# The effectiveness of simulated robots for supporting the learning of introductory programming: a multi-case case study

Louis Major, Theocharis Kyriacou & Pearl Brereton

**To cite this article:** Louis Major, Theocharis Kyriacou & Pearl Brereton (2014) The effectiveness of simulated robots for supporting the learning of introductory programming: a multi-case case study, Computer Science Education, 24:2-3, 193-228, DOI: 10.1080/08993408.2014.963362

**To link to this article:** http://dx.doi.org/10.1080/08993408.2014.963362

Published online: 30 Sep 2014.

*Computer Science Education*, 2014
Vol. 24, Nos. 2–3, 193–228, http://dx.doi.org/10.1080/08993408.2014.963362

# The effectiveness of simulated robots for supporting the learning of introductory programming: a multi-case case study

Louis Major[a]*, Theocharis Kyriacou[b] and Pearl Brereton[b]

[a]*Faculty of Education, University of Cambridge, 184 Hills Rd, Cambridge CB2 8PQ, UK;* [b]*School of Computing and Mathematics, Keele University, Staffordshire ST5 5BG, UK*

(*Received 28 April 2014; accepted 16 August 2014*)

This work investigates the effectiveness of simulated robots as tools to support the learning of programming. After the completion of a systematic review and exploratory research, a multi-case case study was undertaken. A simulator, named Kebot, was developed and used to run four 10-hour programming workshops. Twenty-three student participants (aged 16–18) in addition to 23 pre-service, and 3 in-service, teachers took part. The effectiveness of this intervention was determined by considering opinions, attitudes, and motivation as well as by analysing students' programming performance. Pre- and post-questionnaires, in- and post-workshop exercises, and interviews were used. Participants enjoyed learning using the simulator and believed the approach to be valuable and engaging. The performance of students indicates that the simulator aids learning as most completed tasks to a satisfactory standard. Evidence suggests robot simulators can offer an effective means of introducing programming. Recommendations to support the development of other simulators are provided.

**Keywords:** teaching; learning; java; programming; robots; robotics; simulated; simulator

## 1. Introduction

Many students new to programming hold the belief that the subject is difficult, and laborious, to learn (Esteves, Antunes, Fonseca, Morgado, & Martins, 2008). Negative opinions which novices may hold beforehand are often realised when their first programming task is to complete an unimaginative assignment such as "Hello World!" (Talbot, 2000). In addition, lacklustre initial interactions with programming can later contribute to high-course withdrawal rates and poor academic performance (Kinnunen & Malmi, 2006). To help overcome such issues the use of robots to support the teaching of programming principles has long captured the attention of computer science educators (Fagin, Merkle, & Eggers, 2001). This is partly because

*Corresponding author. Email: lcm54@cam.ac.uk robots are exciting to work with (Sklar, Parsons, & Azhar, 2006), appeal to a variety of people of different ages and backgrounds (Price, Richards, Petre, Hirst, & Johnson, 2003), and are capable of evoking complex emotions in humans (Braitenberg, 1986).

In this article, an investigation into the use of a robot simulator, as a tool to support the learning of programming, is reported. A simulator (named Kebot), and associated 10-hour programming workshop, has been developed. Participants including students and pre- and in-service high school teachers have taken part in empirical research. The case study methodology was used as this can provide a fuller understanding of how an intervention works compared to alternative research methods.

Effectiveness was determined through an analysis of data collected about students' programming performance and consideration of participants' opinions, attitudes, and motivation. The specific criteria used to determine effectiveness of the Kebot robot simulator are as follows:

(1) Enjoyment (do participants find the learning process enjoyable).
(2) Value (do participants find the approach valuable).
(3) Effectiveness (do students learn programming concepts covered).

In the remainder of this section, an overview of related work and the motivation for the study are provided. In Section 2, the case study methodology is outlined and details of Kebot and the workshop are given. The results of Case One "Trainees" and Case Two "Students" are presented in Sections 3 and 4. Section 5 is dedicated to a discussion and includes a set of recommendations to support the future development of robot simulators for supporting the learning of programming. In Section 6, a conclusion and suggestions for future work are offered.

### *1.1. Background*

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). Visual learning, which involves learning based on analogy and abstraction, has successfully supported the teaching of programming (Miliszewska & Tan, 2007; Pears et al., 2007). Visual learning engages students more fully in the ideas presented and can make the learning experience stronger (Lahtinen & Ahoniemi, 2009). It can also increase the motivational aspects of a programming course while enabling an easier transition to more advanced tasks (Kasurinen, Purmonen, & Nikula, 2008; Nevalainen & Sajaniemi, 2006). A number of visual environments are currently used to introduce programming including Scratch (Resnick et al., 2009) and Alice (Dann, Cooper, & Pausch, 2006). This paper focuses on the use of one particular visualisation tool, a robot simulator.

The use of robots to teach programming has long captured the attention of computer science educators (Fagin et al., 2001). 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, was created (Solin, 2013). Developed by Pattis (1981; Pattis, Roberts, & Stehlik, 1995), 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). Both Logo and Karel have influenced, and continue to influence, approaches to the teaching of programming and other computational subjects. From the modelling of decentralised systems using the StarLogo language (Resnick, 1994) to more recent developments concerned with the introduction of programming fundamentals (including GvR (Yadin, 2011), RUR-PLE (desJardins, Ciavolino, Deloatch, & Feasley, 2011) and StarLogo TNG (Paliokas, Arapidis, & Mpimpitsos, 2011)), Logo and Karel continue to shape approaches used by computing educators.

Recent advances in technology 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, Kyriacou, & Brereton, 2012a), which 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 including the Scribbler (Cowden, O'Neill, Opavsky, Ustek, & Walker, 2012), Arduino (Martin & Hughes, 2011), and Koala (Čermák & Kelemen, 2011) robots.

A number of problems have, however, been associated with the use of physical robots in the classroom. These include high financial cost associated with purchasing and maintenance, 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, Congdon, Fagin, Hwang, & Klassner, 2001; McWhorter & O'Connor, 2009). Partly due to these issues, the use of simulated robot tools has been investigated (Flot, Schunn, Liu, & Sharp, 2012). Robot simulators may offer an opportunity to overcome the problems with physical robots (Kammer, Brauner, Leonhardt, & Schroeder, 2011) while retaining key benefits of the approach. 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, Schunn, Flot, & Shoop, 2013). The need to determine the motivational affordances of student and teachers who use virtual robot environments, through qualitative measures, however, remains (Liu et al., 2013).

## *1.2. Motivation*

This work is motivated by the difficulties associated with the learning and teaching of introductory programming, and the authors' interest in how the use of simulated robots may help to overcome these. A systematic review (SR) was undertaken to investigate the effectiveness of using robots to support the learning of programming (Major et al., 2012a). SRs are trustworthy, rigorous, and auditable tools that involve the collection and summary of all existing evidence on a topic and may help to identify gaps in current research (Kitchenham, 2004; Kitchenham & Charters, 2007). For the SR nine electronic databases, the proceedings from six conferences and two journals were searched for evidence. After applying exclusion criteria, and performing validation exercises, 36 papers were included in the SR. Of these papers, 25 examined the effectiveness of physical robots, 7 examined the effectiveness of simulated robots, and 4 examined the use of physical and simulated robots together. In total, 26 papers report robots to be effective when used to teach introductory programming. The potential to further investigate the use of robots persisted, however, particularly with regard to simulated robots. This is because, the quality of the seven papers related to simulated robots was judged to be poor as: four offer a "lessons learned" account and provide no empirical data (Becker, 2001; Buck & Stucki, 2001; Enderle, 2009; Ladd & Harcourt, 2005); one describes the results derived from interviews as being non-generalisable as only four novices were involved (Borge, Fjuk, & Groven, 2004); one specifies the involvement of 15 participants, and the use of a questionnaire, but presents no data (Lemone & Ching, 1996); and one describes lessons, involving 20 students, but does not undertake detailed analysis (Satratzemi, Xinogalos, & Dagdilelis, 2003). The SR provided additional motivation for the work that followed.

Following the SR, an early version of the Kebot robot simulator was used during exploratory empirical research to seek an initial insight using such a tool, to gain feedback and to help with the generation of ideas. Details of this work have been disseminated previously (Major, Kyriacou, & Brereton, 2011). Whilst the research design of the exploratory studies was informal, several important findings were established. Two day-long sessions involving 23 trainee high school teachers and 10 in-service high school staff were held. Pre- and post-workshop questionnaires were used to collect participants' opinions on the potential of using simulated robots as a means of supporting the learning of programming. Feedback was gained on how best to develop a tool, and associated workshop, in the future. Perceptions of simulated robots were found to be positive despite the limitations of the exploratory research.

## 2. Method

This section describes the overall multi-case case study methodology used to investigate the effectiveness of a robot simulator for supporting the learning of introductory programming. A simulator, named Kebot, was developed and used to run four 10-hour programming workshops. Details of the methodology, Kebot, and the workshop are outlined.

### *2.1. Case study methodology*

Case studies are strategies for research that involve an empirical investigation using multiple sources of evidence (Robson, 2011). They have been used to support research in fields including education (Merriam, 1998), robotics (Burdea et al., 2012), software engineering (Verner, Sampson, Tosic, Bakar, & Kitchenham, 2009), and the teaching of programming (Jones, 2010). Case studies provide a deeper understanding than controlled experiments (Runeson, Höst, Rainer, & Regnell, 2012) whilst remaining are capable of achieving scientific objectives (Lee, 1989).

Replication of case study is not possible as it is for experiments. “In-case replication” (e.g. running workshops twice with independent groups) and taking multiple measures of an event can be effective validation strategies (Gibbert & Ruigrok, 2010). While a “traditional” baseline cannot be used with case study, a qualitative baseline can be established (e.g. by asking participants to compare their previous programming learning experience to the one using Kebot). The use of an experimental strategy was considered to be unsuitable due to potential difficulties manipulating behaviour directly, precisely, and systematically (Yin, 2009). This is because there was no way to ensure sufficient control over variables (e.g. research setting, participants, and test instrumentation) other than the chosen independent variables (Easterbrook, Singer, Storey, & Damian, 2008). Experiments also require decisions to be made in advance with regard to what variables to ignore, which can lead to important findings being overlooked (Easterbrook et al., 2008).

A protocol based on the one described by Brereton (Brereton, Kitchenham, Budgen, & Li, 2008) was developed, reviewed by an expert (Barbara Kitchenham of Keele University) and disseminated (Major, Kyriacou, & Brereton, 2012b). The case study asks the following question:

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

Two propositions are addressed:

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

**P2** A robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience.

### *2.2. Participants*

A multi-case case study was undertaken. Case One (discussed in Section 3) involved trainee Information Communication Technology/Computer Science (ICT/CS) high school teachers, all who had some programming experience. Case Two (discussed in Section 4) involved students, aged 16–18 years old, enrolled on a further education course (i.e. a post-compulsory high school, pre-University course). Data from interviews with three in-service teachers (each of whom was associated with one of the student cohorts) have also been used. Participants were assigned to a PC, read an information sheet, and completed a consent form and given a code number.

### *2.3. Data collection and analysis*

The research question and propositions are addressed as follows:

- By using *questionnaires* to determine participants' views before and after the workshops.
- By using *programming tests* to determine programming progress of students.
- By *interviewing* three current high school teachers to determine their views.

The data collection and analysis strategies are outlined in Sections 3 and 4. Details of workshop tasks, including consideration of relevant pedagogy and background literature, in addition to data collection instruments themselves, are available elsewhere (Major, 2014).

### *2.4. Kebot: a robot simulator for supporting the learning of introductory programming*

Influenced by the findings of the SR and exploratory research, a robot simulator was developed. The simulator is referred to as Kebot (derived from the words Keele Robot – KEele-roBOT). Kebot is modelled after the Mark III robot which is designed by the Portland Area Robotics Society.[1] Kebot was devised by two authors (TK was the originator before LM later added a number of additional features). Information related to the development of educational software, informed design decisions taken (ANSI, 2001; Beale & Sharples, 2002; Squires & Preece, 1999). The BlueJ integrated development environment[2] is used with Kebot.

When Kebot is loaded, a user's code is placed in one of the five robot classes (named Gates, Berners, Jobs, Gosling, and Page). Each robot offers a different functionality. Features of the Kebot GUI and arena are highlighted in Figure 1. This arena is viewed from a top-down, 2D, perspective. The simulation is controlled by selecting the Start and Pause buttons. Various methods are used depending on the task. Robots have a full range of 360-degree movement. Both 2D and 3D objects can be drawn in the arena. Arena backgrounds can be saved and loaded. Kebot allows:

- Real-time interaction with a simulated robot.
- Users to customise robots' environments with objects.
- Coding in Java.
- 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 restricted environments where robots inhabit a grid-based world).

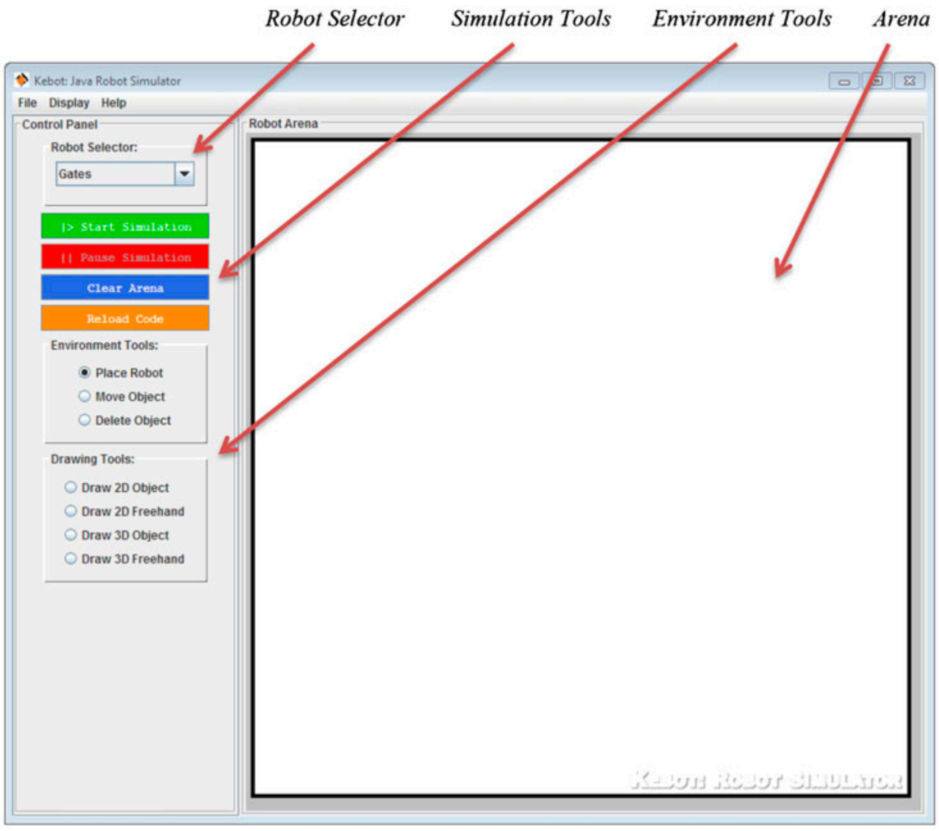


Figure 1. Annotated screenshot of the Kebot robot simulator.

It is important to note that Kebot is considered to be a simulation, not a microworld, as it does not present learners with the "simplest case" of a domain (Rieber, 1996). While Kebot offers a sophisticated representation of a real-world robot, it is not as complex as professional simulation software such as Webots.[3]

## 2.5. The Kebot workshop

The ACM and IEEE Joint Task Force Computer Science Curriculum (ACM/IEEE, 2008) influenced what concepts were taught. These guidelines transcend geographic boundaries (Douglas, Farley, Lo, Proskurowski, & Young, 2010). The fundamental constructs outlined in the subsection of the report entitled "Programming Fundamentals" were identified (see Table 1). To teach these concepts, a coverage time of nine hours is recommended.

Previous research influenced workshop content. LM delivered the workshop. Three novice programmers were involved in a pilot. Keele University's research ethics panel approved this study. 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.

Table 2 displays the workshop format. Details of data collection activities are in bold. Figure 2 displays an example task completed early in the workshop called "Line Tracer." This involved the passing of parameters to methods in order to instruct a robot to follow (or "trace") a pre-drawn route. Later, participants begun to elicit more complex behaviours from robots including "scared," "seeking," "avoiding," and "curious" behaviours. Kebot and workshop materials (e.g. workshop slides) are freely available for download and modification.[4]

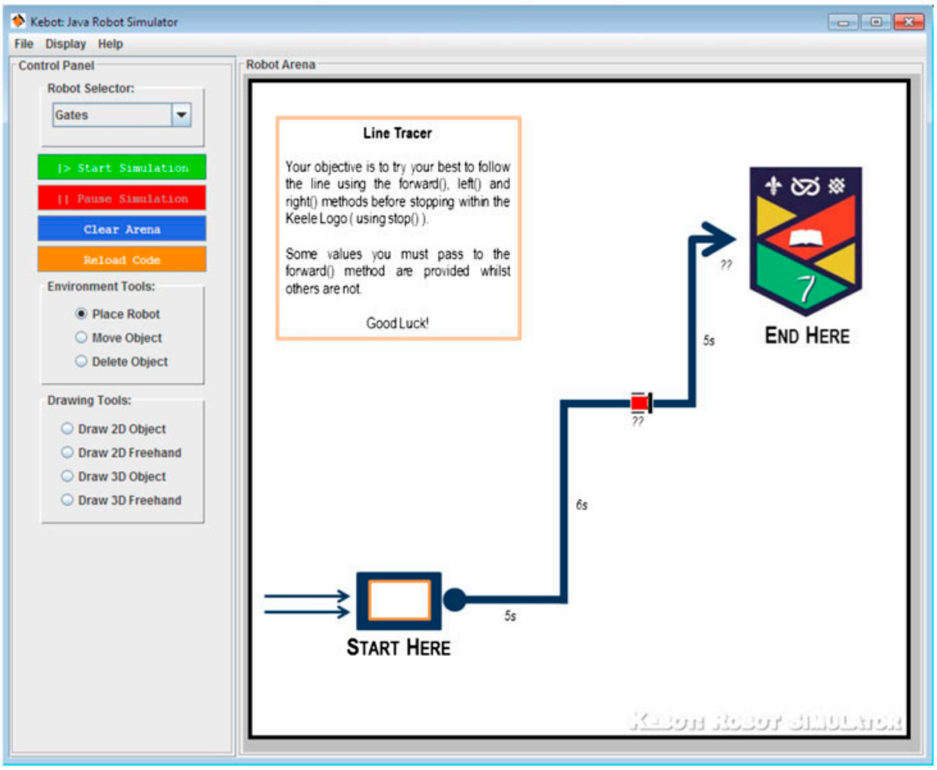
When developing the workshop, the constructivism theory was considered. Constructivism proposes that humans generate knowledge from

Table 1. Fundamental programming constructs identified by the ACM/IEEE.

| Fundamental programming constructs |
|---|
| 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 |
| **Minimum core coverage time: 9 hours** |

Table 2. The Kebot workshop format.

| | Main programming activity | Time required |
|---|---|---|
| Day One | Basic syntax and semantics of a higher-level language | 1 hour |
| | Variables and Constants | 30 minutes |
| | Logical Expressions | 1 hour |
| | Counting, JOptionPane and Nested Statements | 45 minutes |
| | **In-Workshop Programming Exercise One** | **15 minutes** |
| | Introducing the Remaining Data Types | 15 minutes |
| | While and Do While Loops | 45 minutes |
| | Method Creation | 30 minutes |
| Day Two | Conditional and Iterative Control Structures | 2 hours 15 minutes |
| | **In-Workshop Programming Exercise Two** | **15 minutes** |
| | For Loops | 45 minutes |
| | Arrays | 30 minutes |
| | **Post-Workshop Programming Exercise** | **45 minutes** |



Line Tracer

Your objective is to try your best to follow the line using the forward(), left() and right() methods before stopping within the Keele Logo ( using stop() ).

Some values you must pass to the forward() method are provided whilst others are not.

Good Luck!

Figure 2. A screenshot of the “Line Tracer” workshop task.

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) and as programming can be viewed as a constructivist activity (Wulf, 2005).

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.

## *2.6. Study limitations*

Like all empirical research, the study had limitations. The duration and context of the workshop limits the generalisability of findings as the study took place over four two-day periods and not over full semesters or year-long courses. Potentially, the approach may have been more (or less) effective if it was implemented for a longer period.

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 Case One and Case Two workshops were run twice with independent groups), and the use of validation strategies (such as a search for rival explanations), however, ensure that the findings of the case study make a significant contribution to knowledge. 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, are believed to compensate for this absence of a traditional baseline while 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.

Only constructs identified by the ACM/IEEE were introduced, and evidence on effectiveness for supporting the learning of advanced concepts cannot be offered. While a number of concepts and learning objectives are outlined by the ACM/IEEE, specific guidance is limited. Judgements needed to be made, therefore, with regard to the detail required to cover each topic.

In addition, it is not possible to determine the generalisability of delivered content to other programming curricula.

As Java was used, it cannot be ascertained how generalisable findings are compared to alternative programming languages. Java is considered, however, to be a general purpose language and shares commonalities in its fundamentals with other languages such as C#. Moreover, only basic programming concepts were introduced and advanced language features were not drawn upon.

The intensive nature of the workshop may have affected the performance of some participants, especially ones not suited to long periods of concentrated learning. Whilst it was intended that the workshop would be relaxed, it was not possible to truly recreate "real-world" conditions. This may have led to changes in performance and more positive or negative results due to the fact that participants were aware that they were being observed.

Participants may be inclined to compare Kebot to their experience using gaming systems (such as the Xbox One). Kebot 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 is required and this can take up significant hard disk space.

The workshop leader may have unconsciously affected results. Despite having teaching experience, LM alone was responsible for delivering the sessions. As workshops progressed, this may have led to a growth in confidence and a change in delivery style. Researchers may also get to know those involved (and favour participants) or have vested interest (having developed an approach). Questionnaires and interviews are self-reported and can lead to issues such as exaggeration. As participants self-selected to be involved, they may have greater interest in programming compared to others.

## 3. Case One: Trainee Teachers

In this section, details of Case One, "Trainee Teachers," are presented.

### *3.1. Participants*

Twenty-two trainee ICT/CS teachers took part. All had programmed previously as a result of educational or industrial experiences. This allowed prior learning experience to be compared to learning experiences using Kebot, in addition to establishing views on the effectiveness of a simulator. Two workshops were held:

*Trainee Teacher Workshop One (TTW1)* – 17 participants (9 males and 8 females).
*Trainee Teacher Workshop Two (TTW2)* – 5 participants (2 males and 3 females).

All trainees were enrolled on the Keele University's ICT Postgraduate Certificate of Education course, which they all completed successfully. Cohorts were registered on the same course during different academic years. The full complement enrolled on each course took part. TTW1 took place in June 2012 while TTW2 took place in October 2012. Workshops were hosted at the Keele University.

### *3.2. Data collection and analysis*

Pre- and post-workshop *questionnaires* allowed data to be collected. They were completed in less than 10 minutes and were distributed at the start and end of the workshops. These instruments were designed to collect both quantitative and qualitative data and consisted of open and closed questions. Lessons learned during the exploratory studies influenced the layout of the questionnaires. Threats to the validity of the questionnaires (both those used during Case One and Case Two) are that some included items may not have measured what they were intended to measure and questions were biased in a way that 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 two members of the research team (to determine whether the inclusion of each item was reasonable) in addition to consideration of relevant literature related to questionnaire design (Oppenheim, 2000). It should also be noted that the questionnaires shared significant similarities to those used during the exploratory studies (reported in Section 1). 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.

Case One allowed an opportunity to create and test the in- and post-workshop *programming exercises* in advance of workshops involving students. These exercises were not used to assess learning as they were not designed for use by adults with prior programming experience (as was known to be the case with all Case One participants).

### *3.3. Results*

In this section, data collected during TTW1 and TTW2 are presented.

#### *3.3.1. Pre-workshop questionnaire*

Twenty-two participants completed the pre-workshop questionnaire, all of whom had some programming experience. The minimum number of languages used previously was 1 (four participants), the maximum 10 (one participant), and the mean 3.5. Data were collected about participants' most recent experiences of learning programming (see Table 3).

### 3.3.2. *Post-workshop questionnaire*

Twenty-one participants attended Day Two of the workshop and completed the post-workshop questionnaire. One participant who attended Day One was absent. Participants were asked about their workshop enjoyment, the programming tasks set, and their views on Kebot (see Table 4).

Participants were asked to provide their opinions of Kebot, the programming support received, and the workshop presentation on a scale of 1–5 (not at all effective) (extremely effective) (see Table 5). Note that one participant did not respond.

### 3.3.3. *Programming exercises*

As indicated earlier, the programming exercises were used only for validation purposes during TTW1 and TTW2 and so no analysis of the results was carried out.

The actual exercises used during TTW1 and TTW2 were different. Feedback provided during TTW1 established a need to modify all instruments and so, following TTW1, changes were made to these. The validity of the revised exercises is considered in Section 5.

Table 3. Trainees' experience of learning their most recent programming language.

*"How did you learn this language?"*

| | Self-taught | Part of a course or education program | Specify own response |
|---|---|---|---|
| Number of responses | 5 | 16 | 1[a] |

*"How much time did you spend learning this language?"*

| | Less than one week | One week to two months | Two months to four months | Over four months | Unable to determine |
|---|---|---|---|---|---|
| Number of responses | 4 | 5 | 9 | 1 | 3 |

*"How well do you feel you learnt this language?"*

| | Learnt very little | Became familiar with most concepts | Became competent | Became expert |
|---|---|---|---|---|
| Number of responses | 3 | 12 | 5 | 1 |

*"Which of the following best describes your past programming experience?"*

| | Didn't like | Indifferent | Enjoyed |
|---|---|---|---|
| Number of responses | 5 | 8 | 9 |

[a]The one open response received specified "Code Academy".

Table 4. Trainees' opinions of their workshop experience and effectiveness of Kebot.

*"In regards to your programming experience during the workshop, which of the following best describes your enjoyment?"*

| | Enjoyable | Indifferent | Not enjoyable |
|---|---|---|---|
| Number of responses | 17 | 3 | 1 |

*"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?"*

| | Yes | Not sure | No |
|---|---|---|---|
| Number of responses | 19 | 1 | 1 |

*"Would you consider using the robot simulator as a tool to teach programming in your own lessons in the future?"*

| | Yes | Not sure | No |
|---|---|---|---|
| Number of responses | 20 | 1 | 0 |

*"If you have previously learnt a programming language, was your previous introduction to basic programming ..."*

| | Much less effective | Less effective | About the same effectiveness | More effective | Much more effective |
|---|---|---|---|---|---|
| Number of responses | 5 | 6 | 7 | 3 | 0 |

Table 5. Trainees' opinions on the effectiveness of elements of the workshop.

| Workshop component | Mean score (maximum of 5) | Score breakdown (by no. of responses)[a] | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Kebot simulator | 4.35 | 0 | 1 | 2 | 6 | 11 |
| Programming support | 4.0 | 2 | 1 | 2 | 5 | 10 |
| Workshop presentation | 4.15 | 0 | 2 | 3 | 5 | 10 |

[a]1 (not at all effective) to 5 (extremely effective).

## 4. Case Two: Students

In this section, details of Case Two, "Students," are presented.

### *4.1. Participants*

Twenty-three students took some part with 18 attending the full workshop. All were enrolled on a further education (FE) course and aged 16–18 years old. By involving FE students, it was believed that results would be generalisable to students in later high school (aged between 15 and 16) in addition to those in early higher education (aged around 18). Participants were enrolled at different colleges. Two workshops were held:

*Student Workshop One (SW1)* – 12 participants (7 males and 5 females) enrolled on an ICT course. Ten students completed the workshop.
*Student Workshop Two (SW2)* – 11 participants (all male) enrolled on a Computing FE course. Eight students completed the workshop.

An additional data source was also used:

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

SW1 took place in early July 2012, while SW2 took place in mid-July 2012. The workshops were jointly held in participants' own education institute and at the Keele University.

### 4.2. Data collection and analysis

*Questionnaires* were adapted from those administered during Case One. Modifications were made so that the questionnaires were better suited for completion by students with minimal programming experience opposed to adults with some degree of programming knowledge. These changes included revising the wording of certain questions and removing those questions that were not appropriate (e.g. questions that consider past teaching experience, etc.). 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).

Two multiple-choice/constructed response *in-workshop tests* were used to evaluate understanding of syntax and program behaviour as suggested by previous work (McCracken et al., 2001). Exercises were marked out of 10 and completed under "test conditions." LM initially marked completed exercises before these were second marked by PB.

A *post-workshop programming exercise (PWE)*, consisting of four tasks, was completed. This required 35–40 minutes. Designed to evaluate programming proficiency, the post-workshop exercise is an example of performance-based assessment (McCracken et al., 2001). The post-workshop programming exercise (PWE) was completed under test conditions. A robot that was not encountered during the workshop, Page, was programmed. Page differed from other robots as it had different sensors. Using Page enabled deep learning to be established, as participants had no option but to adapt their knowledge. Deep learning is where a learner aims towards understanding whereas surface learning is where learners aim to simply reproduce material without understanding it (Case, 2008). Code from earlier exercises could not be copied as Page, and associated control methods, differed. Programs were collected and graded according to the following criteria:

**A** – Code shows evidence of deep learning as knowledge gained during the workshop was 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 was 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 was made to solve the new problem. An attempt may have been made to simply copy previous code without adapting it. Less than 50% of code is correct.

It was decided that if around[5] three-quarters of participants were awarded an A or B for a task, then this would provide evidence of learning. Two authors (LM and TK) marked code jointly to ensure consistency. Letter grades were used to rate programming performance (Lister & Leaney, 2003). All exercises were designed to address the learning objectives identified by the ACM/IEEE.

The views of three in-service FE teachers were collected during semi-structured teacher *interviews*. Interviews lasted for 15–20 minutes and were conducted one-to-one. Thematic coding, according to guidelines outlined by Robson (2011), was undertaken. Extracts were presented. Coding involves the identification of text that exemplifies an idea before linking these with a code (Gibbs, 2007). The initial coding (by LM) was subject to peer examination (by PB) to ensure consistency.

### *4.3. Results*

#### *4.3.1. Pre-workshop questionnaire*

Twenty-one participants (11 SW1, 10 SW2) attended Day One and completed the pre-workshop questionnaire. Two SW1 participants had previous programming experiences that had been self-taught, enjoyed, and involved the use of several languages (although not Java). Asked if this experience was challenging, trivial, or indifferent, one stated it was challenging while the other was indifferent. The nine other SW1 participants had not encountered programming.

As documented in Teacher Interview 1, the SW2 group had been introduced to elements of Visual Basic (VB) several months before. For six participants, this was their only exposure to programming. Four SW2

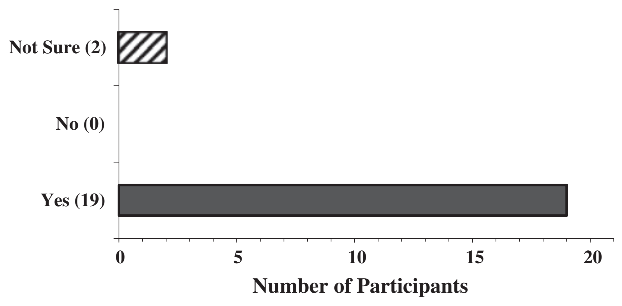


Figure 3. Students' pre-workshop responses to the question, "Would you consider learning to program, in your own time, if you were given appropriate support?"

participants had attempted to learn programming by themselves. Two had previously attempted to learn Java and, with regard to proficiency, described themselves as competent and beginners. All SW2 participants stated they had enjoyed their previous programming experience. For one, this experience had been challenging while the other nine were indifferent. Of the 21 participants, 19 responded they would consider learning programming in their spare time while 2 were unsure (see Figure 3).

### *4.3.2. Post-workshop questionnaire*

Post-workshop responses, from 18 participants (10 SW1, 8 SW2) who attended the full workshop, are considered in this subsection. Data from two participants, who attended Day Two, but not Day One, have been omitted. Figure 4 displays views on the effectiveness of Kebot. Figure 5 displays responses to the question, "Has the robot simulator helped to improve your perception of programming?"

Also investigated was whether Kebot helped to dispel programming stereotypes. Six participants believed Kebot had helped to dispel stereotypes, five were unsure, and three responded no. Three others replied that they did not know of any stereotypes.

Ten participants had encountered introductory programming before the workshop. These compared their previous introduction to programming. Data are displayed in Table 6.

Participants specified aspects they liked and disliked about Kebot. Forty-one liked aspects and 18 disliked aspects were identified. The visual nature of Kebot was identified and how Kebot allows visualisation of code was highlighted:

> You have a representation of what you spent your time doing

The nature of simulated robots, specifically that they are engaging and fun, was mentioned:

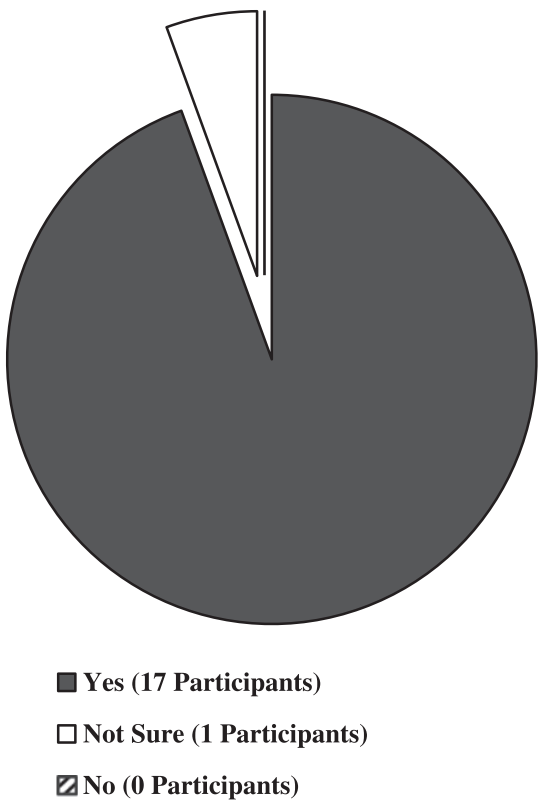


Figure 4. Students' responses to the question, "Do you believe that the robot simulator offers an effective method of introducing basic programming concepts, which you have been taught, to novice programming students?"

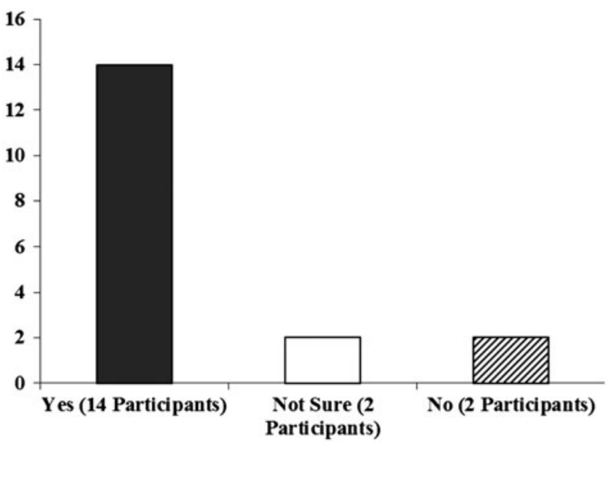


Figure 5. Students' opinions on whether the robot simulator helped to improve their perceptions of programming.

Table 6. Comparison of Students' previous programming learning experience to the one using the Kebot robot simulator.

| "Was your previous introduction to basic programming …?" | | | | | |
|---|---|---|---|---|---|
| | Much less effective | Less effective | About the same effectiveness | More effective | Much more effective |
| Response from participants with prior programming experience (n. 10) | 0 | 6 | 1 | 3 | 0 |

> Simple things seemed more interesting. The practical was more engaging. Robots are awesome

How Kebot was accessible and easy to use was highlighted in four responses including:

> Easy to use interface. Easy preview and simulation of programming code … Good scenarios and arenas

The tasks completed using the simulator, in particular that they were interesting and new, were mentioned. This was in addition to the simulator being:

> Easier than full-on programming. More enjoyable. (You) can see working with robots

Negative comments highlighted issues with Java programming as opposed to Kebot itself:

> Braces. Could be difficult to understand. Difficult to start

Limitations of the simulator, in addition to suggestions on how it could be improved, were recorded:

> Only one robot at a time. Not enough sensors

> Help and selection … give hints on what can be added to code

> Clicking void main every time you run (the simulator)

With regard to enjoyment, 14 participants said that they had enjoyed their programming experience while 3 were indifferent. One found the experience to be not enjoyable (see Figure 6).

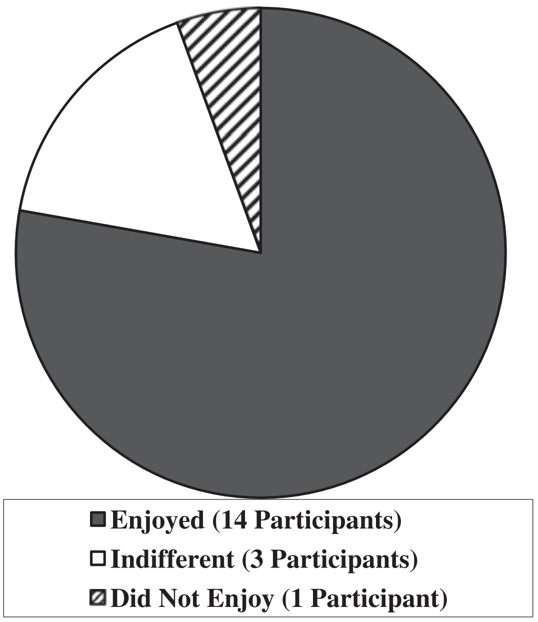


Figure 6. Students' enjoyment of their programming experience during the workshop.

Twelve participants reported completed tasks to be neither difficult nor easy, four easy and two difficult. In relation to the teaching of programming, three workshop elements were rated on a scale of 1–5 (see Table 7). Finally, differences between the pre- and post- responses of participants, with regards to whether they would consider programming in their spare time, are shown in Table 8.

### 4.3.3. *In-Workshop Exercise One*

Twenty participants (10 SW1, 10 SW2) completed In-Workshop Exercise One. This had a maximum score of 10. Figure 7 displays performance.

Table 7. Students' opinions on the effectiveness of elements of the workshop.

| | | Score breakdown (by no. of responses)[a] | | | | |
|---|---|---|---|---|---|---|
| Workshop component | Mean score (maximum of 5) | 1 | 2 | 3 | 4 | 5 |
| Kebot simulator | 4.39 | 0 | 0 | 0 | 11 | 7 |
| Programming support | 4.44 | 0 | 0 | 1 | 8 | 9 |
| Workshop presentation | 4.28 | 0 | 0 | 2 | 9 | 7 |

[a]1 (not at all effective) to 5 (extremely effective).

Table 8. Pre- and post-workshop comparison of students' opinions on programming.

"Would you consider learning to program in your spare time if you were given appropriate support?"

| | Yes | Not sure | No |
|---|---|---|---|
| Pre-workshop | 18 | 0 | 0 |
| Post-workshop | 14 | 4 | 0 |
| Change | −4 | +4 | 0 |

In SW1, the highest score awarded was 10 (one participant), the lowest was 4 (one participant), the mean score is 6.9/10 and the standard deviation is 1.79. In SW2, the highest score awarded was 10 (four participants), the lowest score awarded was 6 (one participant), the mean score is 9.0/10 and the standard deviation is 1.25.

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 participants were enrolled at different institutions, an attempt was made to use the *t*-test to determine whether there was a statistically significant difference in participants' overall scores. 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 groups overall.

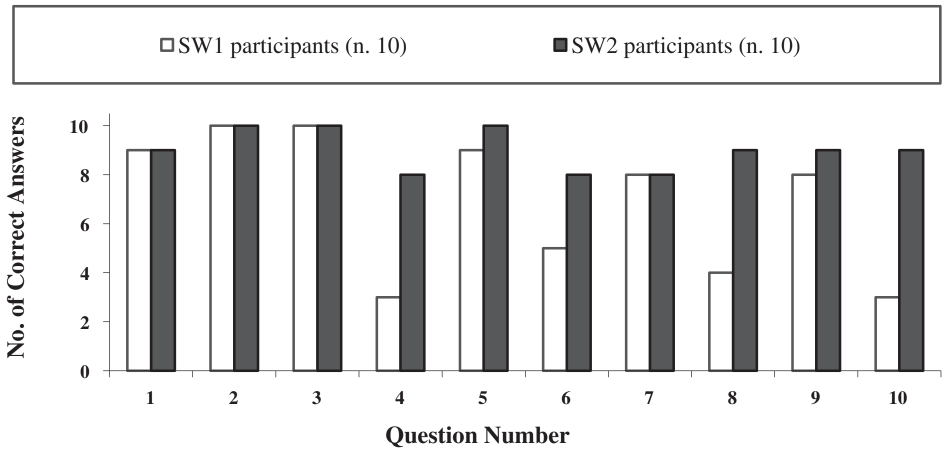


Figure 7. In-Workshop Test One performance.

### *4.3.4. In-Workshop Exercise Two*

Eighteen participants (10 SW1, 8 SW2) completed In-Workshop Exercise Two. Data collected from two participants who attended Day Two, but not Day One, were omitted. This test also had a maximum score of 10. Figure 8 displays performance. With regard to SW1: the highest score awarded was 9 (one participant), the lowest score awarded was 2 (two participants), the mean score is 5.6/10 and the standard deviation is 2.46. With regard to SW2: the highest score awarded was 9 (two participants), the lowest score awarded was 3 (one participant), the mean score is 6.75/10 and the standard deviation is 2.25.

Scores for both groups were normally distributed as assessed by the Shapiro–Wilks test ($p > .05$). There was also a homogeneity of variances, 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. 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 $t(16) = 1.02$, $p = .322$.

### *4.3.5. Post-workshop exercise*

Eighteen participants (10 SW1, 8 SW2) completed the post-workshop exercise. Table 9 displays a breakdown of scores.

Due to the groups' different backgrounds, statistical analysis was undertaken. Letter grades 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. No significant difference was found between the groups: Task One

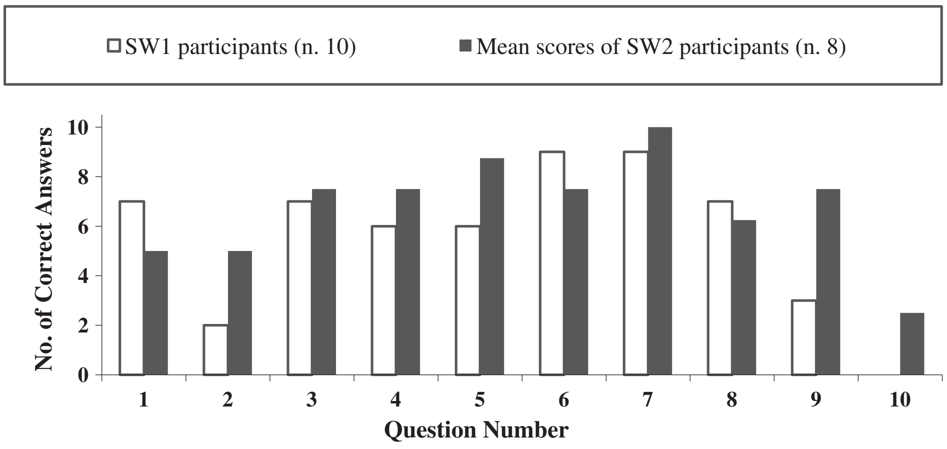


Figure 8. In-Workshop Test Two performance.

Table 9. Breakdown of SW1 and SW2 student groups performance on the post-workshop exercise.

| | Task 1 | | | Task 2 | | | Task 3 | | | Task 4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Participants | *a* | *b* | *c* | *a* | *b* | *c* | *a* | *b* | *c* | *a* | *b* | *c* |
| SW1 *(n. 10)* | 3 | 7 | 0 | 5 | 5 | 0 | 4 | 2 | 4 | 1 | 0 | 9 |
| SW2 *(n. 8)* | 4 | 3 | 1 | 5 | 3 | 0 | 5 | 2 | 1 | 0 | 5 | 3 |
| Total *(n. 18)* | 7 | 10 | 1 | 10 | 8 | 0 | 9 | 4 | 5 | 1 | 5 | 12 |
| | *39%* | *55%* | *6%* | *56%* | *44%* | *0%* | *50%* | *22%* | *28%* | *6%* | *28%* | *66%* |
| No self-taught | 4 | 8 | 1 | 6 | 7 | 0 | 6 | 3 | 4 | 0 | 2 | 11 |
| experience *(n. 13)* | *31%* | *61%* | *8%* | *46%* | *54%* | *0%* | *46%* | *23%* | *31%* | *0%* | *15%* | *85%* |

($U = 35.5$, $p = .696$), Task Two ($U = 35$, $p = .696$), Task Three ($U = 28$, $p = .315$), and Task Four ($U = 21.5$, $p = .101$). Figure 9 displays combined performance of participants.

*P* values 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, Wood, & Knight, 2008). Extreme *p* values (e.g. .0 or 1.0) restrict the reliability of test scores (Matlock-Hetzel, 1997) and may indicate that a question does not discriminate performance. As no statistical difference was found between SW1 and SW2 groups on the PWE, the performance of both groups was considered jointly. *P* values have been calculated based on the number of Grade A's awarded:

- Task One: a *p* value of .39 for SW1 and SW2 groups and .31 for participants with no self-taught experience.
- Task Two: a *p* value of .56 for SW1 and SW2 groups and .46 for participants with no self-taught experience.

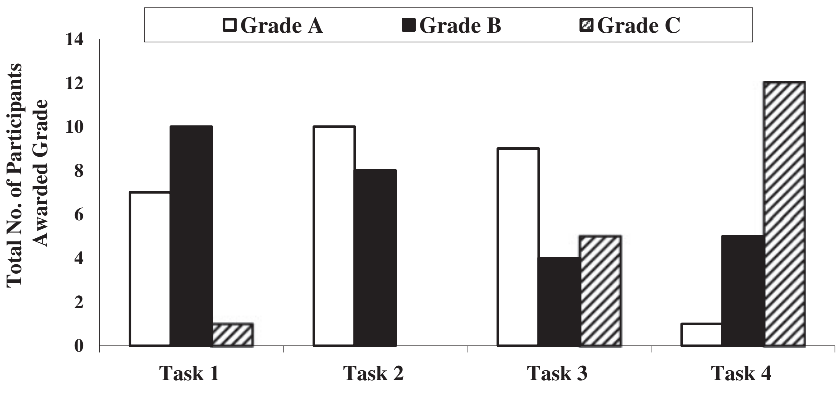


Figure 9. Combined performance of SW1 and SW2 participants on the PWE.

- Task Three: a *p* value of .50 for SW1 and SW2 groups and .46 for participants with no self-taught experience.
- Task Four: a *p* value of .06 for SW1 and SW2 groups and .0 for participants with no self-taught experience.

With the exception of Task Four, the *p* values reported all fall within these ranges which indicates that Tasks One to Three were not disproportionality easy and do discriminate performance. The low *p* value of Task Four, however, suggests that other factors may have impacted on participants' responses. See Section 5 for discussion of such issues.

#### *4.3.6. Teacher interviews*

Two main themes were identified during the thematic coding: (1) Robot simulator and (2) Student participants. A summary of interview data is provided below. Full interview transcripts have been made available separately (Major, 2014). Interviewees have been assigned codes T1, T2, and T3.

*4.3.6.1. Theme One – the robot simulator.* All three teachers believed Kebot to be an effective introductory programming teaching tool. The nature of the simulator, in particular, it's visual and simplistic nature, was highlighted:

> T1: The way it worked I think is a very good idea … they are not having to worry about the nitty gritty stuff, it just works … it's a much more effective way of doing it … they can see something happening as a result of what they are doing.

Only T2 identified a potential issue with Kebot:

> T2: … you have pre-written a lot of the methods it does make it look easier to them then it perhaps would be if they were starting from nothing …

There was a consensus that the simulator offered a more efficient means of introducing programming compared to traditional teaching methods:

> T1: If we were doing it the "traditional way" you would be talking about double (the amount of time), at least.

> 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 … with a medium to low ability group it probably makes a difference.

T1 identifies the "what if scenario" which may prevent some teachers from using educational software:

> T1: There's always the "what if scenario" … if it goes wrong, what happens? … the fact that you can just restart (the simulator) takes away some of the problem.

T3 describes how the simulator helps to break down anxieties which teachers may have about 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 it's done that and it is doing that.

T1 was of the opinion that in its current form Kebot 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 … The idea of (the robot simulator) is to be a tool to get kids thinking about designing and building something … I'd be happy to use (the simulator) in a classroom without further modification.

*4.3.6.2. Theme two – student participants.* T1 remarked that their students (SW2 participants) were capable, motivated, and had some prior elementary programming experience:

> T1: (They) would be the sort of top end, the interested ones. They have done VB (before). Around 15–16 hours in the first term.

Both T2 and T3 taught in the same high school at the time of the interview. Both agreed that their student cohort (SW1 participants) 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.

T1 believed that the robot simulator and workshop session were well received by participants:

> T1: I got a bit of flak from them saying, "Why can't you do it like this!."

T1 suggested that a robot simulator can help to improve a novice's perceptions of the subject, specifically because it enables students to picture the real-world applications of programming. T2 was less certain whether the simulator helped to improve students' perceptions of programming:

> 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.

With regard to whether students would be more encouraged when taught using the robot simulator compared to traditional teaching methods, T3 believed that the nature of the programming virtual robots was a strong positive:

> T3: 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.

All teachers offered views on how the workshop might be modified. The creation of supporting materials, in addition to the development of additional tasks, was also discussed.

## 5. Discussion

In Section 1, it was established that Kebot would be considered effective if three criteria were satisfied. Analysis of collected data indicates this was the case and that a robot simulator is an effective tool for supporting the learning of introductory programming. This conclusion must be considered with caution, however, as a number of factors should be taken into account when interpreting results.

### *5.1. Proposition one: a robot simulator is an effective tool for supporting the learning of introductory programming*

All *Case One* participants had programming knowledge. Questionnaire results demonstrate how many had spent a considerable amount of time learning programming prior to the workshop. The majority of trainees believe Kebot offers an effective and enjoyable means of introducing programming. No trainee responded that they would not use Kebot in their own future lessons. Aspects of Kebot 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).

An underlying assumption was that if the simulator motivated students, then this would offer some evidence of effectiveness. This is because increased enjoyment can enhance levels of learner effort, persistence, performance, and cognitive processing (Jerez, Bueno, Molina, Urda, & Franco, 2012; Ring, Giordan, & Ransbottom, 2008). *Case Two* post-workshop questionnaire results demonstrate how Kebot was believed to be an effective programming learning tool. No student stated the simulator was ineffective, only one did not enjoy their experience using it and the software scored high when rated on a five-point scale. The visual nature of Kebot and the fact that simulated robots are engaging were among attributes liked. This was in addition to the approach being accessible, interesting, and innovative. It should be noted that a slight negative change was observed between pre- and post-workshop responses when students were asked if they would consider learning programming in their own time. Based on the fact that that other qualitative data were positive, however, the authors do not consider this change substantial enough to suggest that the simulator had an adverse effect on perceptions.

Three exercises were used to assess performance. For the In-Workshop Exercises, it was decided that a mean score greater than 6 (out of 10) would indicate learning. Most participants performed well on the first exercise. A statistical difference between groups was, however, found with SW2 participants performing significantly better. Such participants may have adapted their previous, if limited, knowledge of VB to fit the tasks set during the early part of the workshop. For the second exercise scores were lower. This was expected due to increased complexity of concepts assessed. Unlike the first exercise, there was no statistically significant difference between groups.

With regard to the post-workshop exercises, for the reasons already outlined, it was decided beforehand that if around three-quarters of participants were awarded an A or B for a task, then this would provide evidence of learning. Performance on Tasks One, Two, and Three is judged to demonstrate deep learning due to this criteria being satisfied. The *p* values for these tasks all fall within an acceptable range and this indicates that the tasks were not disproportionately easy and do discriminate performance. Performance on Task Four, however, differed. This was also the most substantial challenge. Several 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 exercise. Analysis of code supports this view as only four participants made a meaningful attempt to solve the problem in full. The timing of the post-workshop exercises may be potentially responsible for performance on Task Four. This is because tasks were administered at the very end of the second workshop day when fatigue could have been an issue. It remains, however, unclear whether it was the design of the task, the nature of the task, or other factors that were responsible for performance observed.

As programming exercises were developed for the purposes of this work and reliability has not been independently verified, it is not possible to make strong claims about the simulator's effectiveness based on these alone. Bearing these cautionary points in mind, however, the programming exercises are considered to show:

- For the In-Workshop Exercises that the performance of students demonstrates learning.

- For Post-Workshop Tasks One, Two, and Three that the performance of students demonstrates how they were able to complete several programming challenges unassisted. Performance on Task Four, however, raises questions and caution should be used when drawing conclusions.

Three *interviews* were held with 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 as was the simplicity of the simulator. Kebot was believed to offer an enjoyable means of learning. As already discussed, this suggests that a simulator can be effective for supporting the learning of programming due to it motivating learning.

### *5.2. Proposition two: a robot simulator offers a more effective introduction to basic programming concepts when compared to participants' prior programming learning experience*

*Case One* participants had all learned programming beforehand. Asking trainees to compare their previous learning experience to the one using Kebot allows a "baseline" to be established. Only 3 of 22 trainees believed their previous introduction to programming was more effective than the one using Kebot and this is further indicative of effectiveness.

The SW2 group involved in *Case Two* had been introduced to aspects of VB, six months before the workshop. This had been in the "traditional" mould of learning programming (i.e. a text-based approach to general syntax, statements, etc.). This introduction to VB lasted for a similar amount of time as the Kebot workshop. Six SW2 participants believed their previous introduction to programming was less effective than the one using the simulator.

Teachers *interviewed* largely agreed that a simulator offers a more effective means of introducing the subject compared to more widely used approaches. How a robot simulator provides a "hook," which serves to entice novices, was responsible for these observations.

### *5.3. Consideration of 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 pre-dispositions or bias. Several rivals are considered below:

- *Null Hypothesis* (i.e. observations are the result of chance). *How addressed*: Workshops replicated. Multiple sources of evidence used.

- *Novelty of Simulator* (i.e. interest in the simulator is confused for learning and encourages participants to say they have learned). *How addressed*: By the scoring process which distinguishes deep and surface learning.
- Experimenter Expectation Effect (i.e. results are influenced by the experimenter's expectation that the simulator is effective). *How addressed*: By adhering to marking schedules and by subjecting exercises to second marking.
- "Good Subject Effect" (i.e. participants mark opinions in favour of the simulator to aid the project). *How addressed*: By participants identifying up to three aspects they liked/disliked as they are more likely to be truthful when identifying positive and negatives than answering "yes/no."
- Implementation Rival (i.e. workshop sessions and not the simulator accounts for results). *How addressed*: By asking participants to rate the effectiveness of the simulator and workshop. If more rated the workshop as effective this may have accounted for results.
- Practice Effects (i.e. when participants are exposed to repeated measures of similar data collection, their performance on second and subsequent tests may differ from what it would be otherwise). *How addressed*: By ensuring that all exercises were substantially different.

### *5.4. Recommendations to support the development and use of robot simulators*

Kebot is considered to be a "generic" simulator as it has no exclusive features. Viewed in its simplest form, the main features of Kebot allow:

- Agents to move forwards and backwards.
- Visual arena backgrounds to be loaded (or "drawn") onto the arena floor.
- 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.
- Environmental interaction through sensors that can detect the presence of 2D and 3D objects.

As the key features of Kebot do not extend beyond those outlined, the generalisability of the software is enhanced as it is considered that these would form a part of any comparable simulator. It was intentional that Kebot's functionality would remain neutral (as far as is possible) to counter suggestions that the results were not applicable to other simulators. Recommendations to support the development and use of robot simulators, in an introductory programming context, are presented in Table 10.

Table 10. Recommendations to support the future development and use of robot simulators in an introductory programming context.

| Recommendation | Description |
|---|---|
| Recommendation One: *Maintain a focus on programming* | Simulators can overcome issues with physical robots that distract learners (e.g. mechanical failure and problems with storage). Take advantage of these reduced issues by focusing on programming and not the simulator software itself |
| Recommendation Two: *Appreciate the importance of visualisation* | Data highlight the importance of visualisation when learning programming using a simulator. Visual feedback provided by a simulated environment should reflect users' programs and make it possible to trace code and on-screen activities. There should also be an indication when the simulation is running |
| Recommendation Three: *Encourage motivation through accessibility* | A simulator has been found to have a significant motivating appeal, and it is fair to assume that this would translate to an increased level of effort. A simulator should seek to encourage motivation by ensuring it remains accessible to new users |
| Recommendation Four: *Support the needs of diverse users* | To ensure users are not overwhelmed by the features of a simulator, nor left frustrated by a lack of pace, software (and associated tasks) should allow complexity to be gradually increased by learners to allow them to continually work within, but at the limits of, their ability |
| Recommendation Five: *Beware unnecessary complexity* | A simulator should not add an additional layer of complexity that distracts learners. GUIs should contain only frequently used features while the names of relevant control methods and variables should be identifiable and not abstract. Unnecessary code should be concealed |
| Recommendation Six: *Understand the needs of all learners involved* | There may be more than one group of learners using a simulator, especially if it is intended for use in high schools. Educators responsible for introducing the approach (in addition to technical support staff) may have little or no prior programming experience and should be viewed as learners themselves |

## 6. Conclusion and future work

The aim of this research was to investigate whether a robot simulator is an effective tool for supporting the learning of introductory programming. The work was carried out as the findings of a previously completed SR, and exploratory research, identified how such research was required. A robot simulator, named Kebot, was developed and used to run four 10-hour programming workshops. Kebot allows for real-time interaction with a simulated agent, the customisation of an agent’s environment using objects, coding in Java, and the creation of imaginative programming tasks. Kebot 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).

A multi-case case study was undertaken. Student programmers, in addition to pre- and in-service high school teachers, have taken part in empirical research. Effectiveness was determined after considering opinions, attitudes, and motivation in addition to an analysis of students’ programming performance. Pre- and post-workshop questionnaires, interviews, and programming assessments have been used. A simulator was judged as effective because:

(1) Participants enjoy learning programming in such a manner.
(2) Participants believe the approach to be valuable.
(3) Most evidence suggests that students successfully learnt introductory programming concepts as tasks were completed to a satisfactory standard (although several factors must be taken into account when interpreting the results of completed exercises).

How Kebot was highly regarded by almost all participants indicates how the approach appeals to people of both genders and of various experience and ages. It is believed that findings are applicable to other simulators that may be used. A set of recommendations to support the future development of other robot simulators have been provided.

The findings of this work are important as they help to enhance our knowledge about using robot simulators as tools to support the learning of 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 classes. 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 carried out.

### *6.1. Future work*

It is recommended that further work is undertaken to consider the application of Kebot in alternative learning settings and to support the teaching of alternative programming curricula. This would allow an opportunity to support findings reported or to contradict them. The development of alternative instruments, in particular the programming exercises completed by students, would allow for additional insights into how a simulator supports learning. This may also help to overcome questions related to performance on some of the programming tasks (in particular, the final task of the post-workshop exercise) and could provide further confirmatory evidence with regard to effectiveness. There also remains scope for an independent evaluation of Kebot and other existing robot simulators. See Marshall, Brereton, and Kitchenham (2014) for a recent example where several tools were evaluated to determine their suitability for purpose.

It is believed that there is potential to conduct a more specific investigation, as opposed to the exploratory one reported here. Indeed, it would be interesting to explore whether learners who have used a 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. 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 simulator go on to show greater aptitude during a more extensive programming course).

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, for both ethical reasons (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 ones (given that a more substantial workshop would have required the development of additional materials). It would be valuable to determine the effectiveness of using simulated robots to support the teaching of programming over a sustained period of time (such as one complete academic year) and to establish the affordances that such an approach provides. There is also significant scope to compare the effectiveness of physical and simulated robots, and a comparative study where these tools were used for the same programming tasks would make a strong contribution to existing knowledge. Potentially, if a negligible difference was found with regard to learner performance and motivation, then this would provide a compelling argument for the use of simulated robots in place of physical models.

## Acknowledgements

Louis Major would like to thank Marian Petre and Barbara Kitchenham for their advice and constructive comments. Gratitude is also owed to the staff and students whom made this study possible.

## Notes

1. http://www.junun.org/MarkIII (Accessed 2 August 2014).
2. http://www.bluej.org (Accessed 2 August 2014).
3. http://www.cyberbotics.com/ (Accessed 2 August 2014).
4. http://www.scm.keele.ac.uk/staff/l_major/files.php (Accessed 2 August 2014).
5. A precise figure could not be decided in advance because the exact number of participants was unknown, although it was intended that this would be in the range of 70–75% of participants.

## References

ACM/IEEE. (2008). *Computer science curriculum 2008: An interim revision of CS 2001*. Report from the Interim Review Task Force. Retrieved August 2, 2014, from http://www.acm.org/education/curricula/ComputerScience2008.pdf

Alavi, M. (1994). Computer-mediated collaborative learning: An empirical evaluation. *MIS Quarterly, 18*, 159–174.

Alimisis, D., Moro, M., Arlegui, J., Pina, A., Frangou, S., & Papanikolaou, K. (2007). Robotics & constructivism in education: The TERECoP project. *EuroLogo, 40*, 19–24).

ANSI Standards Committee on Dental Informatics. (2001). *Guidelines for the design of educational software*. Working Group Educational Software Systems.

Beale, R., & Sharples, M. (2002). *Design guide for developers of educational software*. British Educational Communications and Technology Agency. Retrieved August 2, 2014, from http://www.eee.bham.ac.uk/sharplem/Papers/Design%20Guide.pdf

Becker, B. W. (2001). Teaching CS1 with karel the robot in Java. *ACM SIGCSE Bulletin, 33*, 50–54.

Biggs, J. (2003). Aligning teaching and assessing to course objectives. *Teaching and Learning in Higher Education: New Trends and Innovations, 2*, 13–17.

Borge, R., Fjuk, A., & Groven, A. K. (2004). Using Karel J collaboratively to facilitate object-oriented learning. In C. L. Kinshuk, E. Sutinen, D. G. Sampson, I. Aedo, L. Uden, & E. Kähkönen (Eds.), *Proceedings of IEEE International Conference on Advanced Learning Technologies* (pp. 580–584). Joensuu: IEEE.

Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT Press.

Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). Using a protocol template for case study planning. In G. Visaggio, M. T. Baldassarre, S. Linkman, & M. Turner (Eds.), *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Italy: University of Bari.

Buck, D., & Stucki, D. J. (2001). JKarelRobot. *ACM SIGCSE Bulletin, 33*, 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. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 21*, 165–173.

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 A. Gottscheber & D. Obdržálek (Ed.), *Research and education in robotics-EUROBOT 2011* (pp. 78–87). Berlin: Springer.

Cowden, D., O'Neill, A., Opavsky, E., Ustek, D., & Walker, H. M. (2012). A C-based introductory course using robots. In L. S. King & D. R. Musicant (Eds.), *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 27–32). Raleigh, NC: ACM.

Dann, W., Cooper, S., & Pausch, R. (2006). *Learning to program with Alice*. Upper Saddle River, NJ: Prentice Hall.

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*.

Douglas, S., Farley, A., Lo, G., Proskurowski, A., & Young, M. (2010). Internationalization of computer science education. In G. Lewandowski & S. Wolfman (Eds.), *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 411–415). Milwaukee, WI: 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). London: Springer.

Edwards, S. H. (2003). Rethinking computer science education from a test-first perspective. In R. Crocker & G. L. Steele Jr (Eds.), *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (pp. 148–155). Anaheim, CA: ACM.

Enderle, S. (2009). Grape–graphical robot programming for beginners. In A. Gottscheber, S. Enderle, & D. Obdrzalek (Eds.), *Research and education in robotics – EUROBOT 2008* (pp. 180–192). Berlin: Springer.

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). Berlin: Springer.

Fagin, B. S., Merkle, L. D., & Eggers, T. W. (2001). Teaching computer science with robotics using Ada/Mindstorms 2.0. *ACM SIGAda Ada Letters, XXI*, 73–78.

Flot, J., Schunn, C., Liu, A., & Sharp, R. (2012, November/December). Learning how to program via robot simulation. *Robot Magazine*. ISSN: 1555-1016. 68-70.

Gibbert, M., & Ruigrok, W. (2010). The "what" and "'how" of case study rigor: Three strategies based on published work. *Organizational Research Methods, 13*, 710–737.

Gibbs, G. R. (2007). *Analyzing qualitative data (Book 6 of The SAGE qualitative research kit)*. London: Sage.

Goldweber, M., Congdon, C., Fagin, B., Hwang, D., & Klassner, F. (2001). The use of robots in the undergraduate curriculum. *ACM SIGCSE Bulletin, 33*, 404–405.

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*, 202–208.

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). Berlin: Springer.

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: PPIG.

Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? In R. Anderson, S. A. Fincher, & M. Guzdial (Eds.), *Proceedings of the Second International Workshop on Computing Education Research* (pp. 97–108). Canterbury: 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.

Ladd, B., & Harcourt, E. (2005). Student competitions and bots in an introductory programming course. *Journal of Computing Sciences in Colleges, 20*, 274–284.

Lahtinen, E., & Ahoniemi, T. (2009). Kick-start activation to novice programming – A visualization-based approach. *Electronic Notes in Theoretical Computer Science, 224*, 125–132.

Lee, A. S. (1989). A scientific methodology for MIS case studies. *MIS Quarterly, 13*, 33–50.

Lemone, K. A., & Ching, W. (1996). Easing into C++. *ACM SIGCSE Bulletin, 28*, 45–49.

Lister, R., & Leaney, J. (2003). Introductory programming, criterion-referencing, and bloom. *ACM SIGCSE Bulletin, 35*, 143–147.

Liu, A. S., Schunn, C. D., Flot, J., & Shoop, R. (2013). The role of physicality in rich programming environments. *Computer Science Education, 23*, 315–331.

Major, L. (2014, March). *An empirical investigation into the effectiveness of a robot simulator as a tool to support the learning of introductory programming* (PhD Thesis). Keele University. Retrieved from http://opac.keele.ac.uk/record=b1558010~S4

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

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

Major, L., Kyriacou, T., & Brereton, O. P. (2012b). Teaching novices programming using a robot simulator: Case study protocol. In Y. Jing (Ed.), *Proceedings of the 24th Psychology of Programming Interest Group PPIG 2012* (pp. 93–104). London: London Metropolitan University, PPIG.

Marshall, C., Brereton, P., & Kitchenham, B. (2014). Tools to support systematic reviews in software engineering: A feature analysis. In M. Shepperd, T. Hall, & I. Myrtveit (Eds.), *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (p. 13). London: ACM.

Martin, C., & Hughes, J. (2011, September 6–8). Robot dance: Edutainment or engaging learning. In *Proceedings of the 23rd Psychology of Programming Interest Group PPIG 2011*. York: PPIG.

Matlock-Hetzel, S. (1997). *Basic concepts in item and test analysis*. Texas A&M University.

McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., … Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*, 125–180.

McWhorter, W. I., & O'Connor, B. C. (2009). Do LEGO Mindstorms motivate students in CS1? *ACM SIGCSE Bulletin, 41*, 438–442.

Merriam, S. B. (1998). *Qualitative research and case study applications in education*. San Francisco, CA: Jossey-Bass.

Miliszewska, I., & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Informing Science: International Journal of an Emerging Transdiscipline, 4*, 277–289.

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.

Oppenheim, A. N. (2000). *Questionnaire design, interviewing and attitude measurement*. Continuum International Publishing Group.

Paliokas, I., Arapidis, C., & Mpimpitsos, M. (2011). Playlogo 3d: A 3d interactive video game for early programming education: Let logo be a game. In F. Liarokapis, A. Doulamis, & V. Vescoukis (Eds.), *Third International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)* (pp. 24–31). Athens: IEEE.

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*. Wiley.

Pattis, R. E., Roberts, J., & Stehlik, M. (1995). *Karel the robot: A gentle introduction to the art of programming* (2nd ed.). Wiley.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., … Devlin, M. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin, 39*, 204–223.

Piaget, J. (1967). *The child's conception of the world*. Routledge & Kegan Paul.

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*, 190–205.

Resnick, M. (1994). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. MIT Press.

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., … Eastmond, E. (2009). Scratch. *Communications of the ACM, 52*, 60–67.

Rieber, L. P. (1996). Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research and Development, 44*, 43–58.

Ring, B. A., Giordan, J., & Ransbottom, J. S. (2008). Problem solving through programming: Motivating the non-programmer. *Journal of Computing Sciences in Colleges, 23*, 61–67.

Robson, C. (2011). *Real world research* (3rd ed.). Wiley.

Runeson, P., Höst, M., & Rainer, A., & Regnell, B. (2012). *Case study research in software engineering*. Wiley.

Satratzemi, M., Xinogalos, S., & Dagdilelis, V. (2003). An environment for teaching object-oriented programming: ObjectKarel. In V. Devedzic, J. Spector, D. Sampson, & Kinshuk (Eds.), *Proceedings of IEEE Advanced Learning* (pp. 342–343). Athens: IEEE.

Siegel, S. (1957). Nonparametric statistics. *The American Statistician, 11*, 13–19.

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.

Smith, M. K., Wood, W. B., & Knight, J. K. (2008). The genetics concept assessment: A new concept inventory for gauging student understanding of genetics. *Cell Biology Education, 7*, 422–430.

Solin, P. (2013). *Learn how to think with Karel the Robot*. NCLAB Public Computing. FEMhub Inc.

Soule, T., & Heckendorn, R. B. (2011). COTSBots: Computationally powerful, low-cost robots for Computer Science curriculums. *Journal of Computing Sciences in Colleges, 27*, 180–187.

Squires, D., & Preece, J. (1999). Predicting quality in educational software. *Interacting with Computers, 11*, 467–483.

Stewart, L. G., & White, M. A. (1976). Teacher comments, letter grades, and student performance: What do we really know? *Journal of Educational Psychology, 68*, 488.

Talbot, H. (2000). wxPython, a GUI Toolkit. *Linux Journal, 74*.

Thota, N., & Whitfield, R. (2010). Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education, 20*, 103–127.

Varma, S. (2006). *Preliminary item statistics using point-biserial correlation and p-values* (Vol. 16). Morgan Hill, CA: Educational Data Systems.

Verner, J. M., Sampson, J., Tosic, V., Bakar, N. A. A., & Kitchenham, B. A. (2009). Guidelines for industrially-based multiple case studies in software engineering. In A. Flory & M. Collard (Eds.), *Third International Conference on Research Challenges in Information Science 2009* (pp. 313–324). Fes: IEEE.

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*, 71–76.

Yin, R. K. (2009). *Case study research: Design and methods* (4th ed.). Sage.