulator was for longer. Potentially, participants may have grown frustrated and annoyed with the approach if exposure to the simulator was prolonged beyond the 10 hours.

The Sampling Method

All those who took part in the research did so voluntarily and this may threaten generalisability. This is because volunteers may be more motivated than others unconnected with the research. It is also possible that the novel nature of the research could have provoked interest and desire to take part. Due to the nature of the project it was not possible to use random samples from the population. To allow for generalisability the demographic characteristics of participants have been reported along with the results of data collection activities. The study also involved two separate workshops within each case being conducted and differences/similarities have been discussed.

The number of people who took part also presents an additional threat to external validity. As a relatively small number of participants comprised each group there is a risk that findings may not be applicable outside each specific case. The location where the workshops were held is an additional issue. This is because research results can be impacted when environmental conditions differ. As it was not possible to use the same location for each workshop, for various reasons, efforts were made to make the laboratory conditions as similar as possible (e.g. each participant had access to an individual PC, the lab layout was similar etc). As the research is considered to have been undertaken in a ‘typical’ computer laboratory (i.e. ones which come equipped with Microsoft Windows PCs and standard projection facilities) the results are not considered to have been limited by any specific environmental factor.

Reliability

Reliability is concerned with the extent to which research is dependent on a specific researcher(s). Efforts were made to ensure bias did not impact the results of the study including delivering the same presentation, and following the same format, during each workshop. It is not possible, however, to measure the success of these attempts at neutrality and there is a threat that the delivery of the workshop would be different if delivered by a person other than the author. This is because the author designed the workshop materials and, although attempts were made to ensure they were sufficiently self-explanatory and could be delivered by alternative people, some aspects of the workshop may confuse a person without the workshop leader's prior experience and knowledge of the project.

The author's relative inexperience teaching introductory programming could have led to modifications in participant behaviour as could the author's authority and personality. In addition, there is a chance that the workshop leader grew more accustomed to delivering the workshop and that modifications in behaviour may have had an impact. Whilst beyond the scope of this research, one means of ensuring such factors do not have an impact would be for one (or several) of the workshops to be delivered by an independent party and for collected data to be contrasted with that gathered during workshops led by the author. The fact that the workshop lasts for a minimum of 10 hours, however, coupled with the fact that substantial time would have to be spent bringing an additional workshop leader "up-to speed" with the Kobot software and associated materials, meant that it was not possible to do this. The fact that all workshop materials were newly developed leads to additional questions about reliability. This is because it is not possible to contrast collected data with that from unconnected research. Several measures were taken to minimise this risk including trialling data collection instruments in advance of their use, a PhD supervisory team reviewing such instruments, conducting a multiple-case case study as opposed to a 'one-shot' case study and considering the reliability of data collection instruments in the analysis phase.

8.7 Summary

In this chapter the evidence generated from the multiple sources of data used during the case study have been brought together. Data reported in Chapter Six (Trainee Teachers) and Chapter Seven (Students) has been used to address four propositions and the thesis research question. Evidence suggests that a robot simulator is an effective tool for supporting the learning of introductory programming concepts. This conclusion must be considered with caution, however, as a number of factors must be taken into account when interpreting results. The findings of the research have been examined critically and compared with previous work and an assessment has been made as to what has been learnt from the activities undertaken. Several recommendations, based on the lessons learned, have been presented. Ways in which future work may seek to build on the findings reported have also been provided. Threats to validity and rival explanations have also been discussed. In the following chapter a summary and conclusion is outlined.

Chapter Nine

Summary and Conclusion

This chapter offers some concluding remarks on the results of the research as a whole. How the project aim, to investigate whether simulated robots are effective tools for supporting the learning of introductory programming, has been achieved is outlined. The meaning, importance and contribution of this research are all also discussed.

9.1 Summary and Conclusion

The overall aim of this thesis was to investigate whether simulated robots are effective tools to support the learning of introductory programming. Novel and innovative research has been undertaken to achieve this aim. In the early stages of the project a Systematic Literature Review (SLR) was completed before exploratory empirical research was conducted. The SLR established the potential to investigate robots as programming teaching tools. This was particularly true in regards to simulated robots, as limited high quality research was found to consider such an approach. Lessons learned during the exploratory studies also had a significant influence on the work that followed. The SLR and exploratory studies provided the justification for this research and demonstrated the viability of it.

The Kobot robot simulator and an associated 10-hour programming workshop were subsequently developed. Previous work, and lessons learned when conducting the preliminary research, significantly influenced the development of the simulator and workshop. Kobot allows for real-time interaction with a simulated agent, the customisation of an agent's environment using objects, coding in 'real' Java and the creation of imaginative programming tasks. Kobot provides an accurate representation of a real-world robot and agents can move freely through 360 degrees and interact with their environment through various sensors. A realistic representation of a physical robot is, therefore, provided and this can be considered as advanced when compared to more restricted simulated environments (i.e. ones where robots inhabit a grid-based world). Kobot has been optimised for supporting the teaching of introductory programming concepts after a review of educational software guidelines. The associated workshop allows the introduction of fundamental programming concepts identified by the ACM and IEEE Joint Task Force Computer Science Curriculum. Use of these concepts was considered suitable for a number of reasons.

Student programmers, in addition to pre- and in-service high school teachers, have taken part in empirical research designed to investigate the use of Kobot. This has allowed for unique research into the effectiveness of a robot simulator. A large quantity of qualitative and quantitative data has

been collected and interpreted. Pre- and post-workshop questionnaires, observations, teacher interviews and three programming assessment exercises have been used. In total, over 75 participants have taken part in research conducted during the project. Steps taken to ensure the rigour of the study have been provided and threats to validity, rival explanations and limitations have all been considered. The case study methodology was used to ask the following question:

Is a robot simulator an effective tool for supporting the learning of introductory programming?

Effectiveness has been determined after considering participants' opinions, attitudes and motivation using the robot simulator in addition to an analysis of students' programming performance. A robot simulator has been judged to offer an effective means of supporting the learning of introductory programming because:

- 1) Participants enjoy learning programming in such a manner, and,
- 2) Participants believe the approach to be valuable and useful, and,
- 3) Most evidence suggests that students successfully learnt introductory programming concepts (although several factors must be taken into account when interpreting the results of completed programming exercises)

The cohort of Case One participants (22 trainee high school teachers) believed Kebot offered an interesting and enjoyable means of introducing programming. Most trainees had some prior programming experience and the majority believed that the robot simulator offered a more effective approach compared to their previous introduction to the subject. Following the workshop none of the trainees stated they would not consider using Kebot in their own future lessons and several requested a copy of the software and workshop materials for use in their own teaching. The confidence of trainees in their ability to teach programming, however, did not improve following the workshop. Several programming exercises were used during the trainees' workshops as a means of testing and validation in advance of the student workshops.

Case Two participants (21 students aged 16 to 18 who were in Further Education) likewise perceived learning programming through the robot simulator to be enjoyable, effective and engaging. Following the sessions none of the students stated that they would not consider learning programming further. The fact that robots have a visual representation, and that output is more entertaining than textual output, was considered to be one advantage of using simulated robots. As it has been recognised that learning is improved and better enjoyed when students work on tasks that interest them, this has contributed to the conclusion that a simulator is an effective mechanism for supporting the learning of programming. Student performance on in- and post-workshop exercises is also considered to offer evidence of effectiveness. Due to the nature of the exercises used, however, it is not possible to make strong claims about effectiveness based on these alone. This is because all of the exercises were developed for the purposes of this work and as the reliability of these instruments has not been independently verified. The poor performance of students on one of the post-workshop tasks also raises questions. Other sources of data have been used to verify the findings reported and these help to provide a further insight. Specifically, in-workshop observations and semi-structured interviews with in-service high school teachers were undertaken.

The fact that the robot simulator was highly regarded by almost all participants indicates how the approach appeals to people of both genders and of various experience and ages. When critically compared with existing literature, the results are considered to support those of previous work. It is believed that the findings of the research are applicable to other robot simulators that may be used to support the teaching of programming. A set of recommendations, based on the results of the research and lessons learned, have also been provided. These will help to support the future development of robot simulator software for use in an introductory programming context.

9.2 The importance, meaning and contribution of this research

To conclude the thesis the meaning, importance and contribution of the research are discussed.

How is this research important?

The SLR (and supplementary literature search) established a need to investigate the use of simulated robots as programming teaching tools. This is because the SLR found limited high quality research related to the use of simulated robots. The work reported in this thesis is considered to be valuable and necessary given that it helps to address the gap in knowledge identified during the SLR. The findings are important as they help to influence our knowledge of, and experience using, robot simulators as tools to support the learning of programming. The fact that evidence suggests a robot simulator to be a motivating tool is arguably the most important finding. The link between learner attitude, learner achievement and educational effectiveness has already been discussed. As the simulator was found to be enjoyable and engaging the potential to further develop the approach has been established. This is because learners are far more likely to successfully learn when these factors are the case.

What do the findings of the research mean?

Whilst other issues must also be considered, the work completed demonstrates how a robot simulator is an effective tool that supports the teaching of introductory programming. Knowledge has been advanced by the evidence presented and this may help to inform educators' decisions about whether or not to use a robot simulator in their own lessons. The work will also have implications for future research as, now that an extensive exploratory study has been undertaken, and such a tool has been judged to be effective, more specific research can be completed.

How does this research make an original contribution to knowledge?

This thesis reports on an extensive and rigorous investigation into the use of a robot simulator to support the learning of introductory programming for the first time. A substantial amount of new information has been set down and this demonstrates how the work is novel. Several diverse groups of participants, including students (aged 16 to 18 years old) and high school teachers (both pre- and in-service), were involved. This has enabled an investigation into effectiveness in a manner not previously completed. The timing of the study also allows an additional original contribution. The launch of a new high school curriculum, in 2014, will bring changes to how programming is taught in British high schools (Copping, 2012; Wells, 2012). The fact that recent high school leavers, in addition to pre- and in-service high school teachers, were involved has relevance in the context of such reforms.

The SLR also contributes to knowledge as it represents the first time that the methodology has been applied to this research area. The SLR was disseminated in conference and journal papers and these have subsequently been cited by several independent studies. It is believed that this highlights the value and importance of the findings documented. Likewise, the case study methodology has underpinned work in this research area for the first time. As with the SLR, the case study protocol has been disseminated at an academic conference and has the potential to influence future work. A set of recommendations have also been produced to support the future development and use of robot simulators in an introductory programming context. Other dissemination activities have also taken place and the results of the exploratory studies have previously been reported.

Finally, the Kobot robot simulator developed during the project has not been used previously in an education research context. Moreover, the supporting 10-hour programming workshop and associated data collection instruments were newly created for the purposes of this work. All of these resources have been made freely available for use by educators and researchers alike and this allows an original contribution beyond this thesis that is both original and substantial.

References

- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication*, 5(3), 438.
- ACM/IEEE. (2008). *Computer Science Curriculum 2008: An Interim Revision of CS 2001*. Report from the Interim Review Task Force. Available online: <http://www.acm.org/education/curricula/ComputerScience2008.pdf> (Accessed 1st October 2013).
- Alavi, M. (1994). Computer-Mediated Collaborative Learning: An Empirical Evaluation. *MIS Quarterly*, 18(2), 159-174.
- Aleman, J., & Cervera, E. (2012). Appealing Robots as a Means to Increase Enrollment Rates: a Case Study. In *Proceedings of the 3rd International Conference on Robotics in Education RiE2012* (pp. 15-19). MatfyzPress, Czech Republic.
- Alimisis, D., Moro, M., Arlegui, J., Pina, A., Frangou, S., & Papanikolaou, K. (2007). Robotics & Constructivism in Education: the TERECop project. In *EuroLogo* (Vol. 40, pp. 19-24).
- Anderson, E. F., & McLoughlin, L. (2006). Do Robots Dream of Virtual Sheep: Rediscovering the Karel the Robot for the Plug & Play Generation. Available online: <http://eprints.bournemouth.ac.uk/477/1/gdtw06aml.pdf> (Accessed 1 October 2013).
- Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2005). *A taxonomy for learning, teaching, and assessing*. Longman.
- ANSI Standards Committee on Dental Informatics. (2001). *Guidelines for the design of educational software*. Working Group Educational Software Systems.
- Avison, D. E., Lau, F., Myers, M. D., & Nielsen, P. A. (1999). Action research. *Communications of the ACM*, 42(1), 94-97.
- Baxter, P., & Jack, S. (2008). Qualitative case study methodology: Study design and implementation for novice researchers. *The Qualitative Report*, 13(4), 544-559.
- BCS. (2012). Guidelines on course accreditation: Information for universities and colleges - September 2010 updated for use from Autumn 2012. Available at: http://www.bcs.org/upload/pdf/hea-guidelinesfull-2012_1.pdf (Accessed 1st October 2013). British Computer Society.
- Beale, R., & Sharples, M. (2002). Design guide for developers of educational software. *British Educational Communications and Technology Agency*. Available online: <http://www.eee.bham.ac.uk/sharplem/Papers/Design%20Guide.pdf> (Accessed 1st October 2013)
- Becker, B. W. (2001). Teaching CS1 with Karel the robot in Java. *ACM SIGCSE Bulletin*, 33(1), 50-54.
- Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45-73.
- Bennedsen, J., & Caspersen, M. E. (2006). Assessing process and product-a practical lab exam for an introductory programming course. In *36th Annual Frontiers in Education Conference* (pp. 16-21). IEEE.
- Bergin, J. (2000). Fourteen Pedagogical Patterns. In *EuroPLoP* (pp. 1-49).
- Bergin, J., Lister, R., Owens, B., & McNally, M. (2006). The first programming course: ideas to end the enrolment decline. In *ACM SIGCSE Bulletin* (Vol. 38, No. 3, pp. 301-302). ACM.

- Biggs, J. (2003). Aligning teaching and assessing to course objectives. *Teaching and Learning in Higher Education: New Trends and Innovations*, 2, 13-17.
- Black, P. (2004). *Working inside the black box: Assessment for learning in the classroom*. Granada Learning.
- Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives: Handbook I: Cognitive domain*. New York: David McKay, 19, 56.
- Bonner, S. E. (1999). Choosing teaching methods based on learning objectives: An integrative framework. *Issues in Accounting Education*, 14(1), 11-15.
- Borge, R., Fjuk, A., & Groven, A. K. (2004). Using Karel J collaboratively to facilitate object-oriented learning. In *Proceedings of IEEE International Conference on Advanced Learning Technologies* (pp. 580-584). IEEE.
- Bradshaw, P., Cameron, K. & Younie, S. (2011). A case study examination of the BBC News School Report project in Initial Teacher Education across three sites for the Training and Development Agency for Schools (TDA) 2011. ITTE.
- Bradshaw, P., & Woollard, J. (2012). Computing at school: an emergent community of practice for a re-emergent subject. Accessed online: http://oro.open.ac.uk/34002/2/ICICTE_paper_138_Bradshaw_and_Woollard_FINAL_v3.pdf (Accessed 1st October 2013).
- Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT Press.
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2), 77-101.
- Brauner, P., Leonhardt, T., Ziefle, M., & Schroeder, U. (2010). The effect of tangible artifacts, gender and subjective technical competence on teaching programming to seventh graders. In *Teaching Fundamentals Concepts of Informatics* (pp. 61-71). Springer Berlin Heidelberg.
- Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4), 571-583.
- Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). Using a protocol template for case study planning. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. University of Bari, Italy.
- Brereton, P., Turner, M., & Kaur, R. (2009). Pair programming as a teaching tool: a student review of empirical studies. In *22nd Conference on Software Engineering Education and Training* (pp. 240-247). IEEE.
- Brooks-Young, S. (Ed.). (2010). *Teaching with the tools kids really use: Learning with web and mobile technologies*. SAGE.
- Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: a way to learn programming principles. *Education and Information Technologies*, 2(1), 65-83.
- Buck, D., & Stucki, D. J. (2001). JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum. *ACM SIGCSE Bulletin*, 33(1), 16-20.
- Burdea, G. C., Cioi, D., Kale, A., Janes, W. E., Ross, S. A., & Engsberg, J. R. (2012). Robotics and Gaming to Improve Ankle Strength, Motor Control and Function in Children With Cerebral Palsy - A Case Study Series. *Neural Systems and Rehabilitation Engineering*. 21(2). IEEE.
- Butterworth, D.T. (2012), Teaching C/C++ Programming with Lego Mindstorms, In *Proceedings of the 3rd International Conference on Robotics in Education RiE2012* (pp. 61-65). MatfyzPress, Czech Republic.

- Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. In *ACM SIGCSE Bulletin* (Vol. 38, No. 1, pp. 27-31). ACM.
- Case, J. (2008). *Education theories on learning: an informal guide for the engineering education scholar*. Higher Education Academy Engineering Subject Centre. HEA.
- Čermák, P., & Kelemen, J. (2011). An Attempt to Teaching Programming with Robots. In *Research and Education in Robotics-EUROBOT 2011* (pp. 78-87). Springer Berlin Heidelberg.
- Cliburn, D. C. (2006). Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum. In *36th Annual Frontiers in Education Conference* (pp. 1-6). IEEE.
- Cliburn, D. C. (2006b). The effectiveness of games as assignments in an introductory programming course. In *36th Annual Frontiers in Education Conference* (pp. 6-10). IEEE.
- Computing at School (CAS). (2011). Switched On – Computing at School Newsletter Winter/Spring. BCS.
- Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 191-195). ACM.
- Copping, A. (2012). Curriculum approaches. *Primary Professional Studies*, 24.
- Cowden, D., O'Neill, A., Opavsky, E., Ustek, D., & Walker, H. M. (2012). A C-based introductory course using robots. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 27-32). ACM.
- Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, Inc.
- Crisp, G. T., & Palmer, E. J. (2007). Engaging academics with a simplified analysis of their multiple-choice question (MCQ) assessment results. *Journal of University Teaching & Learning Practice*, 4(2), 4.
- Cruzes, D. S., & Dybå, T. (2011). Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53(5), 440-455.
- Currie, E. (2006). *The Fundamentals of Programming Using Java*. CENGAGE Learning.
- da Silva, F. Q., Santos, A. L., Soares, S., França, A. C. C., Monteiro, C. V., & Maciel, F. F. (2011). Six years of systematic literature reviews in software engineering: An updated tertiary study. *Information and Software Technology*, 53(9), 899-913.
- desJardins, M., Ciavolino, A., Deloatch, R., & Feasley, E. (2011). Playing to Program: Towards an Intelligent Programming Tutor for RUR-PLE. In *Second AAAI Symposium on Educational Advances in Artificial Intelligence*.
- de Vaus, D. (2002). *Surveys in social research*. 5th Edition. London: Routledge.
- du Boulay, B. (2000). Fallible, Distractible, Forgetful, Willful and Irrational Learners. In *Computers as Cognitive Tools* (339-376). Routledge.
- Daly, C., & Waldron, J. (2004). Assessing the assessment of programming ability. In *ACM SIGCSE Bulletin* (Vol. 36, No. 1, pp. 210-213). ACM.
- Data Protection Act. (1998). Parliament, U.K.
- Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. *Information Systems Journal*, 14(1), 65-86.

- Douglas, S., Farley, A., Lo, G., Proskurowski, A., & Young, M. (2010). Internationalization of computer science education. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 411-415). ACM.
- Drever, E. (1995). *Using Semi-Structured Interviews in Small-Scale Research - A Teacher's Guide*. The Scottish Council for Research in Education.
- Drummond, S. A. (2009). Investigating the Impact of Entry Qualifications on Student Performance in Computing Programmes at Undergraduate Level. Doctoral Thesis. Durham University. Available online: <http://etheses.dur.ac.uk/154/> (Accessed 1st October 2012).
- Duncan, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In *Proceedings from the 14th Workshop of the Psychology of Programming Interest Group*, Brunel University (pp. 89-99). PPIG.
- Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9), 833-859.
- Eagle, M., & Barnes, T. (2008). Wu's castle: teaching arrays and loops in a game. In *ACM SIGCSE Bulletin* (Vol. 40, No. 3, pp. 245-249). ACM.
- Easterbrook, S., Singer, J., Storey, M. A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285-311). Springer London.
- Edwards, S. H. (2003). Rethinking computer science education from a test-first perspective. In *Companion of the 18th Annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 148-155). ACM.
- Enderle, S. (2009). Grape—graphical robot programming for beginners. In *Research and Education in Robotics—EUROBOT 2008* (pp. 180-192). Springer Berlin Heidelberg.
- Escudero, E., Reyna, N. & Morales, M. (2000). The level of difficulty and discrimination power of the basic knowledge and skills examination. *REDIE*, 2(1).
- Esteves, M., Antunes, R., Fonseca, B., Morgado, L., & Martins, P. (2008). Using Second Life in programming's communities of practice. In *Groupware: Design, implementation, and use* (pp. 99-106). Springer Berlin Heidelberg.
- Fagin, B. (2000). An Ada interface to Lego Mindstorms. *ACM SIGAda Ada Letters*, 20(3), 20-40.
- Fagin, B. S., Merkle, L. D., & Eggers, T. W. (2001). Teaching computer science with robotics using Ada/Mindstorms 2.0. In *ACM SIGAda Ada Letters* (Vol. 21, No. 4, pp. 73-78). ACM.
- Fagin, B. (2003). Ada/Mindstorms 3.0. *Robotics & Automation Magazine, IEEE*, 10(2), 19-24.
- Fagin, B., & Merkle, L. (2003). Measuring the effectiveness of robots in teaching computer science. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 307-311). ACM.
- Fincher, S., & Petre, M. (Eds.). (2004). *Computer science education research*. Taylor & Francis.
- Flot, J., Schunn, C., Lui, A. & Sharp, R. (2012). Learning How to Program via Robot Simulation. *Robot Magazine November/December 2012*. Issue 37. ISSN: 1555-1016. 68-70.
- Flowers, T. R., & Gossett, K. A. (2002). Teaching problem solving, computing, and information technology with robots. *Journal of Computing Sciences in Colleges*, 17(6), 45-55.
- Flyvbjerg, B. (2006). Five misunderstandings about case-study research. *Qualitative inquiry*, 12(2), 219-245.
- Gable, G. G. (1994). Integrating case study and survey research methods: an example in information systems. *European Journal of Information Systems*, 3(2), 112-126.

- Garner, S. (2001). A tool to support the use of part complete solutions in the learning of programming. In *Proceedings of Informing Science 2001*. Available online: <http://proceedings.informingscience.org/IS2001Proceedings/pdf/GarnerEBKATool.pdf> (Accessed 1st October 2013).
- Garrett, A., & Thornton, D. (2005). A web-based programming environment for Lego Mindstorms robots. In *Proceedings of the 43rd Annual Southeast Regional Conference* (pp. 349-350). ACM.
- Gavan, C., & Anderson, M. (2013). Engaging Undergraduate Programming Students: Experiences using Lego Mindstorms NXT. In *SIGITE 2013*. ACM.
- Gee, J. P. (2003). What video games have to teach us about literacy and learning. New York: Palgrave Macmillan.
- Gibbert, M., & Ruigrok, W. (2010). The “What” and “How” of case study rigor: three strategies based on published work. *Organizational Research Methods*, 13(4), 710-737.
- Gibbs, G. R. (2007). Analyzing Qualitative Data (Book 6 of The SAGE Qualitative Research Kit). London: Sage Publications.
- Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012a). Coming to terms with Bloom: an online tutorial for teachers of programming fundamentals. In *Proceedings of the Fourteenth Australasian Computing Education Conference* (pp. 147-156). Australian Computer Society, Inc.
- Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012b). Over-confidence and confusion in using bloom for programming fundamentals assessment. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 147-152). ACM.
- Goldweber, M., Congdon, C., Fagin, B., Hwang, D., & Klassner, F. (2001). The use of robots in the undergraduate curriculum: experience reports. In *ACM SIGCSE Bulletin* (Vol. 33, No. 1, pp. 404-405). ACM.
- Grandell, L., Peltomäki, M., Back, R. J., & Salakoski, T. (2006). Why complicate things? Introducing programming in high school using Python. In *Proceedings of the 8th Australasian Conference on Computing Education* (pp. 71-80). Australian Computer Society, Inc.
- Griffiths, D., & Blat, P. (2004). ETui: scaffolding children’s reflection with behaviour based robots. In *Developing new technologies for young children*, 37-54. Trentham Books.
- Gross, P., & Powers, K. (2005). Evaluating assessments of novice programming environments. In *Proceedings of the First International Workshop on Computing Education Research* (pp. 99-110). ACM.
- Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. *Communications of the ACM*, 45(4), 17-21.
- Haberman, B., & Kolikant, Y. B. D. (2001). Activating “black boxes” instead of opening “zipper” - a method of teaching novices basic CS concepts. In *ACM SIGCSE Bulletin* (Vol. 33, No. 3, pp. 41-44). ACM.
- Haghighi, P. D. & Sheard, J. (2005). Summative Computer Programming Assessment Using Both Paper and Computer. In *Proceeding of Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences: Sharing Good Practices of Research, Experimentation and Innovation* (pp. 67-75). Amsterdam, The Netherlands.
- Henriksen, P., & Kölling, M. (2004). Greenfoot: Combining object visualisation with interaction. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications* (pp. 73-82). ACM.
- Hirst, A. J., Johnson, J., Petre, M., Price, B. A., & Richards, M. (2003). What is the best programming environment/language for teaching robotics using Lego Mindstorms?. *Artificial Life and Robotics*, 7(3), 124-131.

- Hove, S. E., & Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Symposium on Software Metrics*. IEEE.
- Howard, A. M., Park, C. H., & Remy, S. (2012). Using haptic and auditory interaction tools to engage students with visual impairments in robot programming activities. *IEEE Transactions on Learning Technologies*, 5(1), 87-95.
- Imberman, S. P., & Klibaner, R. (2005). A robotics lab for CS1. *Journal of Computing Sciences in Colleges*, 21(2), 131-137.
- Imberman, S. P., Klibaner, R., & Zelikovitz, S. (2007). Student Feedback on Robotics in CS1. In *AAAI Spring Symposium: Semantic Scientific Knowledge Integration* (pp. 65-68).
- Jackson, S. L. (2011). *Research methods and statistics: A critical thinking approach*. 4th Edition. CENGAGE Learning Custom Publishing.
- Jain, J., Cross II, J. H., Hendrix, T. D., & Barowski, L. A. (2006). Experimental evaluation of animated-verifying object viewers for Java. In *Proceedings of the 2006 ACM symposium on Software visualization* (pp. 27-36). ACM.
- Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, pp. 53-58).
- Jenkins, T., & Davy, J. (2002). Diversity and motivation in introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences*, 1(1).
- Jerez, J. M., Bueno, D., Molina, I., Urda, D., & Franco, L. (2012). Improving Motivation in Learning Programming Skills for Engineering Students. *International Journal of Engineering Education*, 28(1), 202.
- Jones, M. (2010). *An extended case study on the introductory teaching of programming*. Doctoral dissertation. University of Southampton.
- Kammer, T., Brauner, P., Leonhardt, T., & Schroeder, U. (2011). Simulating LEGO Mindstorms robots to facilitate teaching computer programming to school students. In *Towards Ubiquitous Learning* (pp. 196-209). Springer Berlin Heidelberg.
- Kasurinen, J., Purmonen, M., & Nikula, U. (2008). A study of visualization in introductory programming. In *Proceedings of the 20th Annual Meeting of Psychology of Programming Interest Group*, Lancaster, UK. PPIG.
- Kay, J. S., & Moss, J. G. (2012). Using Robots to Teach Programming to K-12 Teachers. In *2012 Frontiers in Education Conference Proceedings* (pp. 1-6). IEEE.
- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2), 83-137.
- Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, 50(7), 58-64.
- Kelly, J. F. (2010). *Lego Mindstorms NXT-G Programming Guide*. Apress.
- Kinnunen, P. & Malmi, L. (2006). Why students drop out CS1 course?. In *Proceedings of the Second International Workshop on Computing Education Research* (pp. 97-108). ACM.
- Kitchenham, B. (2004). Procedures for Undertaking Systematic Reviews. Joint Technical Report, Keele University TR/SE-0401 and NICTA 0400011T.

- Kitchenham, B. A., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01. Keele University and Durham University Joint Report.
- Kitchenham, B., Pretorius, R., Budgen, D., Brereton, P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering—a tertiary study. *Information and Software Technology*, 52(8), 792-805.
- Kitzinger, J. (1994). The methodology of focus groups: the importance of interaction between research participants. *Sociology of health & illness*, 16(1), 103-121.
- Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, 67-93.
- Klopfer, E., Osterweil, S., & Salen, K. (2010). Moving learning games forward: obstacles, opportunities, & openness. MA: MIT/The Education Arcade.
- Kochakornjarupong, D. (2010). A Web-based System Design for Enhancing Learning Problem Solving in Artificial Intelligence. *Special Issue of the International Journal of the Computer, the Internet and Management*, 18, 62-1.
- Kölling, M., & Rosenberg, J. (2001). Guidelines for teaching object orientation with Java. In *ACM SIGCSE Bulletin*, 33(3), pp. 33-36. ACM.
- Kottner, J., Audigé, L., Brorson, S., Donner, A., Gajewski, B. J., Hróbjartsson, A., Roberts, C., Shoukri, M. & Streiner, D. L. (2011). Guidelines for reporting reliability and agreement studies (GRRAS) were proposed. *International Journal of Nursing Studies*, 48(6), 661-671.
- Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into practice*, 41(4), 212-218.
- Kurkovsky, S. (2013). Mobile computing and robotics in one course: why not?. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education* (pp. 64-69). ACM.
- Ladd, B., & Harcourt, E. (2005). Student competitions and bots in an introductory programming course. *Journal of Computing Sciences in Colleges*, 20(5), 274-284.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 14-18). ACM.
- Lahtinen, E., & Ahoniemi, T. (2009). Kick-start activation to novice programming - A visualization-based approach. *Electronic Notes in Theoretical Computer Science*, 224, 125-132.
- Lajoie, S. P. (2005). Extending the scaffolding metaphor. *Instructional Science*, 33(5-6), 541-557.
- Lauwers, T., Nourbakhsh, I., & Hamner, E. (2009). CSbots: design and deployment of a robot designed for the CS1 classroom. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 428-432). ACM.
- Lauwers, T. (2013). The Finch, a robot for the CS classroom. *The Journal of Computing Sciences in Colleges*, 28(3), 104.
- Lee, A. S. (1989). A scientific methodology for MIS case studies. *MIS quarterly*, 13(1), 33-50.
- Lemone, K. A., & Ching, W. (1996). Easing into C++: experiences with RoBOTL. *ACM SIGCSE Bulletin*, 28(4), 45-49.

- López, R. N., Muñoz, R., Barría, M., & Pérez, F. Un Taller de Robótica para el Apoyo de la Enseñanza de Programación de Computadores Basado en Estilos de Aprendizaje. Available online: http://www.researchgate.net/publication/234166868_Un_Taller_de_Robtica_para_el_Apoyo_de_la_Enseanza_de_Programacin_de_Computadores_Basado_en_Estilos_de_Aprendizaje/file/72e7e515de85427e5b.pdf (Accessed 1 October 2013).
- Lumby, J. (2011). Enjoyment and learning: Policy and secondary school learners' experience in England. *British Educational Research Journal*, 37(2), 247-264.
- Lister, R., & Leaney, J. (2003). Introductory programming, criterion-referencing, and Bloom. In *ACM SIGCSE Bulletin* (Vol. 35, No. 1, pp. 143-147). ACM.
- Lister, R. (2004). Teaching Java first: experiments with a pigs-early pedagogy. In *Proceedings of the Sixth Australasian Conference on Computing Education* (pp. 177-183). Australian Computer Society, Inc.
- Liu, A., Newsom, J., Schunn, C., & Shoop, R. (2013). Students Learn Programming Faster through Robotic Simulation. *Tech Directions*, 72(8), 16-19.
- Major, L. (2010). Teaching Novices Programming Using Robotics: A Systematic Literature Review Protocol. Available online: http://www.dur.ac.uk/ebse/resources/studies/protocol/teaching_programming_with_robots_protocol_v1-4.pdf (Accessed 13 December 2012).
- Major, L., Kyriacou, T., & Brereton, O. P. (2011a). Systematic literature review: Teaching novices programming using robots. In *Proceedings of 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, (pp. 21-30). IET.
- Major, L., Kyriacou, T., & Brereton, P. (2011b). Experiences of prospective high school teachers using a programming teaching tool. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research* (pp. 126-131). ACM.
- Major, L., Kyriacou, T., & Brereton, O. P. (2012a). Systematic literature review: teaching novices programming using robots. *IET Software*, 6(6), 502-513.
- Major, L., Kyriacou, T. and Brereton, O.P. (2012b). Teaching Novices Programming Using a Robot Simulator: Case Study Protocol. In *Proceedings of the 24th Psychology of Programming Interest Group PPIG 2012* (pp. 93-104). London Metropolitan University, UK, PPIG.
- Marotto, M., Roos, J., & Victor, B. (2007). Collective virtuosity in organizations: A study of peak performance in an orchestra. *Journal of Management Studies*, 44(3), 388-413.
- Marsh, H. W. (1984). Students' evaluations of university teaching: Dimensionality, reliability, validity, potential biases, and utility. *Journal of Educational Psychology*, 76(5), 707.
- Martin, C., & Hughes, J. (2011). Robot Dance: Edutainment or Engaging Learning. In *Proceedings of the 23rd Psychology of Programming Interest Group PPIG 2011*, York, UK, 6-8 September. PPIG.
- Matlock-Hetzel, S. (1997). Basic Concepts in Item and Test Analysis. Texas A&M University.
- McAuley, A. (2012). BCS Consultations - Call to respond to: Removing the duty on maintained schools to follow the information and communication technology (ICT) Programmes of Study, Attainment Targets and statutory assessment arrangements. British Computer Society.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4), 125-180.
- McGill, M. M. (2012). Learning to program with personal robots: Influences on student motivation. *ACM Transactions on Computing Education (TOCE)*, 12(1), 4.

- McGrath, M. B., & Brown, J. R. (2005). Visual learning for science and engineering. *Computer Graphics and Applications*, 25(5), 56-63. IEEE.
- McWhorter, W. I., & O'Connor, B. C. (2009). Do LEGO Mindstorms motivate students in CS1?. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 438-442). ACM.
- Merriam, S. B. (1998). *Qualitative Research and Case Study Applications in Education*. Jossey-Bass Publishers, San Francisco, CA.
- Meyer, J., & Land, R. (2006). *Overcoming barriers to student understanding: Threshold concepts and troublesome knowledge*. Taylor & Francis.
- Miliszewska, I., & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Informing Science: International Journal of an Emerging Transdiscipline*. 4(1), 277-289.
- Moher, D., Pham, B., Jones, A., Cook, D. J., Jadad, A. R., Moher, M., Tugwell, P. & Klassen, T. P. (1998). Does quality of reports of randomised trials affect estimates of intervention efficacy reported in meta-analyses?. *The Lancet*, 352, 609-613.
- Moskal, B., Lurie, D., & Cooper, S. (2004). Evaluating the effectiveness of a new instructional approach. In *ACM SIGCSE Bulletin* (Vol. 36, No. 1, pp. 75-79). ACM.
- Nevalainen, S., & Sajaniemi, J. (2006). An experiment on short-term effects of animated versus static visualization of operations on program perception. In *Proceedings of the Second International Workshop on Computing Education Research* (pp. 7-16). ACM.
- Oliveira, N., Henriques, P. R., Da Cruz, D., Varanda Pereira, M. J., Mernik, M., Kosar, T., & Crepinsek, M. (2009). Applying program comprehension techniques to Karel robot programs. In *International Multi-conference on Computer Science and Information Technology (IMCSIT '09)*. (pp. 699-706). IEEE.
- Oppenheim, A. N. (2000). *Questionnaire design, interviewing and attitude measurement*. Continuum International Publishing Group.
- Paas, F., Renkl, A., & Sweller, J. (2004). Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional science*, 32(1), 1-8.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Basic Books.
- Pattis, R. E. (1981). *Karel the robot: a gentle introduction to the art of programming*. John Wiley & Sons, Inc.
- Pattis, R. E., Roberts, J., & Stehlik, M. (1995). *Karel the robot: A gentle introduction to the art of programming*. 2nd Edition. Wiley.
- Patton, M. Q. (Ed.). (2002). *Qualitative Research & Evaluation Methods*. SAGE.
- Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. In *ACM SIGCSE Bulletin* (Vol. 39, No. 4, pp. 204-223). ACM.
- Perkins, D. N., Schwartz, S., & Simmons, R. (1988). Instructional strategies for the problems of novice programmers. In *Teaching and Learning Computing* (pp. 153-178). Hillsdale.
- Petre, M., Fincher, S., & Tenenberg, J. et al. (2003). 'My Criterion is: Is it a Boolean?' A card-sort elicitation of students' knowledge of programming constructs. Technical Report (6-03). University of Kent, Computing Laboratory, UK.

- Petre, M., & Price, B. (2004). Using robotics to motivate 'back door' learning. *Education and Information Technologies*, 9(2), 147-158.
- Petre, M., & Blackwell, A. F. (2007). Children as unwitting end-user programmers. In *IEEE Symposium on Visual Languages and Human-Centric Computing VL/HCC 2007*. (pp. 239-242). IEEE.
- Pfleeger, S. L. (1994). Design and analysis in software engineering: the language of case studies and formal experiments. *ACM SIGSOFT Software Engineering Notes*, 19(4), 16-20.
- Piaget, J., & Cook, M. T. (1952). *The origins of intelligence in children*. Routledge & Kegan Paul Limited.
- Piaget, J. (1967). *The child's conception of the world*. Routledge & Kegan Paul.
- Poindexter, Sandra. (2003). Assessing active alternatives for teaching programming. *Journal of Information Technology Education*, 2, 257-265.
- Posey, E. (2012). Save our turtle robots?. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 157-158). ACM.
- Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V., & Carlisle, M. (2006). Tools for teaching introductory programming: what works?. *ACM SIGCSE Bulletin*, 38(1), 560-561.
- Prayaga, L., Prayaga, C., Wade, A., Suri, N., Whiteside, A., Hawthorne, J., Kulich, M. & Preucil, L. (2012). RILE–Robotic Interactive Learning Environment. In *Proceedings of the 3rd international conference on Robotics in Education RiE2012* (pp. 121-126). MatfyzPress, Czech Republic.
- Price, B. A., Richards, M., Petre, M., Hirst, A., & Johnson, J. (2003). Developing robotics e-teaching for teamwork. *International Journal of Continuing Engineering Education and Life Long Learning*, 13(1), 190-205.
- Raab, J., Rasala, R., & Proulx, V. K. (2000). Pedagogical power tools for teaching Java. In *ACM SIGCSE Bulletin* (Vol. 32, No. 3, pp. 156-159). ACM.
- Randolph, J. J., & Adviser-Julnes, G. (2007). Computer science education research at the crossroads: a methodological review of computer science education research, 2000-2005. Utah State University.
- Reiser, B. J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *The Journal of the Learning Sciences*, 13(3), 273-304.
- Riaz, M., Sulayman, M., Salleh, N., & Mendes, E. (2010). Experiences conducting systematic reviews from novices' perspective. In *Proceedings of EASE 2010* (pp. 1-10).
- Ring, B. A., Giordan, J., & Ransbottom, J. S. (2008). Problem solving through programming: motivating the non-programmer. *Journal of Computing Sciences in Colleges*, 23(3), 61-67.
- Roberts, T. S. (2006). The use of multiple choice tests for formative and summative assessment. In *Proceedings of the 8th Australasian Conference on Computing Education* (pp. 175-180). Australian Computer Society, Inc.
- Robins, A., Rountree, J., & Rountree, J. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*. 13(2), 132-172.
- Robson, C. (2011). *Real World Research*. 3rd Edition. John Wiley & Sons.
- Rosnow, R. L., & Rosenthal, R. (1997). *People studying people: Artifacts and ethics in behavioral research*. New York: WH Freeman.
- Rossmann, G. B., & Rallis, S. F. (2011). *Learning in the field: An introduction to qualitative research*. Sage.

- Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131-164.
- Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. Wiley.
- Saad, A., Kroutil, R., Strauser, E., & Reed, J. (2012). OssaBEST 2011: Increasing student inquiry into computer programming using robotics. In *Society for Information Technology & Teacher Education International Conference* (pp. 35-38).
- Sahami, M., Guzdial, M., McGettrick, A., & Roach, S. (2011). Setting the stage for computing curricula 2013: computer science-report from the ACM/IEEE-CS joint task force. In *Proceedings of the 42nd ACM technical symposium on Computer Science Education* (pp. 161-162). ACM.
- Salcedo, S. L., & Idrobo, A. M. O. (2011). New tools and methodologies for programming languages learning using the scribbler robot and Alice. In *Frontiers in Education Conference (FIE) 2011* (pp. F4G-1). IEEE.
- Sandy, G. A., Hall, M. J. J., & Bellucci, E. (2007). Indicative markers of leadership provided by ICT professional bodies in the promotion and support of ethical conduct. In *ACIS 2007: Proceedings of the 18th Australasian Conference on Information Systems* (pp. 92-99). ACIS.
- Satratzemi, M., Xinogalos, S., & Dagdilelis, V. (2003). An environment for teaching object-oriented programming: ObjectKarel. In *Proceedings of IEEE Advanced Learning* (pp. 342-343). IEEE.
- Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4), 557-572.
- Shaffer, D., Doube, W., & Tuovinen, J. (2003). Applying cognitive load theory to Computer Science Education. In *15th Workshop of the Psychology of Programming Interest Group*. Keele University, UK. PPIG.
- Shang, A., Huwiler-Müntener, K., Nartey, L., Jüni, P., Dörig, S., Sterne, J. A., Pewsner, D. & Egger, M. (2005). Are the clinical effects of homoeopathy placebo effects? Comparative study of placebo-controlled trials of homoeopathy and allopathy. *The Lancet*, 366, 726-732.
- Sheard, J., & Hagan, D. (1998). Our failing students: a study of a repeat group. In *ACM SIGCSE Bulletin* (Vol. 30, No. 3, pp. 223-227). ACM.
- Siegel, S. (1957). Nonparametric statistics. *The American Statistician*, 11(3), 13-19.
- Silver Pacuilla, H., Brown, S., Overton, C., & Stewart, A. (2011). Assistive technology research matters. *Washington, DC: American Institutes for Research*.
- Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N. K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering*, 31(9), 733-753.
- Sklar, E., Eguchi, A., & Johnson, J. (2003). RoboCupJunior: learning with educational robotics. In *RoboCup 2002: Robot Soccer World Cup VI* (pp. 238-253). Springer Berlin Heidelberg.
- Sklar, E., Parsons, S., & Azhar, M. Q. (2006). Robotics Across the Curriculum. Technical Report. Department of Computer and Information Science. Brooklyn College, City University of New York, 2006.
- Smith, M. K., Wood, W. B., & Knight, J. K. (2008). The genetics concept assessment: a new concept inventory for gauging student understanding of genetics. *CBE-Life Sciences Education*, 7(4), 422-430.
- Solin, P. (2013). Learn how to think with Karel the Robot. NCLAB Public Computing. FEMhub Inc.

- Song, S., & Yu, G. (2011). Education reform on software programming courses for non-IT-majoring undergraduates. In *6th International Conference on Computer Science & Education (ICCSE 2011)*, (pp. 715-719). IEEE.
- Soule, T., & Heckendorn, R. B. (2011). COTSBots: computationally powerful, low-cost robots for Computer Science curriculums. *Journal of Computing Sciences in Colleges*, 27(1), 180-187.
- Squires, D., & Preece, J. (1999). Predicting quality in educational software: Evaluating for learning, usability and the synergy between them. *Interacting with computers*, 11(5), 467-483.
- Suliman, A., & Nazeri, S. (2012). The effectiveness of teaching and learning programming using Embedded System. In *8th International Conference on Information Science and Digital Content Technology (ICIDT), 2012* (pp. 32-36). IEEE.
- Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D., & Balch, T. (2009). Personalizing CS1 with robots. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 433-437). ACM.
- Stewart, L. G., & White, M. A. (1976). Teacher comments, letter grades, and student performance: What do we really know?. *Journal of Educational Psychology*, 68(4), 488.
- Stiggins, R. J. (2005). Student-involved assessment for learning. Upper Saddle River, NJ: Prentice Hall.
- Szweda, L., Wilusz, D., & Flotyński, J. (2012a). Application of NXT based robots for teaching java-based concurrency. In *E-Learning and Games for Training, Education, Health and Sports* (pp. 54-64). Springer Berlin Heidelberg.
- Szweda, L., Flotyński, J., Wilusz, D., & Dabrowski, P. (2012b). Robot simulator facilitating the education of concurrent programming. In *Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS) 2012* (pp. 891-896). IEEE.
- Talbot, H. (2000). wxPython, a GUI Toolkit. *Linux Journal*, 74(5).
- Thota, N., & Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, 20(2), 103-127.
- Thurmond, V. A. (2001). The point of triangulation. *Journal of Nursing Scholarship*, 33(3), 253-258.
- Untch, R. H. (1990). Teaching programming using the Karel the robot paradigm realized with a conventional language. Available online: <http://www.mtsu.edu/untch/karel/karel90.pdf> (Accessed 1 October 2013).
- Varma, S. (2006). Preliminary item statistics using point-biserial correlation and p-values. 16(7). *Educational Data Systems Inc.* Morgan Hill CA.
- Verner, J. M., Sampson, J., Tasic, V., Bakar, N. A. A., & Kitchenham, B. A. (2009). Guidelines for industrially - based multiple case studies in software engineering. In *Third International Conference on Research Challenges in Information Science 2009* (pp. 313-324). IEEE.
- Watson, D. G., & Harrison, T. V. (2012). Teaching Control Programming and Problem-Solving Techniques with Karel++. *Journal of Agricultural Systems, Technology, and Management*, 23, 13-25.
- Weintrop, D., & Wilensky, U. (2013). Robobuilder: a computational thinking game. In *SIGCSE 2013* (p. 736). ACM.
- Wellman, B. L., Anderson, M., & Vrbsky, S. (2009). PREOP as a tool to increase student retention in CS. *Journal of Computing Sciences in Colleges*, 25(2), 167-175.
- Wells, D. (2012). Computing in schools: time to move beyond ICT. *Research in Secondary Teacher Education*, 2(1), 8-13.

- Woodall, P. and Brereton, P. (2006). Conducting a systematic literature review from the perspective of a PhD student. In *Proceedings of the 10th International Conference on Evaluation and Assessment in Software Engineering (EASE '06)*, BCS-eWiC, 138.
- Wong, K. (2001). Teaching programming with LEGO RCX robots. In *Proceedings of the 18th Information Systems Education Conference (ISECON 2001)*. Cincinnati, Ohio, USA.
- Wu, C. C., Tseng, I. C., & Huang, S. L. (2008). Visualization of program behaviors: Physical robots versus robot simulators. In *Informatics Education-Supporting Computational Thinking* (pp. 53-62). Springer Berlin Heidelberg.
- Wulf, T. (2005). Constructivist approaches for teaching computer programming. In *Proceedings of the 6th Conference on Information Technology Education* (pp. 245-248). ACM.
- Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. *ACM Inroads*, 2(4), 71-76.
- Yin, R. K. (1999). Enhancing the quality of case studies in health services research. *Health Services Research*, 34(5), 1209.
- Yin, R. K., (2003). *Case Study Research: Design and Methods*. 3rd Edition. Sage.
- Yin, R. K. (2009). *Case study research: Design and methods*. 4th Edition. Sage.
- Yousoof, M., Sapiyan, M., & Kamaluddin, K. (2006). Reducing cognitive load in learning computer programming. In *Proceedings of the World Academy of Science, Engineering and Technology*, 12.
- Yousoof, M., Sapiyan, M., & Kamaluddin, K. (2007). Measuring cognitive load - A solution to ease learning of programming. In *Proceedings of World Academy of Science, Engineering and Technology*, 20.
- Yu, X. (2012). Using LEGO Mindstorms in the undergraduate curriculum of IT. In *Proceedings of International Symposium on Information Technology in Medicine and Education ITME 2012* (pp. 270-273). IEEE.
- Zannier, C., Melnik, G., & Maurer, F. (2006). On the success of empirical studies in the international conference on software engineering. In *Proceedings of the 28th International Conference on Software Engineering* (pp. 341-350). ACM.

Appendix A1 – Articles Included in the SLR

[Barnes02]: Barnes, D.J.: ‘Teaching Introductory Java through LEGO MINDSTORMS models’. Proc. 33rd SIGCSE Technical Symposium on Computer Science Education, New York, NY, USA, 2002, pp. 147–151

[Becker01]: Becker, B.W.: ‘Teaching CS1 With Karel the Robot in Java’. Proc. 32nd SIGCSE Technical Symposium on Computer Science Education, New York, NY, USA, 2001, pp. 50–54

[Bell08]: Bell, S., Heeler, L., and Heeler, P.: ‘A Preliminary Report on the Use of Robots with Elementary School Students’, J. Comput. Small Coll., 2008, 23, (4), pp. 263–268

[Borge04]: Borge, R., Fjuk, A., and Groven, A.K.: ‘Using Karel J Collaboratively to Facilitate Object-Oriented Learning’. Proc. IEEE International Conference on Advanced Learning Technologies, August 2004, pp. 580–584

[Brauner10]: Brauner, P., Leonhardt, T., Ziee, M., and Schroeder, U.: ‘The Effect of Tangible Artefacts, Gender and Subjective Technical Competence on Teaching Programming to Seventh Graders’. Proc. 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives, Zurich, Switzerland, January 2010, pp. 61–71

[Buck01]: Buck, D., and Stucki, D.J.: ‘JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum’. Proc. 32nd SIGCSE Technical Symposium on Computer Science Education, Charlotte, North Carolina, USA, February 2001, pp. 16–20

[Cliburn06]: Cliburn, D.C.: ‘Experiences with the LEGO Mindstorms throughout the Undergraduate Computer Science Curriculum’. Proc. Frontiers in Education Conference 36th Annual, San Diego, California, USA, October 2006, pp. 1–6

[Czejdo09]: Czejdo, B.D., and Bhattacharya, S.: ‘Programming Robots With State Diagrams’, J. Comput. Small Coll., 2009, 24, (5), pp. 19–26

[Enderle08]: Enderle, S.: ‘Grape: Graphical Robot Programming for Beginners’. Proc. Research and Education in Robotics - EUROBOT, Heidelberg, Germany, 2008, pp. 180–192

[Fagin01]: Fagin, B.S.: ‘Teaching Computer Science with Robotics using Ada/Mindstorms 2.0’. In Proceedings of the 2001 Annual ACM SIGAda International Conference, pp 73–78. ACM.

[Fagin03a]: Fagin, B.: ‘Measuring the Effectiveness of Robots in Teaching Computer Science’. Proc. 34th SIGCSE Technical Symposium on Computer Science Education, Reno, Nevada, USA, February 2003, pp. 307–311

[Fagin03b]: Fagin, B.: ‘Ada/Mindstorms 3.0’, IEEE Robotics & Automation Magazine, 2003, 10, (2), pp. 19–24.

[Flowers02]: Flowers, T. R., and Gossett, K. A.: ‘Teaching Problem Solving, Computing, and Information Technology with Robots’, J. Comput. Small Coll., 2002, 17, (6), pp. 45–55

[Garrett05]: Garrett, A., and Thornton, D.: ‘A Web-Based Programming Environment for LEGO Mindstorms Robots’. Proc. 43rd Annual Southeast Regional Conference, Kennesaw, Georgia, Alabama, USA, March 2005, pp. 349–350

[Goldweber01]: Goldweber, M., Congdon, C., Fagin, B., Hwang, D., and Klassner, F.: ‘The Use of Robots in the Undergraduate Curriculum: Experience Reports’. Proc. 32nd SIGCSE Technical Symposium on Computer Science Education, Charlotte, North Carolina, USA, February 2001, pp. 404–405

[Hirst03]: Hirst, A.J., Johnson, J., Petre, M., Price, B.A., and Richards, M.: ‘What is the best programming environment/language for teaching robotics using Lego Mindstorms?’, Artificial Life and Robotics, 2003, 7, (3), pp. 124–131

[Imberman05]: Imberman, S.P., and Klibaner, R.: ‘A Robotics Lab for CS1, J. Comput. Small Coll., 2005, 21, 2, pp. 131 – 137

[Imberman07]: Imberman, S.P, Klibaner, R., and Zelikovitz, S.: ‘Student Feedback on Robotics In CS1’. <http://www.cs.hmc.edu/roboteducation/papers2007/c1> (Accessed 11 March 2010)

[Jadud03]: Jadud, M.C. and Schleter, J.: ‘Little Languages and Little Robots’. http://www.jadud.com/people/mcj/_les/2003-PPIG-II DBLP (Accessed 11 March 2010)

- [**Kurebay06**]: Kurebayashi, S., Kamada, T., and Kanemune, S.: ‘Learning Computer Programming with Autonomous Robots’, *Informatics Education: The Bridge Between Using and Understanding Computers*, 2006, 4226, *Lecture Notes in Computer Science*, pp. 138–149
- [**Ladd05**]: Ladd, B., and Harcourt, E.: ‘Student Competitions and Bots in an Introductory Programming Course’, *J. Comput. Small Coll.*, 2005, 20, (5), pp. 274–284
- [**Lauwers09**]: Lauwers, T., Nourbakhsh, I., and Hamner, E.: ‘CSbots: Design and Deployment of a Robot Designed For the CS1 Classroom’. *Proc. 40th SIGCSE Technical Symposium on Computer Science Education*, Chattanooga, TN, USA, March 2009, pp. 428–432
- [**Lawhead02**]: Lawhead, P.B., Duncan, M.E., Bland, C.G., Goldweber, M., Schep, M., Barnes, D.J., and Hollingsworth, R.G.: ‘A Road Map For Teaching Introductory Programming Using LEGO Mindstorms Robots’. *Proc. ITiCSE-WGR '02: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, Aarhus, Denmark, June 2002, pp. 191–201
- [**Lemone96**]: Lemone, K.A., and Ching, W.: ‘Easing into C++: Experiences with RoBOTL’, *SIGCSE Bull.*, 1996, 28, (4), pp. 45–49
- [**Mcwhorter09**]: McWhorter, W.I., and O'Connor, B.C.: ‘Do LEGO Mindstorms Motivate Students in CS1?’ *Proc. SIGCSE ACM Technical Symposium on Computer Science Education*, Chattanooga, TN, USA, March 2009, pp. 438–442
- [**Petre04**]: Petre, M., and Price, B.: ‘Using Robotics to Motivate ‘Back Door’ Learning’, *Education and Information Technologies*, 2004, 9, (2), 147–158
- [**Sartzatsemi03**]: Sartzatsemi, M., Xinogalos, S., and Dagdilelis, V.: ‘An Environment for Teaching Object-Oriented Programming: objectKarel’. *Proc. 3rd IEEE International Conference on Advanced Learning Technologies*, Athens, Greece, July 2003, pp. 342–343


- [Sartatzemi05]:** Sartatzemi, M., Dagdilelis, V., and Kagani, K.: ‘Teaching Programming with Robots: A Case Study on Greek Secondary Education’, *Advances in Informatics*, 2005, 3746, *Lecture Notes in Computer Science*, pp. 502–512
- [Sklar06]:** Sklar, E., Parsons, S., and Azhar, M.Q.: ‘Robotics Across the Curriculum’. Technical Report, Dept of Computer and Information Science Brooklyn College, 2006
- [Summet09]:** Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D., and Balch, T.: ‘Personalizing CS1 with Robots’. In *Proc. 40th ACM Technical Symposium on Computer Science Education*, Chattanooga, TN, USA, March 2009, pp. 433–437
- [Vandelden08]:** van Delden, S., and Zhong, W.: ‘Effective Integration of Autonomous Robots into an Introductory Computer Science Course: a Case Study’, *J. Comput. Small Coll.*, 2008, 23, (4), pp. 10–19
- [Weiss08]:** Weiss, R., and Overcast, I.: ‘Finding Your Bot-Mate: Criteria for Evaluating Robot Kits for Use in Undergraduate Computer Science Education’, *J. Comput. Small Coll.*, 2008, 24, (2), pp. 43–49
- [Wellman09]:** Wellman, B.L., Anderson, M., and Vrbsky, S.V.: ‘PREOP as a Tool to Increase Student Retention in CS’, *J. Comput. Small Coll.*, 2009, 25, (2), pp. 167–175
- [Williams03]:** Williams, A.B.: ‘The Qualitative Impact of Using LEGO MINDSTORMS Robots to Teach Computer Engineering’, *IEEE Transactions on Education*, 2003, 46, (1), p. 206
- [Wong01]:** Wong, K.: ‘Teaching Programming with LEGO RCX Robots’, *Proc. 18th Information Systems Education Conference*, Cincinnati, Ohio, USA, November 2001
- [Wu08]:** Wu, C., Tseng, I., and Huang, H.: ‘Visualization of Program Behaviours: Physical Robots Versus Robot Simulators’, *Informatics Education - Supporting Computational Thinking*, 2008, 5090, *Lecture Notes in Computer Science*, pp. 53–62

Appendix A2 – Electronic Search Results

	<i>ACM</i>	<i>CiteSeerX</i>	<i>EBSCO</i>	<i>ERIC</i>	<i>IEEE</i>	<i>ISI</i>	<i>Keele</i>	<i>AEI</i>	<i>BEI</i>
	Number of Papers Returned During Each Search								
“amateur”	20	2	2	1	7	0	0	1	0
“beginner”	42	5	1	0	4	1	0	1	1
“first time”	575	3	1	8	2	77	0	0	0
“introductory”	120	71	13	21	99	16	0	53	2
“novice”	93	69	6	10	209	6	0	35	2
“teaching”	34	111	17	21	75	97	0	18	6
“learning”	26	346	21	23	58	258	0	16	3
<i>Abstract</i>	379	385	94	0	12	336	0	9	0
<i>Title</i>	4	25	7	1	4	10	0	0	0
Total Papers Returned	1293	1017	162	85	470	801	0	133	14
Included in SLR (before removal of duplicates)	20	8	1	0	3	8	0	0	0

Appendix A3 – Exploratory Questionnaires

Exploratory Questionnaire One (EQ1)



KEELE
UNIVERSITY

Questionnaire—Pre-Workshop

This questionnaire is designed to gauge your opinions on the topic of computer programming and to identify the level of your programming ability. All questionnaire responses will remain anonymous and will not be identifiable by either the researchers involved nor during any future dissemination activities.

Thank you for your participation.

Section One: Your Programming Background

Q1. Which of the following statements relate to your programming experience? (Tick all that apply)

I have no past programming experience— If selected skip to Section Two (overleaf)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I have programmed because a University (or other) course required me to	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I have programmed as a past job required me to	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I have previously programmed out of personal interest (e.g. hobby, for fun)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I previously programmed to learn a new skill	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Other (Please Specify) _____				

Q2. Please write which computer languages you have used and to what level (Please Specify and Circle)

_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert
_____	Beginner	Intermediate	Expert

Q3. Which of the following best describes your past programming experience? (Tick only one box)

Didn't like programming	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Indifferent	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Enjoyed programming	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Q4. Did you find programming challenging? (Tick only one box)

Challenging	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Neither challenging nor trivial	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Trivial	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Q5. The following is a list of programming concepts. Please circle those concepts which you have used previously in programming code. (Circle all which you have used previously in programming code)

Program Methods/Functions/Subroutines	Arrays
Object-oriented Programming	File Input/output
Iteration (e.g. while loops, for loops)	Variables
Selection (if, if ... else)	Strings
Recursion (when a function calls itself)	Multithreading
Expressions	Graphical User Interfaces (GUIs)

Section Two: Your Thoughts On Programming

Q6. Do you believe that it is important to teach basic programming concepts to all High School students enrolled on an IT or Computing course at some stage during their time at School? (Tick only one box)

Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
No	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Q7. With your current knowledge would you be confident teaching High School students about introductory programming concepts? (Tick only one box)

Yes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
No	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Q8. How difficult do you think it would be to teach elementary programming concepts to High School students using your current knowledge? (Tick only one box)

Difficult	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Neither difficult nor easy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Section Three: Miscellaneous

Q9. What is your gender?

Male	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Female	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Exploratory Questionnaire Two (EQ2)



KEELE
UNIVERSITY

Questionnaire—Post-Workshop

Section One: Your Experience Today

Q1. In regards to today's programming experience, which of the following best describes your enjoyment of today's session? *(Tick only one box)*

Not Enjoyable	<input type="checkbox"/>
Indifferent	<input type="checkbox"/>
Enjoyable	<input type="checkbox"/>

Q2. In regards to today's session, which of the following best describes how challenging the programming tasks have been? *(Tick only one box)*

Difficult	<input type="checkbox"/>
Neither easy nor difficult	<input type="checkbox"/>
Easy	<input type="checkbox"/>

Q3. Do you believe that the Robot Simulator offers an effective method of introducing basic programming concepts, which you have been taught today, to novice programming students? *(Tick only one box)*

Yes	<input type="checkbox"/>
No	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>

Q4. In regards to the effectiveness of teaching introductory programming, rate the different elements of today's workshop on a scale of 1 (Not at all effective) to 5 (Extremely effective). *(Please circle your choice)*

	<i>(Not at all effective)</i>				<i>(Extremely effective)</i>
Robot Simulator	1	2	3	4	5
Teaching	1	2	3	4	5
Presentation	1	2	3	4	5
Environment	1	2	3	4	5

Section Two: Your Thoughts On Programming

Q5. With your current knowledge would you be confident teaching High School students about introductory programming concepts? *(Tick only one box)*

Yes	<input type="checkbox"/>
No	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>

Q6. How difficult do you think it would be to teach elementary programming concepts to High School students using your current knowledge? *(Tick only one box)*

Difficult	<input type="checkbox"/>
Neither Difficult nor Easy	<input type="checkbox"/>
Easy	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>

Q7. Would you consider using the Robot Simulator as a tool to teach programming in your own lessons in the future? *(Tick only one box)*

Yes	<input type="checkbox"/>
No	<input type="checkbox"/>
Not Sure	<input type="checkbox"/>

Q8. In relation to the Robot Simulator what aspects (if any) did you like the most? *(Please specify)*

Q9. In relation to the Robot Simulator what aspects (if any) did you like the least? *(Please specify)*

Q10. In relation to this workshop session do you have any general comments? *(Please specify)*

Q11. What is your gender?

Male	<input type="checkbox"/>
Female	<input type="checkbox"/>

Appendix A4 – Ethical Approval Documentation



K E E L E
UNIVERSITY

ACADEMIC SERVICES DIRECTORATE
RESEARCH AND ENTERPRISE SERVICES

5 November 2010

Mr Louis Major
Room 141
Colin Reeves Building

Dear Louis,

Re: Innovative Methods for Teaching Computer Programming

Thank you for submitting your revised project for review.

I am pleased to inform you that your project has been approved by the Ethics Review Panel.

Amendments to your project after a favourable ethical opinion has been given you must notify the Ethical Review Panel via Michele Dawson.

If you have any queries, please do not hesitate to contact Michele Dawson in writing to m.dawson@usa.keele.ac.uk

Yours sincerely

Dr Nicky Edelstyn
Chair – Ethics Review Panel

cc: RI Manager



THE QUEEN'S
ARMS
For Orders and Honours bestowed
2009

T: +44(0)1782 733371 F: +44(0)1782 733340 E: usa@usa.keele.ac.uk
W: www.keele.ac.uk/businessenterprise
Deputy Hoeglin Building, Room 115

Keele University, Staffordshire, ST5 5BG, United Kingdom
T: +44(0)1782 732000 W: www.keele.ac.uk
Printed on 100% recycled paper

Appendix A5 – Trainee Teacher (Case One) Questionnaires

Trainee Teacher Questionnaire One (TTQ1)



Questionnaire - Pre-Workshop

Section One: Your Programming Background

1. Have you learnt any programming languages? If not go to Section 2 overleaf. If Yes:

a. Please write which computer languages you have learnt and to what level *(Please specify and circle)*

_____	Beginner	Competent	Expert
_____	Beginner	Competent	Expert
_____	Beginner	Competent	Expert
_____	Beginner	Competent	Expert
_____	Beginner	Competent	Expert

b. Which language did you learn most recently? *(Please specify)* _____

i. How did you learn this language? *(Select one)*

Self-taught _____ Part of a course/education program _____ Other _____

ii. How much time did you spend learning this language? *(Please specify)* _____

iii. How well do you feel you learnt this language? *(Circle one)*

Learned very little _____ Became familiar with most programming concepts _____ Became competent _____ Became expert _____

c. Which of the following best describes your past programming experience? *(Circle one)*

Didn't Like _____ Indifferent _____ Enjoyed _____

d. The following is a list of programming concepts. Please circle those concepts which you have previously used in programming code? *(Please circle all that apply)*

Program Methods/Functions/Subroutines	Arrays
Object-oriented Programming	File Input/output
Iteration (e.g. while loops, for loops)	Variables
Selection (if, if ... else)	Strings
Recursion (when a function calls itself)	Multithreading
Expressions	Graphical User Interfaces (GUIs)

PLEASE TURN OVER FOR SECTION TWO...



Section Two: Your Thoughts on Programming

2. Do you associate any issues with learning to program? If Yes please identify up-to three of these issues:

3. Do you associate any stereotypes with learning to program? If Yes please identify up-to three of these stereotypes:

4. Do you believe that it is important to teach basic programming concepts to all High School students enrolled on an IT or Computing course at some stage during their time at School? *(Circle one)*

Yes _____ No _____ Not Sure _____

5. With your current knowledge would you be confident teaching High School students about introductory programming concepts? *(Circle one)*

Yes _____ No _____ Not Sure _____

6. How difficult do you think it would be to teach elementary programming concepts to High School students using your current knowledge? *(Circle one)*

Difficult _____ Neither difficult or easy _____ Easy _____ Not Sure _____

7. What is your gender? *(Circle one)*

Male _____ Female _____

8. What is your participant code number? *(Please specify)*

END OF QUESTIONNAIRE



Trainee Teacher Questionnaire Two (TTQ2)



Questionnaire - Post-Workshop

Section One: Your Workshop Experience

1. In regards to your programming experience during the workshop, which of the following best describes your enjoyment? *(Circle one)*
- | | | |
|---------------|-------------|-----------|
| Not Enjoyable | Indifferent | Enjoyable |
|---------------|-------------|-----------|
2. In regards to the workshop session, which of the following best describes how difficult the programming tasks have been? *(Circle one)*
- | | | |
|-----------|---------------------------|------|
| Difficult | Neither difficult or easy | Easy |
|-----------|---------------------------|------|
3. Do you believe that the Robot Simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
4. Please specify up-to three things you *liked* about the Simulator *(Please specify)*
- _____
- _____
- _____
5. Please specify up-to three things you *did not like* about the Simulator *(Please specify)*
- _____
- _____
- _____
6. In regards to the effectiveness of teaching introductory programming, rate the different elements of the workshop on a scale of 1 (Not at all effective) to 5 (Extremely effective) *(Please circle your choices)*
- | | | | | | |
|---------------------|---|---|---|---|---|
| Robot Simulator | 1 | 2 | 3 | 4 | 5 |
| Programming Support | 1 | 2 | 3 | 4 | 5 |
| Presentation | 1 | 2 | 3 | 4 | 5 |
| Environment | 1 | 2 | 3 | 4 | 5 |

PLEASE TURN OVER FOR SECTION TWO...



Section Two: Your Thoughts on Programming

7. If you have previously learnt a programming language, was your *previous* introduction to basic programming... *(Circle one)*
- | | | | | | |
|---------------------|----------------|------------------------------|----------------|---------------------|----------------------------|
| Much less effective | Less effective | About the same effectiveness | More effective | Much more effective | Little previous experience |
|---------------------|----------------|------------------------------|----------------|---------------------|----------------------------|
8. Do you believe that it is important to teach basic programming concepts to all High School students enrolled on an IT or Computing course at some stage during their time at School? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
9. With your current knowledge would you be confident teaching High School students about introductory programming concepts? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
10. How difficult do you think it would be to teach elementary programming concepts to High School students using your current knowledge? *(Circle one)*
- | | | | |
|-----------|---------------------------|------|----------|
| Difficult | Neither difficult or easy | Easy | Not Sure |
|-----------|---------------------------|------|----------|
11. Do you believe that programming should be an important part of the National Curriculum in Schools? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
12. Would you consider using the Robot Simulator as a tool to teach programming in your own lessons in the future? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
13. Do you have any general comments about the Robot Simulator or the session in general? *(Please specify)*
- _____
- _____
- _____
- _____
14. What is your gender? *(Circle one)*
- | | |
|------|--------|
| Male | Female |
|------|--------|
15. What is your participant code number? *(Please indicate)*
- _____



Appendix A6 – Trainee Teacher (Case One) Workshop Log Transcripts

Trainee Teacher Workshop One (TTW1) – Day One

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

P1 arrived around 30 minutes late for the Workshop. P18 arrived around 30 minutes late for the Workshop. P15 arrived around 20 minutes late for the Workshop. P15 left the Workshop for around 20 minutes after one hour. The participant said they had an important matter to attend to. The late arrival of these participants is not considered to have been an issue due to the early part of the workshop being dedicated to the setting up of the research environment and the completion of informed consent forms.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants appeared to be “on task” during the workshop. Participants appeared to enjoy the session and seemed eager to learn and use Kebot. Discussion amongst participants also occurred at several points during the day with some trainees eager to help their peers who needed assistance.

Did participants seem enthusiastic throughout the session? If not, why?

The majority of participants managed to complete the programming tasks they were set. However, P3, P4, P12 and P15 seemed to fall behind during the Workshop a little more and time was required to explain programming concepts in greater depth to these participants. In particular P15 struggled. As the majority of the group progressed well in their coding, however, it was difficult to always ensure that these participants completely understood and completed the task at hand despite the workshop leader’s best efforts. Nonetheless, these participants appeared to enjoy the session and no negative feedback was provided. On the contrary, the general perception these trainees gave was positive.

What aspects of the workshop/programming concepts did participants struggle to grasp?

Missing curly braces caused most participants an issue at some point as did general if...else syntax. As the tasks advanced, and participants gained more experience using these concepts, fewer problems were encountered. Forgetting to repeat the variable’s name when wanted to compare a variable in the condition of an if-statement also initially created issues and produced syntax errors. The way in which knowledge was gradually built-up received praise from two trainees.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable. Indeed, the simplicity of the approach was praised by two participants who stated they appreciated the easy-to-use nature the software.

Were any concerns raised about the nature of assessment used during the workshop?

Several trainees commented that the layout of the first in-workshop programming exercise caused some confusion and that it was difficult to distinguish between some questions and potential responses.

Was a considerable amount of time spent diverging from workshop related activities?

No. Very little time was spent diverging from Workshop related activities. Discussion appeared to relate mainly to programming.

Any other points that need to be noted?

On the whole the first part of the workshop went well and all participants seemed to be looking forward to attending the second (and concluding part) of the session. Participants made good progress during the workshop. It should be noted, however, that when completing the in-workshop programming exercise that the majority of trainees finished this task extremely quickly. Several comments were made in regards to the layout of the instrument and this appeared to confuse some trainees. The number of participants involved must also be noted as, at several points, a number of trainees required assistance at the same time and this meant that the workshop had to be delayed.

Trainee Teacher Workshop One (TTW1) – Day Two

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

P3 was only able to take a moderate part in the programming tasks during Day 2 as she complained that she had suffered an injury to her wrist overnight. This meant that P3 did not complete the post-workshop programming test. However, P3 did sit in for the entirety of the session and completed the in-workshop programming exercise and the post-workshop questionnaire. P15 arrived around 30 minutes late for the Workshop. P18 arrived around 45 minutes late for the Workshop. The participant said that they had travelled a long way to attend. P14 did not attend Day 2 of the Workshop although he attended Day One in its entirety.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants appeared to be “on task” during the workshop despite the concepts introduced gradually increasing in complexity. When a participant ran into difficulty on several occasions it was noted how their peers around them were quick to assist. Four participants appeared to have difficulty grasping the programming concepts they were using throughout the session. These were P4, P12, P15 and P18. Attempts were made to spend extra time discussing programming issues with these participants although the rest of the group could not be neglected in order to maintain the pace of the session (so other participants did not lose interest). On the whole all participants appeared to be “on task” during the Workshop and remained focused on the tasks at hand. P3 was unable to take a substantial part in the programming exercises during the Workshop due to a wrist injury. When taking a break from coding P3 took an interest in the programming exercises by watching others around her code. The large size of the group resulted in several participants requiring assistance at the same point during the workshop which at times delayed proceedings.

Did participants seem enthusiastic throughout the session? If not, why?

The overwhelming impression I got from participants was that the session was both helpful and enjoyable. Several participants asked for the Workshop slides and kebot to be sent to them after the Workshop ended. P2, P13 and P17 seemed particularly enthusiastic during the session and were keen to always expand on the tasks they were given by eliciting additional behaviour from the robotic agents they were programming. Specifically P2 was seen to write down large quantities of code by hand for future reference. Similarly P5 and P11 seemed to effortlessly complete the programming tasks they were set. No participant was heard to voice discontent during the workshop even when they had difficulty completing a particular programming task.

What aspects of the workshop/programming concepts did participants struggle to grasp?

Arrays and For Loops caused some participants difficulty although most appeared to grasp at least the underlying concepts. It is considered that further practice of these concepts would likely help to reinforce the programming syntax in participants' minds. The introduction to Arrays and For Loops was not intended to be comprehensive as these concepts were not specifically identified by the ACM/IEEE Joint Task Force.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

The second in-workshop programming exercise appeared to be completed extremely quickly by the majority of trainees. In regards to the post-workshop programming exercise, several participants may have not given 100% effort towards the end of the exercises (particularly Task 4). This may have been because several participants finished the Tasks and then left the Lab which prompted some of the remaining participants to lose concentration and to hastily attempt the final programming task. Task Two also caused significant issues and the nature of this exercise caused substantial confusion. Many trainees appeared to attempt the task without understanding what was expected of them and voiced such concerns after the session.

Was a considerable amount of time spent diverging from workshop related activities?

No. Participants spent virtually no time diverging from Workshop related activities. Throughout the session even the discussions between participants mainly revolved around the programming tasks at hand.

Any other points that need to be noted?

On the whole I got the impression that the workshop was a success. In a discussion two participants (P8 and P10) specifically stated how they believe interactive sessions (like the Workshop) were much more effective and enjoyable than traditional lecture based delivery. Even those participants who appeared to struggle with the programming tasks (P4, P12, P15 and P18) appeared to remain interested in the session, even if they had difficulty with the actual coding at times. The workshop slides and Kobot simulator were requested by several participants after the session. These commented how they felt they would be able to make use of this material in their own future lessons.

Trainee Teacher Workshop Two (TTW2) – Day One

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

All participants arrived on time. 5 participants took part in the workshop as was expected.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants appeared to be “on task” throughout the workshop. Participants seemed to enjoy the session and were eager to use the simulator. The group seemed knowledgeable about programming in general (although not all participants had previously used Java).

Did participants seem enthusiastic throughout the session? If not, why?

All participants appeared to complete the programming tasks set. None of the participants appeared to struggle during the session although P2 asked for the most assistance. P1 seemed to be the most inquisitive and asked many questions in regards to the nature of the simulator. P4 and P5 quickly adapted to the simulator environment and the approach of the workshop and were noted on several occasions to try and elicit additional behaviour out of the robotic agents. One participant remained behind after the session to discuss programming concepts (P3).

What aspects of the workshop/programming concepts did participants struggle to grasp?

Few consistent syntax errors repeatedly caused problems during the workshop. P2 discussed how she sometimes felt confused by the Java syntax although this did not cause significant problems at any point.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

Not applicable.

Was a considerable amount of time spent diverging from workshop related activities?

No. Very little time was spent diverging from Workshop related activities.

Any other points that need to be noted?

On the whole I got the impression that the first part of the workshop went extremely well and that all participants look forward to attending the second day. Participants were evidently keen to use their previous programming knowledge and to adapt it for use with the simulator and Java. Participant's made good very good progress during the workshop.

Trainee Teacher Workshop Two (TTW2) – Day Two

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

All participants arrived on time, and were present, for the second day of the workshop.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

As with the first day, all participants were enthusiastic and motivated during the session. There were no notable distractions and participants were keen to learn. All participants required some coding assistance on the second day, and more than during the first day. Upon having concepts explained, and examples discussed, participants almost always were able to grasp the construct and then implement it in their code.

Did participants seem enthusiastic throughout the session? If not, why?

The overwhelming impression I got from participants was that the session was both helpful and enjoyable. Several participants asked for the Workshop slides and Robot Simulator software to be sent to them after the Workshop ended as they felt they could make use of these.

What aspects of the workshop/programming concepts did participants struggle to grasp?

Few concepts caused participants consistent difficulty. Once explained, participants were often able to devise a solution to any problems encountered themselves. P2 seemed to struggle more than the other participants although with peer, and workshop leader, help this did not become a major problem.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

No concerns were voiced about the nature of the assessments used during the workshop. Participants were enthusiastic and wanted to perfect code, to the point where they needed to be instructed to move on and complete the other tasks. Completion of the post-workshop assessment task took around 35 minutes in total.

Was a considerable amount of time spent diverging from workshop related activities?

No. Participants spent virtually no time diverging from Workshop related activities.

Any other points that need to be noted?

On the whole I got the impression that the workshop was a great success. Several of the participants (P1, P2, P5) recognised that once they qualified as teachers they would have to teach programming themselves given recent educational reforms. Possibly due to this, participants seemed very eager to ask questions and to learn about the approach taken during the workshop. Copies of the workshop slides (and simulator) were requested by one of the trainees (P3) and the whole group agreed that this would be a good, and helpful, idea. The session ended with around 45 minutes to spare as good progress was made during the workshop.

1. Which of the following code is a valid way to call a method called 'move' in JAVA? (Circle one)

`move;` `move();` `move`

```
move;      move();      move
```

2. Which of the following is the correct way to declare the name of a Variable? (*Circle one*)

VariableName	variableName	variable NAME
--------------	--------------	---------------

3. Parameters (values) are passed to methods how? (*Circle one*)

By putting the value to be passed in the method's brackets e.g. `move(1.0)`;

By using the equals sign e.g. move = 1.0;

By typing the value to be passed next to the method name e.g. `move 1.0;`

4. Does the following statement relate to Variables or Constants: "These are used when you want to assign a value to an object once and make it so this value can't change"? (*Circle one*)

Variables	Constants
------------------	------------------

5. Which of the following would be the correct way to declare an integer with a value of 10? (*Circle one*)

```
Final int TIME = 10;      integer time = 10;      int time = 10;
```

6. Assign the correct data types to the following values (*Circle correct data type for each value*)

Value		Data Type	
2,546	bookcan	int	double
True	bookcan	int	double
9872	bookcan	int	double
0.1	bookcan	int	double
15	bookcan	int	double

7. Does the following code mean the same: `value = value - 1` and `value--`? (*Circle one*)

Yes	Yes, some of the time	No
1	2	3

PLEASE CONTINUE OVERLEAF...



8. Examine the following code: `if (length > 10 && length < 20)`. Does this code mean... (*Circle one*)

If length is greater than 10 or length is less than 20

If length is greater than 10 and length is less than 20

If length is not greater than 10 and length is not greater than 20

9. Which of the following is true? If...Else If statements... (*Circle one*)

Allow a program to select which block of code to execute if a condition is true

Are executed in sequence (e.g. one after another) regardless of the result of each condition

Must always end with an Else statement

10. System.out.println allows text output to be displayed in the terminal window. What would be printed in the case of the following If...Else If statement?

```
int value = 15;
```

```
if (value < 10)
```

```
System.out.println("The First If");
```

```
else if (value > 10)
```

```
System.out.println("The Second If");
```

"The First If"

(Circle one)

Nothing Printed

What is your participant code number? (Please specify)

□



In-Workshop Programming Exercise Two (TTW1-IWE2)



In Workshop Programming Exercise Two

1. "Java code will not compile unless it is indented" *(Circle one)*

True False

2. Apart from int, what are the three data types which are used to store whole numbers? *(Circle three)*

Float Byte Short Double Long String Char

3. Strings are used to store textual information. Which of the following is the correct way to declare a string variable called 'data'? *(Circle one)*

String data = "Hello"; String data = Hello; String data = 'Hello';

4. The operator != means what? *(Circle one)*

Less than or equal to Break from loop Not equal to

5. Do while loops always execute the body of a loop at least once. Why? *(Circle one)*

The condition is tested at the end of the loop The condition is tested at the beginning of the loop The condition is tested in the loop itself

6. Methods are particularly useful when... *(Circle one)*

You need to use the same piece of code repeatedly but do not want to keep repeating code You want to store data in computer memory for later use To evaluate whether one or more conditions is true or false

PLEASE CONTINUE OVERLEAF...



7. The following loop is an example of an endless loop. Why will this loop never end? *(Circle one)*

```
boolean condition = true;

while (condition())
{
    someMethod();
}
```

Because System.exit(0) needs to be used to break out of a loop

Because the loop condition will never evaluate to false

A Do While loop should have been used instead

8. What would be printed in the case of the following If...Else If statement? *(Circle one)*

```
double quantity = 9.99;

if (quantity < 0)
{
    System.out.println("The First If");
}
else if (quantity > 10)
{
    System.out.println("The Second If");
}
else
{
    System.out.println("The Second If");
}
```

"The First If"

"The Second If"

Nothing Printed

9. Below is some skeleton code. In the spaces provided write code to reflect the following:

While count is less than 10 increment count by one *(Fill in the blanks)*

```
int count = 10;

while ( )
{
    .....
}
```

What is your participant code number? *(Please specify)*



Post-Workshop Programming Exercise (TTW1-PWE)

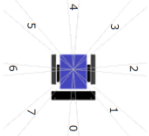


Post-Workshop Programming Exercise

The Workshop is almost over. All that remains is for you to complete a handful of programming challenges which bring together several of the programming concepts that you have used during the sessions. In total there are four challenges to complete. Details of these are described on this sheet. Unlike the previous tasks you have completed, your code in response to these exercises will be collected. This will help to improve both the Simulator and the Workshop in the future. As with all of the other data collected during the workshop, your code will be treated anonymously and will not be later identified.

Introduction

You will be using a new robot, Page, during this part of the workshop. Page is slightly different from the other robots you have used up until now. This is because Page has eight proximity sensors which are capable of detecting 3D objects all around the robot (see the image of Page to the right). Like the other robots you have used Page also has three encoder sensors that are capable of detecting 2D objects that have been drawn on the arena floor.



Task One

Select the Page class. You will notice that several Variables have already been declared for you. Study this code and the names of the Variables as these relate to the position and kind of sensor that they are. You will be using these Variables during the challenges that follow. Now may also be a good time to load the Simulator and to clear the arena of any existing objects by pressing the 'Clear Arena' button.



Your first challenge is to instruct Page to exhibit seeking behaviour. Essentially you want to instruct the robot so that it is able to follow a moving 3D object that you introduce into the arena. There are many ways to complete this task. If you are struggling the following pseudo-code may help:

If the front left sensor is less than the front right sensor move forward and turn left, else if the front left sensor is greater than the front right sensor move forward and turn right, else do not move forward and do not rotate

Do not worry if your code is not quite working correctly. The whole essence of the workshop has been about becoming familiar with programming concepts and ideas... not to be a master robot programmer!

In order to collect your code from all of the tasks the Notepad application will be used. Load Notepad by Clicking *Start > All Programs > Accessories > Notepad*. Once you have loaded Notepad click *File* and then *Save*. Name the file after your participant code number and save it somewhere that is easy to access (it is recommended that you save it to the Desktop). Once this file has saved, copy your code from Task One into Notepad using the Copy and Paste commands. Click Save again. Please ask for assistance if you are struggling with this.



CONTINUED OVERLEAF...



Task Two

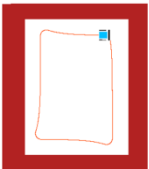
Once you have copied your code for Task One into Notepad, and have ensured that it has saved, clear your previously entered code. Do not delete the Variables that have been declared for you, however, as you will continue to need these for all of the challenges.

For Task Two you are going to instruct Page to follow walls. As always with programming there are numerous ways to solve this task and feel free to experiment. The following may help if you run into problems:

First move forward

If the front left sensor detects a wall is near then turn away else turn the opposite way

When you are satisfied with your code copy your program into the same Notepad file you have already created. Make sure you do not delete your previous code from Task One. This single Notepad file will contain all of your code in response to these challenges. Separate code within the file by pressing enter a few times between tasks or by numbering each task 1 to 4.



Task Three

Next you will program line following behaviour. You will use Page's encoder sensors (and not proximity sensors as you have for Tasks One and Two). The first part of this code has been completed for you:

```
// Insert Your Code Below

if(leftEncoder == 1 && rightEncoder == 0)
{
    move(0.1);
    turn(60);
}
```



Think about how this code works. This code instructs Page to move forward and turn left if the left encoder sensor detects a black line and the right encoder does not detect a black line. Expand this code so that Page is able to turn both left and right when following a black line. If neither of Page's encoder sensors detect a black line then Page should move straight forward. Copy your code and Save your Notepad file as before.

Task Four

For this task you will use a For Loop to instruct Page to perform a square as in the image to the right. Remember how a For Loop has three parts: initialisation (i.e. creating a variable and assigning a value), test (testing whether this variable evaluates to false) and increment/decrement. Within the For Loop instruct the robot to move forward for a set amount of time, to turn 90 degrees for 1 second before stopping turning. Think about how many times you will need to perform these actions in order to perform a square as in the picture as this will form the test part of your For Loop. After your For Loop instruct the program to exit by using the System.exit(0) command. Copy your code and Save your Notepad file.



Please ensure that your code for all four tasks is contained within the same Notepad file and that this has been saved. You will now be given instructions on how this code will be collected. Thank you.

Appendix A8 – Programming Performance During TTW1

In-Workshop Programming Exercise One (TTW1-IWE1)

Table A displays the performance of each participant on TTW1-IWE1. In this table a ‘1’ indicates a participant answered a particular question correctly while a ‘0’ indicates an incorrect answer was received. All 17 participants who took part in Day One of the workshop completed this programming exercise. The highest score awarded was the maximum ten (two participants), the lowest was six (four participants), the mean score is 7.9/10 and the standard deviation is 1.4.

<i>Participant Number</i>	<i>Question 1</i>	<i>Question 2</i>	<i>Question 3</i>	<i>Question 4</i>	<i>Question 5</i>	<i>Question 6</i>	<i>Question 7</i>	<i>Question 8</i>	<i>Question 9</i>	<i>Question 10</i>	<i>Total Score</i>
TTW1-P1	1	1	0	1	1	0	1	1	0	1	7
TTW1-P2	1	1	1	1	1	0	0	1	1	1	8
TTW1-P3	1	1	1	1	1	1	0	1	1	1	9
TTW1-P4	1	1	1	1	1	1	0	1	0	1	8
TTW1-P5	1	1	1	1	1	1	1	1	1	1	10
TTW1-P6	1	1	1	1	1	1	1	1	1	1	10
TTW1-P7	1	1	0	1	1	1	1	1	1	1	9
TTW1-P8	1	0	1	1	1	0	0	1	1	1	7
TTW1-P9	1	0	1	1	1	0	0	1	1	1	7
TTW1-P10	1	1	1	1	1	0	0	1	0	0	6
TTW1-P11	1	1	1	1	1	1	0	1	1	1	9
TTW1-P12	1	1	1	1	1	0	0	1	0	0	6
TTW1-P13	1	1	1	1	1	0	1	1	1	1	9
TTW1-P14	1	1	1	1	1	1	0	1	1	0	8
TTW1-P15	1	0	1	1	1	0	0	1	1	0	6
TTW1-P17	1	1	1	1	1	1	0	1	1	1	9
TTW1-P18	1	1	0	1	1	0	0	1	1	0	6
Totals	17	14	14	17	17	8	5	17	13	12	/

Table A. Breakdown of TTW1 participants performance on TTW1-IWE1.

The most recent programming experience of eight participants who completed TTW1-IWE1 was in Java. Table B displays the performance of these participants, on TTW1-IWE1, alone. The mean score of these participants was 8/10 (with a standard deviation of 1.6). In Table C the performance of the remaining nine participants, who had not recently learned Java, is provided. The mean score of these participants was 7.8/10 (with a standard deviation of 1.3).

<i>Participant Number</i>	<i>Question 1</i>	<i>Question 2</i>	<i>Question 3</i>	<i>Question 4</i>	<i>Question 5</i>	<i>Question 6</i>	<i>Question 7</i>	<i>Question 8</i>	<i>Question 9</i>	<i>Question 10</i>	<i>Total Score</i>
TTW1-P1	1	1	0	1	1	0	1	1	0	1	7
TTW1-P4	1	1	1	1	1	1	0	1	0	1	8
TTW1-P5	1	1	1	1	1	1	1	1	1	1	10
TTW1-P6	1	1	1	1	1	1	1	1	1	1	10
TTW1-P7	1	1	0	1	1	1	1	1	1	1	9
TTW1-P12	1	1	1	1	1	0	0	1	0	0	6
TTW1-P14	1	1	1	1	1	1	0	1	1	0	8
TTW1-P15	1	0	1	1	1	0	0	1	1	0	6
Totals	8	7	6	8	8	5	4	8	5	5	/

Table B. Breakdown of TTW1 participants' performance on TTW1-IWE1 (participants who most recently learned Java only).

<i>Participant Number</i>	<i>Question 1</i>	<i>Question 2</i>	<i>Question 3</i>	<i>Question 4</i>	<i>Question 5</i>	<i>Question 6</i>	<i>Question 7</i>	<i>Question 8</i>	<i>Question 9</i>	<i>Question 10</i>	<i>Total Score</i>
TTW1-P2	1	1	1	1	1	0	0	1	1	1	8
TTW1-P3	1	1	1	1	1	1	0	1	1	1	9
TTW1-P8	1	0	1	1	1	0	0	1	1	1	7
TTW1-P9	1	0	1	1	1	0	0	1	1	1	7
TTW1-P10	1	1	1	1	1	0	0	1	0	0	6
TTW1-P11	1	1	1	1	1	1	0	1	1	1	9
TTW1-P13	1	1	1	1	1	0	1	1	1	1	9
TTW1-P17	1	1	1	1	1	1	0	1	1	1	9
TTW1-P18	1	1	0	1	1	0	0	1	1	0	6
Totals	9	7	8	9	9	3	1	9	8	7	/

Table C. Breakdown of TTW1 participants' performance on TTW1-IWE1 (participants who had not most recently learned Java only).

In Figure I the combined scores of all TTW1 trainees on TTW1-IWE1, in response to each question, is displayed. In Figure II a comparison of the performance between the trainees with Java programming experience, to those without, can be seen.

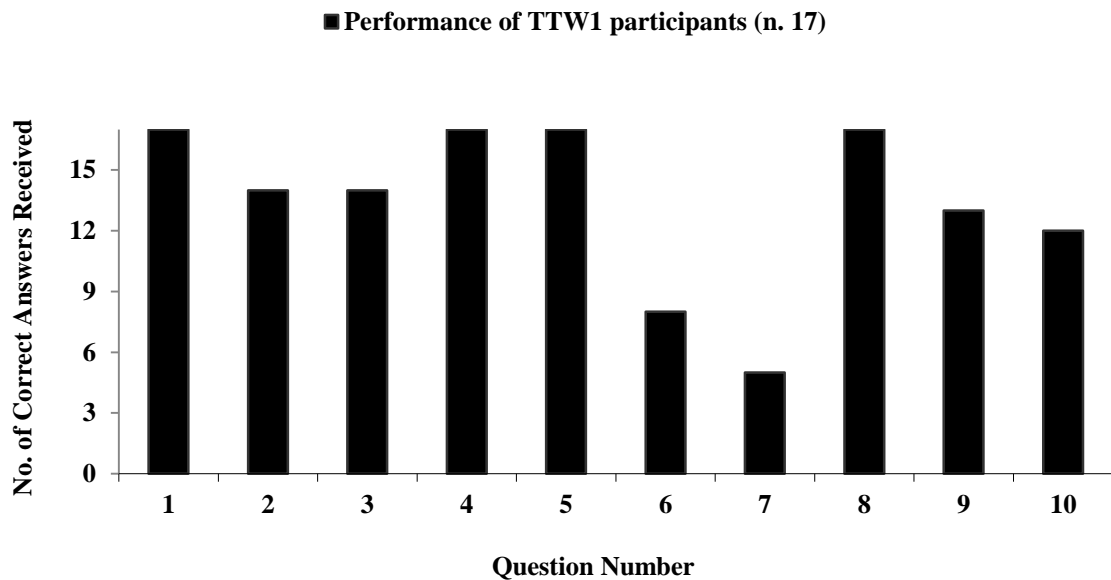


Figure I. Combined scores of all TTW1 trainees on TTW1-IWE1.

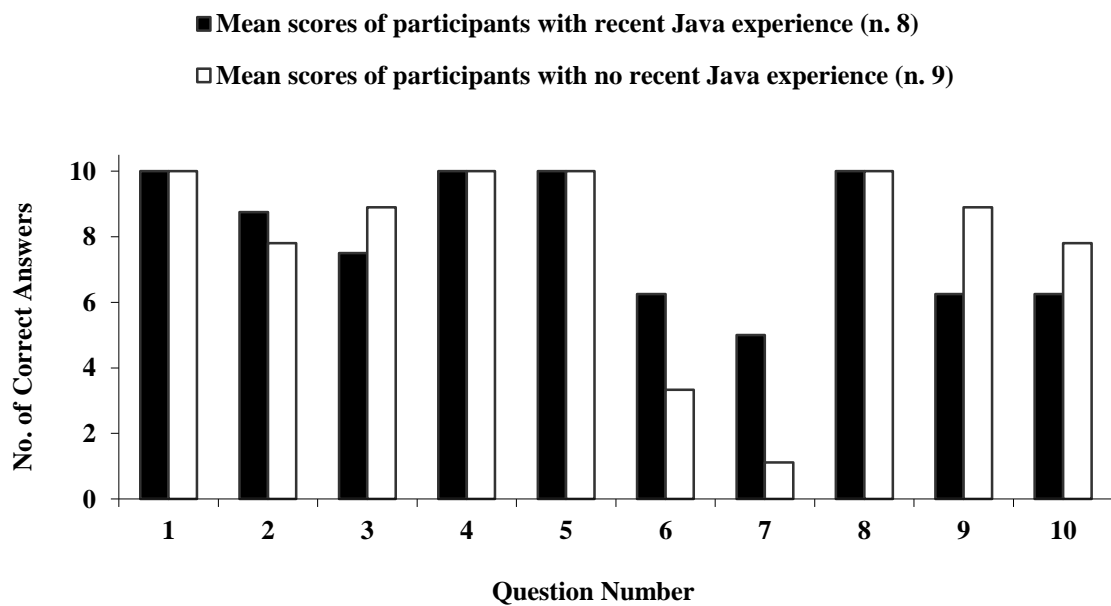


Figure II. Comparison of the performance between TTW1 participants with recent Java programming experience, to those without, on TTW1-IWE1.

In-Workshop Programming Exercise Two (TTW1-IWE2)

Table D displays the performance of each participant on TTW1-IWE2. All 16 participants who took part in Day Two of the workshop completed this programming exercise (note that participant, TTW1-P14 was absent). Also, one participant (TTW1-P3) was not able to take a full part in the programming exercises set after sustaining a wrist injury in the interval between Day One and Day Two of the workshop. This could have had an impact upon this participants' performance. The highest score awarded was the maximum ten (three participants), the lowest was two (two participants), the mean score is 6.6/10 and the standard deviation is 2.83.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
TTW1-P1	0	0	1	0	0	0	0	1	0	0	2
TTW1-P2	0	1	1	1	1	1	1	1	1	1	9
TTW1-P3	0	1	1	0	1	1	0	0	0	0	4
TTW1-P4	1	1	1	1	0	1	0	0	1	1	7
TTW1-P5	1	1	1	1	1	1	1	1	1	1	10
TTW1-P6	1	1	1	1	1	1	1	1	1	1	10
TTW1-P7	1	1	1	1	0	1	1	1	1	1	9
TTW1-P8	1	0	0	1	1	1	1	0	0	0	5
TTW1-P9	1	0	1	1	0	1	1	0	1	1	7
TTW1-P10	0	0	0	1	1	1	1	1	1	1	7
TTW1-P11	1	1	1	1	1	1	1	1	1	1	10
TTW1-P12	1	0	1	1	1	0	0	1	0	0	5
TTW1-P13	1	0	1	1	1	1	1	1	1	1	9
TTW1-P15	1	1	0	0	0	1	0	0	0	0	3
TTW1-P17	0	1	1	0	1	1	1	0	1	1	7
TTW1-P18	1	0	0	0	0	1	0	0	0	1	3
Totals	11	9	12	11	10	14	10	9	10	11	/

Table D. Breakdown of TTW1 participants' performance on TTW1-IWE2.

The most recent programming experience of seven participants who completed TTW1-IWE2 was in Java. Table E displays the performance of these participants alone. The mean score of these participants was 6.8/10 (with a standard deviation of 3.3). In Table F the performance of the remaining nine participants, who had not recently learned Java, is provided. The mean score of these participants was 6.8/10 (with a standard deviation of 2.3).

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
TTW1-P1	0	0	1	0	0	0	0	1	0	0	2
TTW1-P4	1	1	1	1	0	1	0	0	1	1	7
TTW1-P5	1	1	1	1	1	1	1	1	1	1	10
TTW1-P6	1	1	1	1	1	1	1	1	1	1	10
TTW1-P7	1	1	1	1	0	1	1	1	1	1	9
TTW1-P12	1	0	1	1	1	0	0	1	0	0	5
TTW1-P15	1	1	0	0	0	1	0	0	0	0	3
Totals	6	5	6	5	3	5	3	5	4	4	/

Table E. Breakdown of TTW1 participants' performance on TTW1-IWE2 (participants who most recently learned Java only).

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	<u>Total Score</u>
TTW1-P2	0	1	1	1	1	1	1	1	1	1	9
TTW1-P3	0	1	1	0	1	1	0	0	0	0	4
TTW1-P8	1	0	0	1	1	1	1	0	0	0	5
TTW1-P9	1	0	1	1	0	1	1	0	1	1	7
TTW1-P10	0	0	0	1	1	1	1	1	1	1	7
TTW1-P11	1	1	1	1	1	1	1	1	1	1	10
TTW1-P13	1	0	1	1	1	1	1	1	1	1	9
TTW1-P17	0	1	1	0	1	1	1	0	1	1	7
TTW1-P18	1	0	0	0	0	1	0	0	0	1	3
Totals	5	4	6	6	7	9	7	4	6	7	/

Table F. Breakdown of TTW1 participants' performance on TTW1-IWE2 (participants who had not most recently learned Java).

In Figure III the combined scores of all TTW1 trainees on TTW1-IWE2 is displayed. In Figure IV a comparison of the performance between the trainees with Java programming experience, to those without, can be seen.

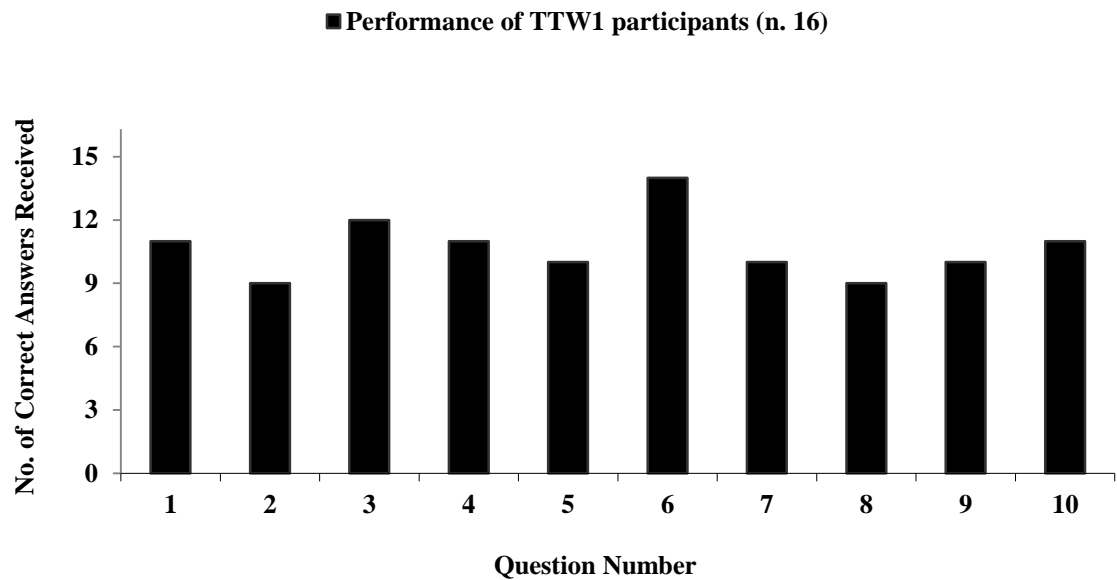


Figure III. Combined scores of all TTW1 participants on TTW1-IWE2.

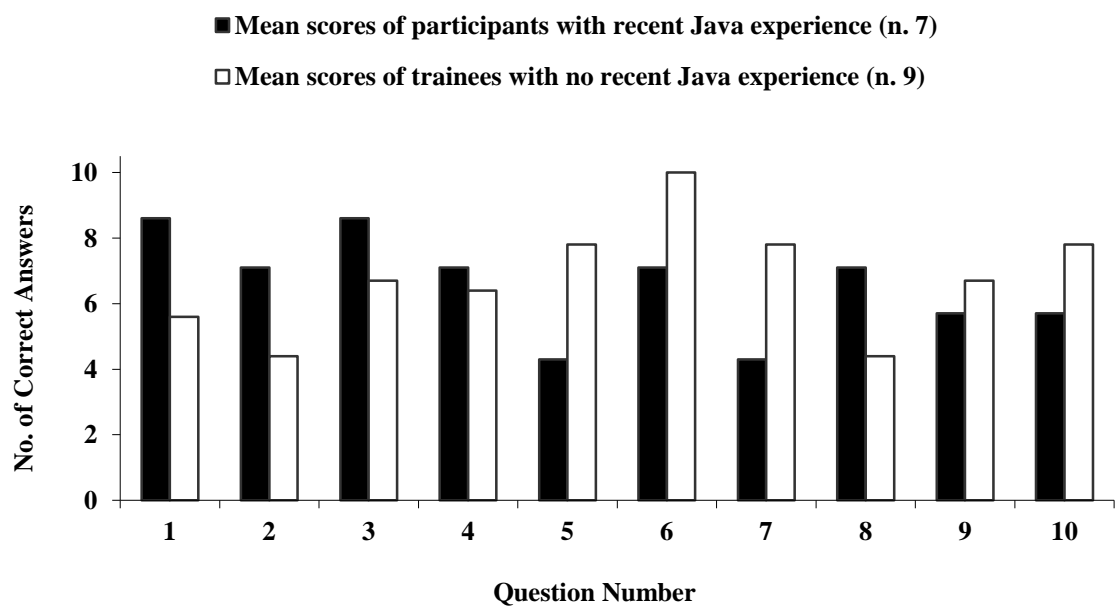


Figure IV. Comparison of the performance between TTW1 participants with recent Java programming experience, to those without, on TTW1-IWE2.

Post-Workshop Programming Exercise (TTW1-PWE)

TTW1-PWE consisted of four separate programming exercises and all participants had submitted their code after around 30 minutes. Each participant was awarded a grade of A, B or C for their performance on each task. Table G displays a breakdown of the scores awarded. Note, TTW1-P3 attended the second day of the workshop but did not completed TTW1-PWE due to a wrist injury.

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
TTW1-P1	A	C	C	C
TTW1-P2	A	C	A	A
TTW1-P4	A	C	B	C
TTW1-P5	A	C	B	C
TTW1-P6	A	C	A	A
TTW1-P7	A	C	B	C
TTW1-P8	B	C	C	C
TTW1-P9	C	C	C	C
TTW1-P10	C	C	B	C
TTW1-P11	A	C	A	A
TTW1-P12	C	C	C	C
TTW1-P13	B	C	A	A
TTW1-P15	C	C	C	C
TTW1-P17	A	C	B	C
TTW1-P18	C	C	C	C
Total (n. 8)	A – 8	A – 0	A – 4	A – 4
	B – 2	B – 0	B – 5	B – 0
	C – 5	C – 15	C – 6	C – 11

Table G. Breakdown of participants' performance on TTW1-PWE (by number of participants awarded Grade A, B or C).

The performance of the TTW1 participants with recent Java programming experience, to those without, on TTW1-PWE has also been presented separately to allow comparison. This data is displayed in Table H and Table I.

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
TTW1-P1	A	C	C	C
TTW1-P4	A	C	B	C
TTW1-P5	A	C	B	C
TTW1-P6	A	C	A	A
TTW1-P7	A	C	B	C
TTW1-P12	C	C	C	C
TTW1-P15	C	C	C	C
Total (n. 7)	A – 4 B – 0 C – 2	A – 0 B – 0 C – 7	A – 1 B – 3 C – 3	A – 1 B – 0 C – 6

Table H. Breakdown of TTW1 participants' performance on TTW1-PWE (participants who most recently learned Java only).

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
TTW1-P2	A	C	A	A
TTW1-P8	B	C	C	C
TTW1-P9	C	C	C	C
TTW1-P10	C	C	B	C
TTW1-P11	A	C	A	A
TTW1-P13	B	C	A	A
TTW1-P17	A	C	B	C
TTW1-P18	C	C	C	C
Total (n. 8)	A – 3 B – 2 C – 3	A – 0 B – 0 C – 8	A – 3 B – 2 C – 3	A – 3 B – 0 C – 5

Table I. Breakdown of TTW1 participants' performance on TTW1-PWE (participants who had not most recently learned Java).

Appendix A9 – Programming Exercises

In-Workshop Programming Exercise One (IWE1)



In Workshop Programming Exercise One

1. Imagine within the simulator a method named drawTriangle exists. This method instructs the robot to perform a triangle. You pass an integer to drawTriangle as a parameter. This tells the robot how many seconds each side of the triangle should be drawn for.
In one line of code call the drawTriangle method and pass a parameter of 5 seconds.

2. Which of the following is the conventional way to name a Variable? (Circle one)

TimeTaken timeTaken time TAKEN timetaken

3. You have been asked to instruct your robot to rotate 180 degrees so it faces in the opposite direction. For this task you always want to rotate by this amount and make it so this value can not change when your program runs. Moreover, you only want to assign the value of 180 once.
Which of the following would be best to use? (Circle one)

Variable Constant Boolean String

4. In one line of code write how you would declare and initialise an integer called power with a value of 10:

5. Assign the most appropriate data type for each of the values below:
(Circle the most appropriate data type for each value)

Values	Correct Data Type			
2,546	boolean	int	double	
True	boolean	int	double	
9872	boolean	int	double	
0.10	boolean	int	double	
15	boolean	int	double	

6. Convert the following statement into code in the space below:
"If length is greater than 10 and length is less than 20":

if (_____)

PLEASE CONTINUE OVERLEAF...



7. System.out.println allows text output to be displayed in the terminal window. What would be printed in the case of the following if...else statement?

int value = 5;

```
if (value < 10)
{
    value = value + 10;
    System.out.println ("The First If");
}
else
{
    System.out.println ("The Second If");
}
```

The First If The Second If The First If Nothing Printed
and
The Second If

(Circle one)

8. Examine the following code:

```
int number = 10;
number--;
```

What is the value of number when this code has been executed once? (Circle one)

9 11 0 10

9. Which of the following statements is FALSE?

- a) if...else if statements allow a program to select which block of code to execute
- b) You can compare values in the condition of an if statement
- c) if statements must always end with an else
- d) an if statement is executed if its condition is true

(Circle one)

a b c d

10. Imagine a double named today'sTemperature has been declared and implemented for you. In one line of code how would you print today'sTemperature to the terminal window:

What is your participant code number?

(Fill in the box)



In Workshop Programming Exercise Two

- Apart from `int`, what are the three data types which are used to store whole numbers?
(Circle three)

Float	Byte	Short	Double	Long	String	Char
-------	------	-------	--------	------	--------	------
- Strings are used to store textual information. In one line of code write how you would declare a String called `message` which is initialised with the string "hello":
- Which of the following statements is TRUE?

a) The operator <code>!=</code> means less than	b) The operator <code>!=</code> means break from a loop	c) The operator <code>!=</code> means not equal to	d) The operator <code>!=</code> means equal to
---	---	--	--

a	b	c	d
---	---	---	---

(Circle one)
- Which of the following is used to execute the code in the loop at least once before the condition is evaluated? (Circle one)

a) <code>while</code>	b) <code>do-while</code>	c) <code>for</code>	d) None of these
-----------------------	--------------------------	---------------------	------------------

a	b	c	d
---	---	---	---

(Circle one)
- The following is an infinite loop. Write in the space below why you think this loop will never end:

```
int i = 0;
while(i < 5)
{
    System.out.println(i);
}
```



- Imagine you have programmed a robot to constantly check the temperature and to increase the speed of a fan depending on how hot it is. This code is quite complex and has taken up many lines. You want to be able to call this code over and over again. Write why it might be a good idea to put this code in a method?

- Which of the following would be a correct and working way to declare a method called `robotStatus`?

a	b	c	d
---	---	---	---

<pre>robotStatus() { ; }</pre>	<pre>public robotStatus() { ; }</pre>	<pre>public void robotStatus() { ; }</pre>	<pre>public void robotStatus; { ; }</pre>
------------------------------------	---	--	---

(Circle one)

- Imagine you have been asked to design a program which continually checks whether the user has entered `Yes` in response to the question "Do you want the robot to move?". This program should not proceed until the user enters `Yes` exactly. If the user inputs anything else the question should be repeated.
Which of the following would be best to use?
(Circle one)

`if...else` statement `while` loop `System.exit(0)` `if...else if` statement

- In the space below use an `if...else if` statement and your programming knowledge to:
inform the user that if `power` is greater than 25 then the power level is ok, else if `power` is between 5 and 25 inform the user that `power` is low, else inform the user that `power` is critical. You do not have to declare a variable for `power`. (Write Below)

What is your participant code number?

(Fill in the box)

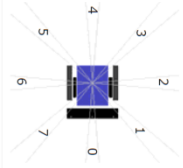


Post-Workshop Programming Exercise

The Workshop is almost over. All that remains is for you to complete a handful of programming challenges which bring together the concepts that you have used during the sessions. In total there are four challenges to complete. Details of these are described on this sheet. Unlike the previous tasks your code in response to these exercises will be collected. This will help to improve the Simulator and the Workshop in the future.

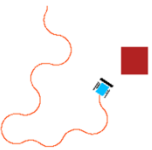
Introduction

You will be using a new robot, Page, during this part of the workshop. Page is slightly different from the other robots you have used so far. This is because Page has eight proximity sensors which are capable of detecting 3D objects all around the robot (see the image of Page to the right). Page also has three encoder sensors that are capable of detecting black 2D objects that have been drawn on the arena floor.



Task One

Select the Page class. You will notice that several Variables have already been declared for you. Study this code and the names of the Variables as these relate to the position and kind of sensor that they are. You will be using these Variables during the challenges that follow. Now may also be a good time to load the Simulator and to clear the arena of any existing objects by pressing the 'Clear Arena' button.



Your first challenge is to instruct Page to exhibit seeking behaviour. Essentially you want to instruct the robot so that it is able to follow a 3D object that you introduce and move around the arena. There are many ways to complete this task. If you are struggling the following pseudo-code may help.

If the front left sensor reads a value less than that read by the front right sensor manoeuvre forward to the left, else if the front left sensor reads a value greater than that read by the front right sensor manoeuvre forward to the right, else stop.

Do not worry if your code is not quite working correctly. The whole essence of the workshop has been about becoming familiar with programming concepts and ideas... not to be a master robot programmer!

In order to collect your code in response to all four tasks the Notepad application will be used. Load Notepad by Clicking *Start > All Programs > Accessories > Notepad*. Once you have loaded Notepad click *File* and then *Save*. Name the file after your participant code number and save it somewhere that is easy to access (it is recommended that you save it to the Desktop). Once this file has saved, copy your code from Task One into Notepad using the Copy and Paste commands. Click Save again. Please ask for assistance if you are struggling with this.



CONTINUED OVERLEAF...

Task Two

Once you have copied and saved your code for Task One into Notepad clear your previously entered code. Do not delete the Variables that have been declared, however, as you will continue to need these for all of the challenges. For Task Two you are going to instruct Page to follow lines. For this task you will use Page's encoder sensors (and not proximity sensors as for Task One). As always with programming there are many ways to solve this task and feel free to experiment. The following code provides you with a start:

```
// Insert Your Code Below

if(leftEncoder == 1 && rightEncoder == 0)
{
    move(0.1);
    turn(60);
}
```



This code instructs Page to move forward and turn left if the left encoder sensor detects black and the right encoder detects white. Expand this code so that Page is able to turn both left and right when following a black line. If neither of Page's encoder sensors detect black then Page should move forward.

When you are satisfied with your code copy your program into the Notepad file you have already created. Make sure you do not delete your previous code from Task One. This single Notepad file will contain all of your code. Separate code within the file by pressing enter a few times or by numbering each task 1 to 4.

Task Three

For this task you will use a For Loop to instruct Page to perform a square as in the image. Within the For Loop instruct the robot to:

Move straight forward for several seconds.

Turn at a right angle for 1 second before stopping to turn.



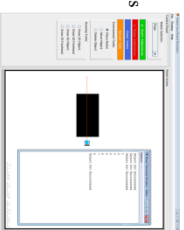
Think about how many times you will need to repeat this code in order to draw a square as this will form the test part of your For Loop. After your loop instruct the program to exit by using System.exit(0). Copy your code and Save your Notepad file.

Task Four

The code below instructs Page to move forward in a straight line. Copy this into the Page class:

```
// Insert Your Code Below.

move(0.1);
duration(1.0);
```



Below this code your objective is to count and report how many seconds Page's middleEncoder detects black for when moving forward. First you need to declare a **class variable** which stores the number of seconds. Then, if black is detected, you need to increment (by one) the seconds variable you have declared and report this value to the user. If the middleEncoder does not detect black inform the user that nothing has been detected.

You need to decide whether a loop (while, do-while or for) or conditional construct (if...else) is best suited for this task. After completing the above you may also wish to create and call a method named goForward() which contains the instructions which you previously copied into the Page class. Once you have attempted the task copy your code into the Notepad file you have been using.

Please ensure that your code for all four tasks is contained within the same Notepad file and that this has been saved. You will now be given instructions on how this code will be collected. Thank you.

Appendix A10 – Programming Performance During TTW2

In-Workshop Programming Exercise One (IWE1)

Table J displays the performance of participants on IWE1. In this table a ‘1’ indicates a correct answer while a ‘0’ indicates an incorrect answer. All five participants completed this exercise. The highest score awarded was the maximum 10 (one participant), the lowest was 7 (two participants), the mean score is 8.4/10 and the standard deviation is 1.34.

<i>Participant Number</i>	<i>Question 1</i>	<i>Question 2</i>	<i>Question 3</i>	<i>Question 4</i>	<i>Question 5</i>	<i>Question 6</i>	<i>Question 7</i>	<i>Question 8</i>	<i>Question 9</i>	<i>Question 10</i>	<i>Total Score</i>
TTW2-P1	1	1	1	1	0	0	0	1	1	1	7
TTW2-P2	1	1	1	1	1	1	1	1	1	1	10
TTW2-P3	1	1	1	1	1	0	1	1	1	1	9
TTW2-P4	1	1	1	1	1	1	0	1	1	1	9
TTW2-P5	1	1	1	1	1	0	0	1	0	1	7
Totals	5	5	5	5	4	2	2	5	4	5	/

Table J. Breakdown of TTW2 trainees' performance on IWE1.

In Figure V the combined scores of TTW1 participants, in response to each question, is displayed.

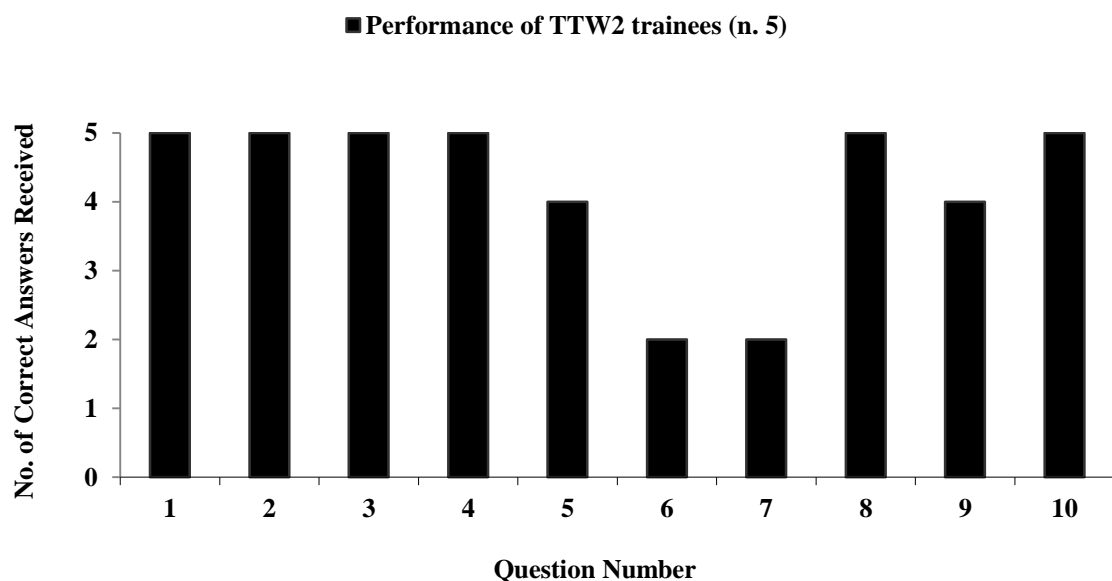


Figure V. Combined scores of all TTW2 trainees on IWE1.

In-Workshop Programming Exercise One (IWE1)

Table K displays participant performance on IWE2. The highest score awarded was the maximum 9 (three participants), the lowest was 4 (one participants), the mean score is 7.2/10 and the standard deviation is 2.49.

<i>Participant Number</i>	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8	Question 9	Question 10	Total Score
TTW2-P1	1	0	0	1	0	1	1	0	0	0	4
TTW2-P2	1	0	1	1	1	1	1	1	1	1	9
TTW2-P3	1	1	1	1	1	1	1	1	1	0	9
TTW2-P4	1	0	1	1	1	1	1	1	1	1	9
TTW2-P5	1	1	0	0	0	1	1	0	1	0	5
Totals	5	2	3	4	3	5	5	3	4	2	/

Table K. Breakdown of TTW2 participants' performance on IWE1.

In Figure VI the combined scores of TTW2 participants, in response to each question, is displayed.

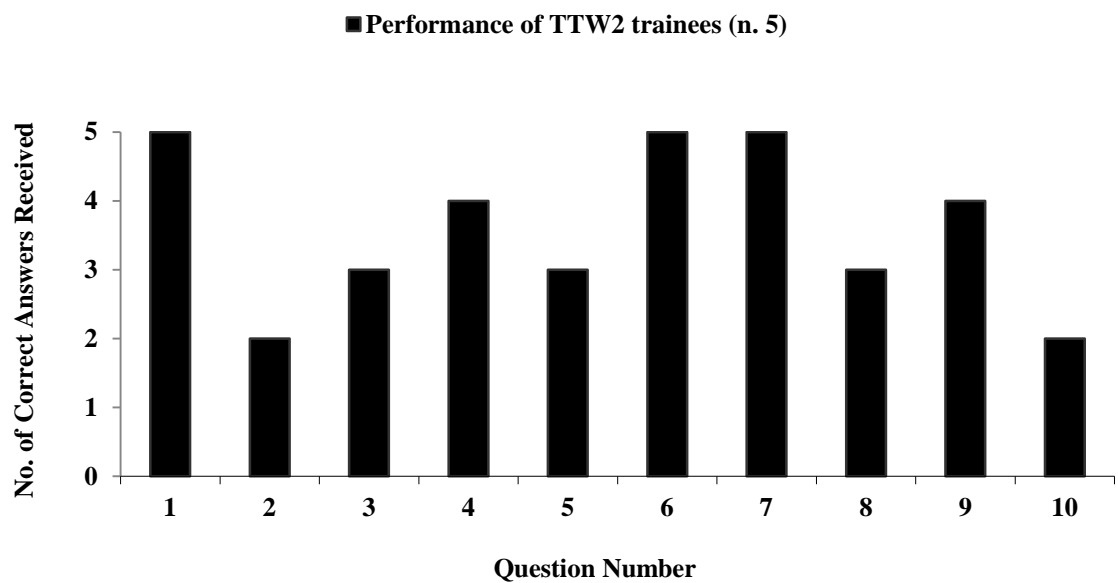


Figure VI. Combined scores of all TTW2 trainees on IWE2.

Post-Workshop Programming Exercises (PWE)

PWE consisted of four separate programming exercises. All participants had submitted their code after around 35 minutes. Each participants was awarded a grade of A, B or C for their performance on each task. Table L displays a breakdown of the scores awarded.

<i>Participant Number</i>	Task 1	Task 2	Task 3	Task 4
TTW2-P1	A	A	B	B
TTW2-P2	A	A	B	B
TTW2-P3	A	A	A	B
TTW2-P4	A	A	A	A
TTW2-P5	B	A	B	A
Total (n. 5)	A – 4	A – 5	A – 2	A – 2
	B – 1	B – 0	B – 3	B – 3
	C – 0	C – 0	C – 0	C – 0

Table L. Breakdown of TTW2 participants performance on PWE (by number of trainees awarded Grade A, B or C).

Appendix A11 – Student Questionnaires

Student Questionnaire One (SQ1)



Questionnaire - Pre-Workshop

Section One: Your Programming Background

1. Have you learnt any programming languages? If NO go to Section 2 overleaf. If Yes:

a. Please write which computer languages you have learnt and to what level (*Please specify and circle*)

[illegible]

b. Which language did you learn most recently? (*Please specify*) _____

c. Previously, how did you mainly learn to program? (Select one)

Self-taught Part of a course/education program

Other _____

d. Which of the following best describes your past programming experience? (*Circle one*)

Didn't Like	Indifferent	Enjoyed
1	2	3
4	5	6
7	8	9
10	11	12
13	14	15
16	17	18
19	20	21
22	23	24
25	26	27
28	29	30
31	32	33
34	35	36
37	38	39
40	41	42
43	44	45
46	47	48
49	50	51
52	53	54
55	56	57
58	59	60
61	62	63
64	65	66
67	68	69
70	71	72
73	74	75
76	77	78
79	80	81
82	83	84
85	86	87
88	89	90
91	92	93
94	95	96
97	98	99
100	101	102
103	104	105
106	107	108
109	110	111
112	113	114
115	116	117
118	119	120
121	122	123
124	125	126
127	128	129
130	131	132
133	134	135
136	137	138
139	140	141
142	143	144
145	146	147
148	149	150
151	152	153
154	155	156
157	158	159
160	161	162
163	164	165
166	167	168
169	170	171
172	173	174
175	176	177
178	179	180
181	182	183
184	185	186
187	188	189
190	191	192
193	194	195
196	197	198
199	200	201
202	203	204
205	206	207
208	209	210
211	212	213
214	215	216
217	218	219
220	221	222
223	224	225
226	227	228
229	230	231
232	233	234
235	236	237
238	239	240
241	242	243
244	245	246
247	248	249
250	251	252
253	254	255
256	257	258
259	260	261
262	263	264
265	266	267
268	269	270
271	272	273
274	275	276
277	278	279
280	281	282
283	284	285
286	287	288
289	290	291
292	293	294
295	296	297
298	299	300
301	302	303
304	305	306
307	308	309
310	311	312
313	314	315
316	317	318
319	320	321
322	323	324
325	326	327
328	329	330
331	332	333
334	335	336
337	338	339
340	341	342
343	344	345
346	347	348
349	350	351
352	353	354
355	356	357
358	359	360
361	362	363
364	365	366
367		

Enjoyed

e. Did you find programming challenging? (*Circle one*)

Challenging Indifferent Trivial

Trivial

PLEASE TURN OVER FOR SECTION TWO...



Section Two: Your Thoughts on Programming

2. Do you associate any issues with learning to program? If Yes please identify up-to three of these issues:

3. Do you associate any stereotypes with learning to program? If Yes please identify up-to three of these stereotypes:

4. Would you consider learning to program in your spare time if you were given appropriate support? (*Circle one*)

Yes	No	Not Sure
-----	----	----------

5. Do you believe that programming should be an important part of the National Curriculum in Schools? (*Circle one*)

Yes		No		Not Sure
-----	--	----	--	----------

7. What is your gender? (Circle one)

	Male	Female
1. <i>Staphylinidae</i>	10	10
2. <i>Curculionidae</i>	10	10
3. <i>Chrysomelidae</i>	10	10
4. <i>Scarabaeidae</i>	10	10
5. <i>Colletidae</i>	10	10
6. <i>Formicidae</i>	10	10
7. <i>Psyllidae</i>	10	10
8. <i>Thysanoptera</i>	10	10
9. <i>Homoptera</i>	10	10
10. <i>Lepidoptera</i>	10	10
11. <i>Diptera</i>	10	10
12. <i>Hymenoptera</i>	10	10
13. <i>Orthoptera</i>	10	10
14. <i>Blattellidae</i>	10	10
15. <i>Dermaptera</i>	10	10
16. <i>Isopoda</i>	10	10
17. <i>Coleoptera</i>	10	10
18. <i>Phthiraptera</i>	10	10
19. <i>Siphonura</i>	10	10
20. <i>Polychaeta</i>	10	10
21. <i>Amphipoda</i>	10	10
22. <i>Crustacea</i>	10	10
23. <i>Mollusca</i>	10	10
24. <i>Arachnida</i>	10	10
25. <i>Insecta</i>	10	10
26. <i>Chelicerata</i>	10	10
27. <i>Phlebotomina</i>	10	10
28. <i>Phlebotomina</i>	10	10
29. <i>Phlebotomina</i>	10	10
30. <i>Phlebotomina</i>	10	10
31. <i>Phlebotomina</i>	10	10
32. <i>Phlebotomina</i>	10	10
33. <i>Phlebotomina</i>	10	10
34. <i>Phlebotomina</i>	10	10
35. <i>Phlebotomina</i>	10	10
36. <i>Phlebotomina</i>	10	10
37. <i>Phlebotomina</i>	10	10
38. <i>Phlebotomina</i>	10	10
39. <i>Phlebotomina</i>	10	10
40. <i>Phlebotomina</i>	10	10
41. <i>Phlebotomina</i>	10	10
42. <i>Phlebotomina</i>	10	10
43. <i>Phlebotomina</i>	10	10
44. <i>Phlebotomina</i>	10	10
45. <i>Phlebotomina</i>	10	10
46. <i>Phlebotomina</i>	10	10
47. <i>Phlebotomina</i>	10	10
48. <i>Phlebotomina</i>	10	10
49. <i>Phlebotomina</i>	10	10
50. <i>Phlebotomina</i>	10	10
51. <i>Phlebotomina</i>	10	10
52. <i>Phlebotomina</i>	10	10
53. <i>Phlebotomina</i>	10	10
54. <i>Phlebotomina</i>	10	10
55. <i>Phlebotomina</i>	10	10
56. <i>Phlebotomina</i>	10	10
57. <i>Phlebotomina</i>	10	10
58. <i>Phlebotomina</i>	10	10
59. <i>Phlebotomina</i>	10	10
60. <i>Phlebotomina</i>	10	10
61. <i>Phlebotomina</i>	10	10
62. <i>Phlebotomina</i>	10	10
63. <i>Phlebotomina</i>	10	10
64. <i>Phlebotomina</i>	10	10
65. <i>Phlebotomina</i>	10	10
66. <i>Phlebotomina</i>	10	10
67. <i>Phlebotomina</i>	10	10
68. <i>Phlebotomina</i>	10	10
69. <i>Phlebotomina</i>	10	10
70. <i>Phlebotomina</i>	10	10
71. <i>Phlebotomina</i>	10	10
72. <i>Phlebotomina</i>	10	10
73. <i>Phlebotomina</i>	10	10
74. <i>Phlebotomina</i>	10	10
75. <i>Phlebotomina</i>	10	10
76. <i>Phlebotomina</i>	10	10
77. <i>Phlebotomina</i>	10	10
78. <i>Phlebotomina</i>	10	10
79. <i>Phlebotomina</i>	10	10
80. <i>Phlebotomina</i>	10	10
81. <i>Phlebotomina</i>	10	10
82. <i>Phlebotomina</i>	10	10
83. <i>Phlebotomina</i>	10	10
84. <i>Phlebotomina</i>	10	10
85. <i>Phlebotomina</i>	10	10
86. <i>Phlebotomina</i>	10	10
87. <i>Phlebotomina</i>	10	10
88. <i>Phlebotomina</i>	10	10
89. <i>Phlebotomina</i>	10	10
90. <i>Phlebotomina</i>	10	10
91. <i>Phlebotomina</i>	10	10
92. <i>Phlebotomina</i>	10	10
93. <i>Phlebotomina</i>	10	10
94. <i>Phlebotomina</i>	10	10
95. <i>Phlebotomina</i>	10	10
96. <i>Phlebotomina</i>	10	10
97. <i>Phlebotomina</i>	10	10
98. <i>Phlebotomina</i>	10	

Female

8. What is your participant code number? (*Please specify*)

END OF QUESTIONNAIRE





Questionnaire - Post-Workshop

Section One: Your Workshop Experience

1. In regards to your programming experience during the workshop, which of the following best describes your enjoyment? *(Circle one)*
- | | | |
|---------------|-------------|-----------|
| Not Enjoyable | Indifferent | Enjoyable |
|---------------|-------------|-----------|
2. In regards to the workshop session, which of the following best describes how difficult the programming tasks have been? *(Circle one)*
- | | | |
|-----------|---------------------------|------|
| Difficult | Neither difficult or easy | Easy |
|-----------|---------------------------|------|
3. Do you believe that the Robot Simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
4. Please specify up-to three things you *liked* about the Simulator *(Please specify)*
- _____
- _____
- _____
5. Please specify up-to three things you *did not like* about the Simulator *(Please specify)*
- _____
- _____
- _____
6. In regards to the effectiveness of teaching introductory programming, rate the different elements of the workshop on a scale of 1 (Not at all effective) to 5 (Extremely effective) *(Please circle your choices)*
- | | | | | | |
|---------------------|---|---|---|---|---|
| Robot Simulator | 1 | 2 | 3 | 4 | 5 |
| Programming Support | 1 | 2 | 3 | 4 | 5 |
| Presentation | 1 | 2 | 3 | 4 | 5 |
| Environment | 1 | 2 | 3 | 4 | 5 |

PLEASE TURN OVER FOR SECTION TWO...



Section Two: Your Thoughts on Programming

7. If you have previously learnt a programming language, was your *previous* introduction to basic programming... *(Circle one)*
- | |
|------------------------------|
| Much less effective |
| Less effective |
| About the same effectiveness |
| More effective |
| Much more effective |
- Little previous programming experience
8. Has the Robot Simulator helped to improve your perception of programming? *(Select one)*
- | | | | |
|-----|----|----------|-------------|
| Yes | No | Not Sure | Other _____ |
|-----|----|----------|-------------|
9. Has the Robot Simulator helped to dispel any negative stereotypes you had about programming before the session? *(Select one)*
- | | | | |
|-----|----|----------|-------------|
| Yes | No | Not Sure | Other _____ |
|-----|----|----------|-------------|
10. Would you consider learning to program in your spare time if you were given appropriate support? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
11. Do you believe that programming should be an important part of the National Curriculum in Schools? *(Circle one)*
- | | | |
|-----|----|----------|
| Yes | No | Not Sure |
|-----|----|----------|
12. Do you have any general comments about the Robot Simulator or the session in general? *(Please specify)*
- _____
- _____
- _____
- _____
- _____
13. What is your participant code number? *(Please specify)*
-



Appendix A12 – Student Workshop Log Transcripts

Student Workshop One (SW1) – Day One

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

P8 left after 2 hours due to having other commitments. Therefore P8 did not submit In Workshop Programming Test One. P14 arrived around 1 hour 30 minutes late for the workshop. I spent around 5 to 10 minutes attempting to bring P14 “up to speed” with several critical points. The participants who sat next to P14 also appeared to offer support to P14 for around 30 minutes after the participant had arrived.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants appeared to be “on task” during the workshop. There was little disruption or distractions during the session. P4, P9, P14 were speaking to each other at times about unrelated matters but this did not become a serious problem.

Did participants seem enthusiastic throughout the session? If not, why?

Whilst most participants seemed to remain interested throughout the workshop, P1, P2 and P5 in particular seemed very engaged with the workshop content. At several points these participants advanced beyond the rest of the group and begun to experiment with code in order to elicit additional behaviour out of the robots. P3 remained willing to work throughout the workshop but occasionally struggled to grasp specific syntax. At times P4, P9 and P14 were occasionally observed to be talking to one another about topics unrelated to the workshop. These participants seemed unwilling to experiment with code and only ran their programs once they had attempted to code a “complete” solution. P6 remained on task and enthusiastic during the session. P7 and P10 mentioned at several points how they felt a little lost by what they were learning and appeared to be a little bemused at times.

What aspects of the workshop/programming concepts did participants struggle to grasp?

Where to put curly braces and the format of if...else statements caused the majority of participants problems at some point. Students reported encountering few other issues when asked although this may be down to an unwillingness to ask questions on their part.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

Not applicable.

Was a considerable amount of time spent diverging from workshop related activities?

No. Very little time was spent diverging from Workshop related activities. The presence of two teaching members of staff at all times likely helped to ensure that students remained “on task”.

Any other points that need to be noted?

In the main the workshop seemed to go well. It should be noted that almost all participants had no programming experience whatsoever and that progress was satisfactory. The scheduled teaching content for the day was completed with around 30 minutes to spare. As a result the Datatypes segment from the second part of the workshop was brought forward. It was noted how the concentration of some participants appeared to wane at times. Whilst these participants did not become disruptive as a result, at a couple of points it became clear that some participants were not 100% focused. The School/Sixth Form normally implements lessons which last no more than 55 minutes. Considering that the workshop is intensive and is spread over several hours (even considering regular breaks were given) this may have something to do with students occasionally losing focus.

All participants were offered hand-outs of slides but all declined. It is possible that hand-outs may not be used as frequently in the participant’s normal learning environment and, as such, they refused the paper version of the slides.

It was also noticed how some (but not all) participants seemed very reluctant to experiment with their code. It almost appeared as if some students were worried about “breaking” the software or wanted a working completed solution first time every time. Again this may stem from the fact that in student’s normal environment experimentation may not always be encouraged.

Finally, despite being asked frequently, students often reported that everything was “ok” even when it might not have been. Rarely did participants seem prepared to ask questions or for help. Potentially a different method of teaching, different teacher or fear of embarrassment in front of peers may be factors responsible for this.

Student Workshop One (SW1) – Day Two

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

P7 did not attend the second day of the workshop. P15 did not attend the first day of the workshop. However, P15 adapted almost instantly to the software and tasks that were implemented and did very well. All other participants arrived on time and there were no further disruptions.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants appeared to be “on task” during the workshop despite the concepts that were introduced gradually increasing in complexity. Participant enjoyment appeared to be increased on Day 2 of the workshop perhaps owing to the fact that the workshop took place in an environment different than participants every Day One. In addition, owing to the nature of the lab students machines were closer together and there appeared to be a good sense of “togetherness”. Likewise, there were very few disruptions during the session.

Did participants seem enthusiastic throughout the session? If not, why?

The overwhelming impression I got from participants was that the session was insightful, enjoyable and different from their normal studies. P10 remarked how she found some of the fundamentals hard to take in and felt a little overwhelmed at times. Nonetheless P10 appeared to struggle less on Day 2 of the workshop. Both P6 and P8 remained attentive and engaged throughout. They displayed a willingness to try and to experiment and always wanted to perfect code whilst exhibiting a determination not to fail. P5 exhibited advanced concept knowledge and effortlessly completed the tasks set. P1 remained quiet and attentive throughout and really made an effort to learn. P2 was always advancing onto new tasks during the workshop and was clearly prepared to try. P4 always displayed a willingness to attempt tasks despite not always succeeding. P4 also seemed to take a much greater interest during Day 2 of the workshop compared to Day 1. P9 and P14, despite always being technically “on task”, often appeared to put little effort into solving the problems they encountered. They also did not display much willingness to experiment with their code. P3 put genuine effort into the programming tasks and remained interested and wanting to learn. This was despite struggling with some of the tasks set.

What aspects of the workshop/programming concepts did participants struggle to grasp?

The syntax of For Loops appeared to cause most participants confusion, particularly at first.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

P6 and P10 both voiced concerns that the in-workshop programming test implemented on Day 2 of the workshop was very challenging and more difficult than the in-workshop test given on Day 1. No concerns were voiced about the nature of the post-workshop assessment used and, indeed, it seemed to be a challenge enjoyed by the majority of participants. However, several students appeared to “give up” easily when tackling some of the later challenges, especially Task 4.

Was a considerable amount of time spent diverging from workshop related activities?

No. Participants spent virtually no time diverging from Workshop related activities.

Any other points that need to be noted?

On the whole I got the impression that the workshop was a success. On Day 2 students seemed more willing to ask for assistance and I believe that they benefited from asking questions. This may be down to a greater familiarity with myself, as they realised that learning programming is difficult and mistakes have to be made (note that few students had any programming experience whatsoever before the session) or for other reasons. By the end of Day 2 all students appeared prepared to ask either myself or their peers for help which was in contrast to Day 1 of the workshop.

Student Workshop Two (SW2) – Day One

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

All participants arrived on time and none of the participants left the workshop early.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants seemed to be firmly “on task” during the workshop. There was virtually no talking or other disruption at any point. The practitioner who sat in the session remarked how participants must have enjoyed themselves as they remained constantly focused throughout.

Did participants seem enthusiastic throughout the session? If not, why?

All participants seemed to remain extremely interested throughout the workshop. At some point almost all of the participants were observed trying to elicit additional behaviour out of the robot simulator by using their programming knowledge to “push the boundaries” (e.g. declaring extra variables, checking on the internet for new syntax). During the first day of the workshop no participant was observed to be struggling. Whilst some questions were asked by almost all participants upon explanation little further help was required.

What aspects of the workshop/programming concepts did participants struggle to grasp?

No aspects of the workshop appeared to cause any of the participants’ trouble. When participants did encounter an issue, on several occasions, the participants were seen to help one another.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

Not applicable.

Was a considerable amount of time spent diverging from workshop related activities?

Not applicable.

Any other points that need to be noted?

The session appeared to go extremely well. All participants seemed to engage and enjoy the experience. Moreover, the Programming Task which participants completed (IWPT1) seemed to offer attendees the chance to put their knowledge “to the test”. The member of staff who sat in the session seemed very enthusiastic in regards to the workshop and discussed the possibility of further future workshops. The Data Types and Loops component from Day Two of the Workshop was incorporated into Day One due to there being sufficient time remaining.

Student Workshop Two (SW2) – Day Two

a) To Be Documented Immediately During the Workshop

Issues with environment/equipment (e.g. problem with laboratory, technical failure)

Not applicable.

Issues involving participant involvement (e.g. non-attendance, missing participants)

P14 did not attend the second day of the workshop due to other commitments. P15 was also unable to attend Day Two of the workshop. P22 did not attend Day One of the workshop. However, the participant attended Day 2 of the Workshop. After a brief individual introduction to bring the participant up to speed with the Robot Simulator P22 took an active part in the workshop. It should be noted that P22 had some prior Java programming experience and this may have helped the student to adapt so quickly.

Other unidentified potentially critical issues

Not applicable.

b) To Be Documented As Soon As Possible After the Workshop

Were most participants “on task” during the workshop? If not, why?

All participants were to be “on task” and engaged throughout the workshop. Participants handled the increase in complexity well and appeared to be enjoying the experience. There were few, if any, disruptions during the session.

Did participants seem enthusiastic throughout the session? If not, why?

The overwhelming impression I got from participants was that the session was interesting and helpful. Almost all students coped well with the tasks and challenges set. Only P17 appeared to struggle at points during the session and required extra help. The remaining participants all appeared to excel during the workshop, often trying to push the simulator software by using prior knowledge to elicit new behaviour from the virtual robots they were using.

What aspects of the workshop/programming concepts did participants struggle to grasp?

Few concepts appeared to cause participants difficulty. Occasionally syntax errors would cause minor problems for participants but once these had been explained once often participants were able to overcome the issue (if it reappeared) by themselves.

Did three or more participants voice concerns about a particular aspect of the workshop?

Not applicable.

Did three or more participants voice concerns about a particular aspect of the simulator?

Not applicable.

Were any concerns raised about the nature of assessment used during the workshop?

Not applicable. The post workshop tasks seemed to be enjoyed by most if not all of the participants. The participants were evidently engaged and keen to put their new knowledge to the test. The participants level of concentration was noted by a member of staff who remarked how the deadly silence was evident of participants enjoyment and enthusiasm.

Was a considerable amount of time spent diverging from workshop related activities?

No. Participants spent virtually no time diverging from Workshop related activities and, on the contrary, several participants remained working with the simulator during lunch and after the session had finished. This was also observed to be the case, according to the participant's lecturer, after Day 1 of the workshop.

Any other points that need to be noted?

On the whole I got the impression that the workshop was a success. The participant's lecturer noted how several participants were interested in using the simulator as part of students A-Level projects the following academic year. In addition, the practitioner himself asked for a copy of the workshop slides and software as he would seek to implement these in his own lessons.

Appendix A13 – Teacher Interview Transcripts

Identification of Interview Code Categories

Details of the codes that were generated are provided below along with the following information:

- Name of the theme and code number
- Definition of each theme
- Hypothetical example of when such a code would be used
- Colour coding has been used to identify specific instances in the full interview transcripts that follow

Theme 1: The Robot Simulator

1A. Strengths

Definition of Theme: Strengths of the simulator as tool to teach introductory programming

Example: “The visual nature of the simulator encourages students to learn”

Specific Instances: ORANGE

1B. Potential Issues

Definition of Theme: Issues which did/could potentially arise as a result of using the simulator as a tool to teach introductory programming

Example: “Students get distracted by the simulator software and their attention wanders from the task”

Specific Instances: PINK

1C. Efficiency

Definition of Theme: Issues relating to the efficiency of the simulator compared to other teaching methods

Example: “Students learned variables quicker using the simulator than they would otherwise”

Specific Instances: YELLOW

1D. Teaching

Definition of Theme: Thoughts on, and potential obstacles to, teachers using the simulator in their lessons

Example: “Teachers may not have the pre-requisite programming confidence to use the simulator”

Specific Instances: TAN

1E. Simulator Modifications

Definition of Theme: Identifies if, and if so how, the simulator needs to be modified if it is to become a more effective introductory programming resource

Example: “The inclusion of FAQ menu option would help novices initially”

Specific Instances: DARK GREY

Theme 2: Student Participants

2A. Ability/Experience	
Definition of Theme:	Identification of the general ability and past programming experience of the students involved in the robot simulator sessions
Example:	“All students had no prior programming knowledge”
Specific Instances:	RED
2B. Generalisability	
Definition of Theme:	Consideration of how many students (from a typical class) would normally struggle to complete the tasks assigned during the workshop
Example:	“In a typical class five students would significantly struggle regardless of the teaching tool being used”
Specific Instances:	LIGHT GREY
2C. Enjoyment	
Definition of Theme:	Thoughts on how much students enjoyed using the simulator
Example:	“Three students commented after how they really enjoyed using the simulator”
Specific Instances:	BLUE
2D. Perceptions	
Definition of Theme:	Thoughts on whether the simulator helps to improve perceptions of programming
Example:	“The simulator helps to improve perceptions about programming as students can better relate programming to the real-world”
Specific Instances:	LIGHT RED
2E. Comparison	
Definition of Theme:	Thoughts on whether students would be more encouraged when taught using the simulator compared to traditional programming teaching methods
Example:	“The visual nature of the simulator helps students to better visualise what their code is doing compared to the traditional text based method of teaching”
Specific Instances:	PURPLE
2F. Workshop Modifications	
Definition of Theme:	Identifies if, and if so how, the workshop could be to be modified in order to improve students learning experience
Example:	“The workshop should allow greater opportunity for group discussion”
Specific Instances:	DARK RED

Theme 3: Miscellaneous

3A. Other Considerations	
Definition of Theme:	Relevant issues not covered by the other thematic codes.
Example:	n/a
Specific Instances:	GREEN

Teacher One (T1)

In regards to the participants who took part in the workshop, would you say they were typical of a similar demographic?

(The students who took part in the workshop) would be the sort of top end, the interested ones. They have done VB (before). Around 15-16 hours taught in the first term and then the second term (in the form of projects). They would probably be one of the most experienced cohorts you have got... But it was interesting how they (didn't say during the workshop), "Oh, we've already done this". And that was a big positive, they just took to it. I got a bit of flak from them saying, "Why can't you do it like this!". The fact that I've seen something, and that they have all been coming back talking about it saying positive things about it (is good).

How do you think the workshop session was received by participants?

They were very pleased with it. They were happy with it. They obviously got something out of it because although they have done that kind of stuff before it doesn't harm to come back and do it again. (Most of the) students have done VB, three have done Java.

In terms of the workshop, do you think that it helped to save time (i.e. would 'traditional' methods of teaching the same programming concepts take longer)?

Since I was planning on pinching it and using it in September I would say definitely. The way it worked I think is a very good idea. The fact that you're introducing quite complex things but in a straight forward way is the way to go. The problem that I have got with the way we've been doing it is that you're going the other way. We've tried to build stuff up and that cuts people out of the race as they try to build bits and pieces to put together. You've done it the other way as you have got these modules and then you (say let's) do something with these pre-written bits and I think that is the better way to go personally. The way it works they are not having to worry about the nitty gritty stuff, it just works. It's making the connections of how to use those building blocks and I think that it's a much more effective way of doing it actually.

Compared to traditional methods, do you think people participants who take part in this workshop would be more encouraged?

I think they would, although I think the difficulty is when they have not done anything like this before. You get this huge difference in pace from the ones who are struggling to type the thing up properly to the others who are just away, who have just understood it, and who are gone. With traditional mechanisms, (like) "Hello, World!" which is relatively simple, the moment you move beyond that that's when (the participants) start to spread out so some will move fairly slowly and some much more quickly and that creates little problems.

How many hours would it normally take to cover the same range of material, in a similar amount of depth, using 'traditional' methods?

I would think probably, if we were doing it the 'traditional way' you would be talking about double, at least. In terms of efficiency if you were to run that workshop in that way and then follow that up for the ones who are struggling a bit we'd see a greater gain in productivity I think very quickly.

Typically, when compared to a similar sized workshop cohort, how many students would you normally expect to significantly struggle/not take much of an interest regardless of the programming intervention that is being used?

You might get one or two. It varies depending on the group. One or two would find it difficult I would have thought. We don't tend to get (students) who just sit back (and don't try). If they are in they'll give it a go. They may struggle but they will at least try it.

What modifications would you suggest in order to improve the workshop?

I don't think I would really do very much with the software. It works. I'd be inclined to leave it alone. I see a lot of stuff that is 'improved' but is not necessarily an improvement. The idea of (the robot simulator) is to be a tool to get kids thinking about designing and building something and it does that job. I'd be happy to use (the simulator) in a classroom without further modification as it does its job. I'd leave it alone.

As far as the workshop goes you sort of got there with the extra bits and support. Some of them will listen and take on board what you said... what you really might want is something that summarises (each) particular concept so they have got something they can take away. Almost like a little summary booklet. Not slide-by-slide-by-slide but (something which describes) the key concepts, this is how it works etc. No more than two sides. Something which they can actually take away at the end (so participants) can refer back. And then the build-on exercises you had right at the end, the four tasks, would stretch it and take it further whether it be an assessment, assignment or whatever. That would be ideal really. You have got all of the right bits, I think, it's just sort of putting it together into a longer running package like we (the college) would do. I am perfectly happy to experiment with that and see where that goes from our end.

Do you believe that a robot simulator is an effective tool for supporting the learning of introductory programming?

As I have got two of the guys (who took part in the workshop) talking about doing this for a project I'd imagine yes! [LAUGHTER]. I think it is a good idea because they can see something happening as a result of what they are doing.

Do you believe that a robot simulator improves novice's perceptions of the subject?

It can do. If you put it into a bigger context, so if we were looking at sort of applications, then you can immediately see how this would be useful or whatever. In terms of the programming, they could see quite clearly being into that, being in charge of the computer. It was a productive exercise and not a chore. I think that's where a lot of the stuff with VB has got to, it's a chore. You just trudge through... until something happens whereas here (the students) were actually actively engaged in doing something and would of kept on going whether we (the staff) were there or not I would have imagined. Which is really where they need to be, if they are actively engaged (that would be good) for OFSTED. Yes they were engaged, fully engaged, with it and I didn't get a peep out of them about (not) doing anything they just wanted to solve it.

What are the obstacles for teachers using the robot simulator?

There's always the "what if scenario"... if it goes wrong, what happens? But that happens with any sort of software... the fact that you can just restart (the simulator) takes away some of the problem. I would think that you could actually use (the simulator) with a variety of people because it's self-contained and as long as they have prepped up what is going to happen and they have got the answers there I don't think there would be too much of a difficulty I think... If (teachers) could work through the workshop, and they were confident with that, then they could take that programming on. I would be quite happy to do that and I think that others would be as well because you have got the security of "this is what happens" and as you can experiment with it and as it isn't going to go horribly, horribly wrong.

In terms of the research project, can you think of any advice or anything to look for when analysing collected data? What approach would you take when investigating such a topic?

It depends on what sort of data you have got. Have you got anything gender wise? It would be worth looking at the responses from the gender perspective. An Equality and Diversity Agenda (for example ethnic diversity could be looked into and may be useful). Gender would be the obvious one because it has a big pay off when it gets to our end. We have 76 A-Level (Computing) students and three are girls. They make that choice somewhere in Year 10 (aged 14) and they don't see it again. Certainly, on that basis, I would look at that... Gender is the big thing for me... Even on an open evening, during a taster session, you don't get many girls. They look at it and think "I am off!". It's very deeply engrained in Schools that Computing is not for (girls) or whatever.

Teacher Two (T2)

In regards to the participants who took part in the workshop, how typical are they of a similar demographic?

They were a mixed ability group so they are really typical... they were really across the board. You are talking of students from Grade A right down to Grade E.

How do you think the workshop session was received by participants?

The majority seemed to enjoy it. As you would expect, probably one or two found it a bit dry but the vast majority seemed to enjoy it. There was an err on the side of ability, hitting the goals was stronger on the boys side. A couple of the girls managed to do that as well. But I was quite impressed with the fact that the girls were having a go. They seemed to do well and seemed to generally enjoy it.

In terms of the workshop, do you think that it helped to save time (i.e. would 'traditional' methods of teaching the same programming concepts take longer)?

It gives some of the concepts a more concrete outcome so it would probably save a small amount of time but I think the time would be saved by the fact that the concepts would probably sink in first time. You could do the same concepts and you could program a "Hello World!" or even just some straightforward maths, or whatever, in the same amount of time but you would probably have to go over it a couple of times without there being some kind of visual representation of what was going on... With a high ability group it probably wouldn't make an awful lot of difference (in terms of saving time). But with a medium to low ability group it probably makes a difference.

Compared to traditional methods, do you think participants who took part in the workshop seemed more encouraged during/after it?

To be honest it is too early to tell because they have not done Computing before at all, they have done some very visually Computing (like Scratch or Gamemaker) (so) I can't really compare.

Typically, when compared to a similar sized workshop cohort, how many students would you normally expect to significantly struggle/not take much of an interest regardless of the programming intervention that is being used?

(Out of a group of ten) ... a couple very high fliers, the majority middle of the road, a couple would struggle and then you might find a couple who just can't do it at all and don't try.

What modifications would you suggest in order to improve the workshop?

From a teaching point of view you would probably want to get them to attempt every instruction. So rather than give them a concept verbally and then move onto another concept verbally and then say let's try them both you would want to say, "Here's one concept, give it a go". Even if it seems trivial, try it anyway so they have all had a go then move onto the next concept because students aren't very good at retaining what you have just told them unless they actually do it. There is an adage which says something like you remember 20% of what you are told and 40% of what you have read and 80% of what you do or something like that. So the doing, the hands-on... I would just put a bit more hands-on. So literally break it up more. So, for instance, "this is how you do a count up, let's try it"... (I don't think it would be too dry) even for the high ability. If you give them too many concepts, for example think about Powerpoint, if you give them an example on a Powerpoint slide and then move on to the next Powerpoint slide you have to work on the assumption that the previous Powerpoint slide is now gone from their memory. What's on the next slide is the bit that matters. Even hand-outs would be useful because then they can always jot notes on as well. Whenever we use hand-outs we encourage the students to jot notes. So, for example, if you are talking about a concept like AND and you say its double ampersand you could just say to (students), "on your notes you highlight (this fact)". I noticed in the session a lot of students were only using one ampersand (in their code) because, even though it's been explained verbally and has been on the slides, they haven't done anything with that. Even circling it (would help) otherwise it is gone... That's the main thing. Stopping at each concept. I think extension activities (are) the key. So, on each task... there will be some who take five minutes which is your allocated time (but) there will be some who do (a task) in one minute and won't know what to do next. And that happened in the session where you had a few students who had got to that stage. The difference is you'll have some students who will say I've finished it, you've checked it, now I am going to have a go at messing around for myself but the majority of students will say I've finished it now I'll stop and wait until I am told what to do next. If you say (to the students)... if you finish (a task) can you extend it... so the main task is covered by everybody and those who do it quickly have got the opportunity to just try a little bit of a modification. I would tag it on the end of your slides just at the bottom. What we call it in education is extension activities... you wouldn't give it as a verbatim instruction because you have got all of the instructions at the top with your sample code. Your extension would always be, "can you do this bit on your own". The idea is if you can't it's no great shame because you have done the main bit that we asked you to do. From an educational point of view that's very important because you just don't know the ability of the people. Even if you sat down and said, "Here's the ability range, this is what I expect to happen" when they sit down and have a go you might find that they just take to it. As an example I have got a student who is very low ability and wants to do Computing and my first thought was that they would struggle. But we actually had a go at doing some coding and they showed a total and utter aptitude towards it and they have been doing at home, in class, and is actually one of the fastest ones in the coding side of things... so having that extension is what makes all the difference.

Do you believe that a robot simulator is an effective tool for supporting the learning of introductory programming?

I think it's great and it really puts things into a visual context so they can understand the idea of a robot moving, stopping and turning. They can't (for instance) visualise the idea of a variable as an abstract concept (as) it means nothing to them. That again is the kind of thing we are teaching all the time... you have to put it in a grounding that they are going to understand.

Do you believe that a robot simulator improves novice's perceptions of the subject?

They wouldn't have come (into the session) thinking it was going to be boring because their only experience (of programming) so far would have been geared towards the exciting. So they will have done game making or possibly animation. They will have also designed websites using HTML. Most of them will have used raw HTML rather than something like Dream Weaver. So they wouldn't have come in thinking it was (going to

be) boring. What they probably think, like everybody does is, “I am going to learn to program, great, I’ll be writing Angry Birds by the weekend and will get myself rich”. So... the way it is done in the simulator with Java, because you have pre-written a lot of the methods it does make it look easier to them than it perhaps would be if they were starting from nothing as obviously a lot of the methods are just built in and they are just calling the methods... If you go onto explain the methods later, like you did in the session, then I think that (way) is fine so they get to see about what actually goes into it and when they get to create their own methods. From a structure point of view that’s probably the right thing to do. Very rarely in the real-world do you start with nothing. You start with a bunch of pre-written libraries and away you go. So it’s probably more realistic (the way the workshop is) in terms of if they did a career in programming they wouldn’t be starting from scratch, they’d be starting (with) stuff people have already done.

What are the obstacles for teachers using the robot simulator?

For teachers there are two (things to consider). The programming knowledge itself, the government want everybody to be learning and teaching programming which is great, but (the) majority of teachers in the UK who are teaching ICT are probably not from a programming background, the majority will be from something else and they will have moved in from ICT. So you have got a subject knowledge barrier. And the second I think is probably, (even) though it sounds trivial, setting the challenges. Some people would look at a robot and think it can (only) move forward, backwards and turn round... and can’t think of anything else to do with it. So being creative (and for example) getting (the robot) to draw a spiral, to draw a figure of eight, to draw an n-sided polygon that’s where I think people would struggle. And if it seems to the kids like it is a trivial activity, you know, that’s as far as it goes, then they’ll get turned off pretty quickly. But if you show, for instance, how it’s doing the line following and the clever bits that’s what sort of keeps their interest going.

In terms of the research project, can you think of any advice or anything to look for when analysing collected data? What approach would you take when investigating such a topic?

... It’s got to come from just what you see. One of the useful things we use in teaching a lot is colour coding, traffic lighting. So if there was a way for instance of very visually flagging up, from a distance, that a student achieves a task, you know the screen goes green or something like that, you can immediately look around the room and go, “Right, excellent, everybody except four of them has managed to get that”. Otherwise you have got to go around each individual (student) and, as you know, you’ll have to stop as the first (student who) has got a problem and then you’re talking to them... looking at it from a teaching point of view, (it would enable you) to get instant feedback...

Teacher Three (T3)

In regards to the participants who took part in the workshop, would you say they were typical of a similar demographic?

I don't think they are (necessarily) the most able Sixth Formers, they are not the brightest we have got, but they are the most hard working... they want to graft.

How do you think the workshop session was received by participants?

All of them really, really enjoyed it... They all enjoyed it and when I spoke to them about it afterwards back in lesson they were all very positive.

In terms of the workshop, do you think that it helped to save time (i.e. would 'traditional' methods of teaching the same programming concepts take longer)?

We tend to teach in that way anyway... Probably it was quite prescriptive in terms of you need to put this in but it has to be at the beginning when they are learning something new. We try to get them to go off and discover and find things out for themselves so (I) am not too sure if it saves time, I don't know.

Compared to traditional methods, do you think people participants who take part in this workshop would be more encouraged?

I think it definitely encourages, yes... What we think they like about your (workshop) is the fact that it is a robotic simulator and you can hook them in with robots and show them what robots can do... It's that what's the hook... and the fact that they've got a simulator and they can do something to make something move that's what's good about yours.

Typically, when compared to a similar sized workshop cohort, how many students would you normally expect to significantly struggle/not take much of an interest regardless of the programming intervention that is being used?

If we say an average group of 15 I would say about three would be probably disengaged.

What modifications would you suggest in order to improve the workshop?

The first day (of the workshop) could have been a little bit pacier. They are used to pace. We are taught as teachers that there has to be pace. So if there are some that are sitting around because they have done what you need them to do but you are waiting for the other ones who aren't as quick to process you need to have something for the others. You need to say (to them) "they are still finishing that but why don't you try this?" ... When you are in a school it's a different situation to when you are in a University because ultimately (at University) if (students) don't pass then it's their problem not ours... but here if they are not listening to you you have got to wait for them to listen to you so don't try and talk over them or anything like that... No, I think it was great. I personally enjoyed the second day better because it was (more about) the robots.

Do you believe that a robot simulator is an effective tool for supporting the learning of introductory programming?

Perfect, it's perfect.

Do you believe that a robot simulator improves novice's perceptions of the subject?

Yes.

What are the obstacles for teachers using the robot simulator?

I've got a Business Studies degree but it's still something I am interested in. It's (about) breaking down the fear barrier for the others and I think it's done that and it is doing that. When I brought staff to you (for the teachers) workshop they were so excited so I think it helps break down the barriers of the unknown and the fear.

In terms of the research project, can you think of any advice or anything to look for when analysing collected data? What approach would you take when investigating such a topic?

Gender is the one that sticks out... I don't know if you have the data to do it but we have to look for is things like kids who are on free school meals, kids that are looked after, kids who have got English as a second language because their processing skills are different. Where they live (may be one)... someone who is on free school meals hasn't got the privileges and it has an impact. Ages of children, what term of the year (could be something to look at)... If you have got a child born on the 31st August they are 12 months younger than (everybody else). They are the sort of things we have to do data analysis on.

Do you have any other comments?

No. Thank you. It's been great.

Appendix A14 – Kobot Control Methods

Robot Movement Methods (used during the early stages of the workshop)

The `forward()` and `backward()` methods were used to control the robotic agents in the early stages of the workshop. Both methods accept integers as parameters and values that are passed to them represent time in seconds (e.g. `forward(5)` = move forward for five seconds). If no value is passed the commands are executed continually.

The `left()` and `right()` control methods instruct the robotic agents to stop and to turn at a 90 degree angle (either right or left). No parameters are passed to these methods.

Primary Robot Movement Methods Used in Kobot

`move()` is used to move forward or backwards. Speed is passed as a parameter in a range between 0 – 0.3 meters per second. To move forward a positive value is passed – e.g. `move(0.1)` and to move backwards a negative value is passed – e.g. `move(-0.1)`.

`turn()` is used to turn left or right. Rotation speed is passed as a parameter in a range up to 180 degrees per second. To turn left a positive value is passed – e.g. `turn(90)` and to turn right a negative value is passed – e.g. `turn(-90)`.

`duration()` instructs a robot on how long to perform the previous command for:

```
move(0.1);
duration(10);
```

(e.g. move forward at a speed of 0.1meters per second for 10 seconds)

Primary Robot Sensor Methods Used in Kobot

The `leftSensorValue()` and `rightSensorValue()` methods are used to calculate the distance between the robotic agent and 3D objects. Both return a double between a 0.0 and 0.8 if a

3D object is detected within a range of 0.8 meters. This is the same for the `frontLeftSensorValue()` and `frontRightSensorValue()` methods that are used during the post-workshop programming exercises. Other sensor methods (namely `leftEncoderValue()`, `middleEncoderValue()` and `rightEncoderValue()`) are used exclusively during the post-workshop exercises to detect the presence of 2D objects drawn by participants. These methods return an integer value of either '0' (if no 2D object is detected) or '1' (if a 2D object is detected).

Miscellaneous Other Methods

Several miscellaneous methods were used during the workshop, typically during just one of the completed tasks:

During the 'Line Counter' tasks a miscellaneous sensor method, `object2DDetected()`, is used. This returns a Boolean value of 'true' if the robot's middle encoder detects a 2D object.

`distanceTravelled()` provides a double which corresponds to that displayed in the 'Information Pane' and was used during the 'Pauser' and 'Shapes' tasks.

`turn360()` was only used during the 'Shapes' task and, when called in code, results in a robotic agent performing a 360 degree turn.

`endCount()` was called by participants during the 'Line Counter' challenge. When the method is called a message is displayed to the user informing them that their program will end shortly before the program is terminated.

The `pause()` method instructs the robotic agents to pause for the amount of time specified by the user (time – specifically seconds – is passed as an integer variable). The `stop()` method instructs the robotic agents to stop moving and halts the simulation. No values are passed.

Appendix A15 – Case Study Design Checklist

Item	Checklist Question	Comments
1	What is the case and its units of analysis?	An investigation into the effectiveness of simulated robots as introductory programming teaching tools. The case study is a multiple-case case study. Case One involved trainee ICT/CS high school teachers, Case Two student programmers aged 16 to 18 years old.
2	Are clear objectives, preliminary research questions, hypotheses defined in advance?	One main research question (“Is a robot simulator an effective tool for supporting the learning of introductory programming?”) and four research propositions (see Chapter Five) have been outlined.
3	Is the theoretical basis - relation to existing literature or other cases - defined?	The results of a previously completed SLR and supplementary literature search (both presented in Chapter Two) provide the theoretical basis for the case study. Chapter Three ‘Exploratory Studies’ demonstrates the viability and potential of using a robot simulator for supporting the teaching of programming.
4	Are the authors’ intentions with the research made clear?	The purpose of the study is to determine whether a robot simulator is an effective tool for supporting the learning of introductory programming (see Chapter One ‘Research Purpose’).
5	Is the case adequately defined (size, domain, process, subjects...)?	See Chapters Five for information on the case study design.
6	Is a cause–effect relation under study? Is it possible to distinguish the cause from other factors using the proposed design?	See Chapter Eight for a consideration of rival explanations and threats to validity.
7	Does the design involve data from multiple sources (data triangulation), using multiple methods (method triangulation)?	The case study design involves collecting multiple forms of data using multiple data collection methods (as detailed in Chapter Five). A triangulation strategy has been adopted as outlined in Chapter Eight.
8	Is there a rationale behind the selection of subjects, roles, artefacts, viewpoints, etc.?	Described in Chapter Five.
9	Is the specified case relevant to validly address the research questions	Expert review of the case study protocol, use of multiple sources of evidence and the establishment of a chain of evidence help to overcome any potential issues with construct validity.
10.	Is the integrity of individuals/organisations taken into account?	As demonstrated in Appendix A4 the project received full ethical approval from Keele University’s Ethical Review Panel.

Based on the case study design checklist suggested by Runeson and Höst (Runeson & Höst, 2009)