Computer Science Education

ISSN: 0899-3408 (Print) 1744-5175 (Online) Journal homepage: http://www.tandfonline.com/loi/ncse20

Mindstorms robots and the application of cognitive load theory in introductory programming

Raina Mason & Graham Cooper

To cite this article: Raina Mason & Graham Cooper (2013) Mindstorms robots and the application of cognitive load theory in introductory programming, Computer Science Education, 23:4, 296-314, DOI: 10.1080/08993408.2013.847152

To link to this article: http://dx.doi.org/10.1080/08993408.2013.847152

Published online: 17 Oct 2013.

Computer Science Education, 2013
Vol. 23, No. 4, 296–314, http://dx.doi.org/10.1080/08993408.2013.847152

Mindstorms robots and the application of cognitive load theory in introductory programming

Raina Mason* and Graham Cooper

Southern Cross Business School, Southern Cross University, Hogbin Drive, Coffs Harbour, NSW 2450, Australia

(*Received 28 May 2013; accepted 30 August 2013*)

This paper reports on a series of introductory programming workshops, initially targeting female high school students, which utilised Lego Mindstorms robots. Cognitive load theory (CLT) was applied to the instructional design of the workshops, and a controlled experiment was also conducted investigating aspects of the interface. Results indicated that a truncated interface led to better learning by novice programmers as measured by test performance by participants, as well as enhanced shifts in self-efficacy and lowered perception of difficulty. There was also a transfer effect to another programming environment (Alice). It is argued that the results indicate that for novice programmers, the mere presence on-screen of additional (redundant) entities acts as a form of tacit distraction, thus impeding learning. The utility of CLT to analyse, design and deliver aspects of computer programming environments and instructional materials is discussed.

Keywords: Mindstorms NXT; cognitive load theory; introductory programming; instructional design; robots

Introduction

This paper reports on a series of workshops utilising Lego Mindstorms robots that were delivered to junior high school students. The results indicate favourably on several measures, including raising knowledge of programming, raising self-efficacy in programming and lowering perceptions of task difficulty in programming. These workshops were initially designed and delivered to girls who had expressed an interest in CS/IT, but evolved to be delivered to an all-of-year population of girls and then both girls and boys, with similar results.

While the key *surface feature* of the workshops was the use of robots as a target and vehicle to execute the programmes that participants had constructed, the *deeper features* of the workshops lay in the application of

*Corresponding author. Email: raina.mason@scu.edu.au cognitive load theory (CLT) (Sweller, 1994) to the analysis, design and delivery of learning activities. It is argued that consideration of learners' cognitive processes with respect to their focus on, and acquisition of, concepts, tasks and procedures associated with programming was critical to the success of the workshops.

In essence, it is argued that the use of robots in and of themselves may both aid and hinder learning. Key to the effectiveness of robots in instructional settings will be how the use of robots impacts upon learners' cognitive resources and processing.

"IT Girls" program

Purpose of the program

The majority of people employed in western countries in the areas of Information Technology (IT) and Computer Science (CS) are males (Australian Computer Society, 2011). The gender bias within tertiary studies of IT and CS are similar, with relatively few females studying these areas at a university level (Anderson, 2009; Barker & Aspray, 2000; Farrell, 2007). The original motivation for the series of studies reported in the current paper lies in the objective of raising the participation rate of females studying CS/IT at the university level by initiating a series of "IT Girl" workshops.

Lynn, Raphael, Olefsky, and Bachen (2003) suggests that female students show increased interest and confidence in technology when they are exposed to computers in single-sex classes with supportive teaching. The IT Girls program was therefore designed to be suitable for female students who were novices to computer programming, by presenting to small female-only groups, with female-only instructors who would be available to help students who requested assistance.

There was also a need to ensure that the instructional materials and activities facilitated learning of fundamental programming concepts by novices. To this end, CLT was used to analyse, design and deliver instructional materials and activities. It is argued that these cognitively designed instructional materials resulted in the concepts and procedures to be relatively "accessible" for novices to learn compared with what is usually the case in more traditional approaches to teaching programming concepts through problem solving using conventional programming languages, often requiring a level of keyboard input (de Raadt, Watson, & Toleman, 2004).

Cognitive load theory

CLT emphasises the limitations of working memory in both capacity (Miller, 1956) and duration (Peterson & Peterson, 1959) as primary causes for learning to falter. If at any time during a learning transaction working memory resources are “overloaded”, then the information processing required for learning is effectively broken.

Information processing models of learning define expert performance through the acquisition of schemas, which are cognitive constructs that store hierarchically organised knowledge scaffolds of concepts and procedures (Chi, Glaser, & Rees, 1982), and their subsequent automation, which enables such concepts and procedures to be applied with very low levels of conscious attention (Kotovsky, Hayes, & Simon, 1985; Shiffrin & Schneider, 1977). The acquisition and automation of schemas frees working memory activities to focus on “novel” aspects of situations for activities such as problem solving (Sweller, 2005). From this perspective, the primary goal of instruction is to impart to learners appropriate schemas and then to facilitate their automation.

CLT identifies three distinct sources of cognitive load (Sweller, Ayres, & Kalyuga, 2011), which cumulatively place impositions upon the limited resources of human working memory.

Intrinsic cognitive load refers to the innate difficulty of the to-be-learnt content, due primarily not to how many separate elements of information need to be learnt, but the extent to which the individual elements need to be considered together in how they interrelate with one another through element interactivity (Tindall-Ford, Chandler, & Sweller, 1997). Computer programming requires very many aspects of concepts, constructs, language and syntax to be considered together (du Boulay, 1986) and so has very high levels of element interactivity and therefore is high in intrinsic cognitive load.

Extraneous cognitive load refers to the cognitive imposition placed upon learners due to the way in which to-be-learnt content is presented in form and activities that need to be interpreted and manipulated by a learner prior to actually dealing with the core concepts and the conceptual interrelations under consideration (Chandler & Sweller, 1991). In the domain of computer programming, a primary source of extraneous cognitive load that is open to being modified is the programming language and the associated development environment. Some programming languages use dedicated line-code entry, requiring programmers to be literally not just word perfect, but letter perfect, along with syntax perfect, in order to execute correctly. Such environments are extremely non-user-friendly for novices who are seeking to raise their conceptual understanding of, for example, how an algorithm is executing. A program that faults due to a syntax error, for example, is at best a hindrance for novice programmers, and at worst, potentially a destructive sidetrack to understanding that may act as confirmation to a learner of his or her inadequacies and ineptitude at programming (Rogerson & Scott, 2010).

Finally, there is *germane cognitive load* that refers to the cognitive resources that a learner brings to bear on learning the to-be-learnt content

(Sweller, 2010) which, as described above, reduces to being the acquisition of schemas and their subsequent automation. While a broad goal of learning environments is to raise the level of germane cognitive load, such a rise in germane cognitive load can only be achieved if there are sufficient cognitive resources available for the task. Consequently, freeing of cognitive resources by reducing extraneous cognitive load is often a necessary prerequisite condition for the effective raising of germane cognitive load.

There are several instructional techniques derived from CLT that have been demonstrated to be effective and efficient (Mayer & Moreno, 2003) which are briefly addressed below in the "IT Girls workshops" section describing the design of workshop materials.

Selection of programming environments

Crucial to the design of the workshops was identification and utilisation of "suitable" programming environments. The primary computer application selected was Lego Mindstorms NXT robots. This was chosen in part because of the perceived motivation that may accrue from such a physical teaching aid (Klassner & Anderson, 2003). The primary reason for the selection of Mindstorms robots, however, was due to the nature of the integrated development environment (IDE), which is simple in number (has few elements to be manipulated), visual (is icon based whereby the icons indicate their functionality, and these are used to build timeline-flowcharts of execution), and drag-and-drop in implementation (click and drag is used to construct the flow lines of execution). This application thus reduces extraneous cognitive load through several means, enabling the freeing of cognitive resources from aspects of language and syntax, which can then be applied to considering newly presented information regarding concepts and procedures.

The second application chosen was Alice. This has been designed for novice programmers to be appealing and "fun" (Kelleher & Pausch, 2007). More importantly, however, is the manner in which the development environment operates. Although specifically developed to teach introductory Object Oriented approaches to programming (Cooper, 2010; Kelleher et al., 2002) and requiring a basic level of understanding and manipulation of objects, classes, methods and properties, Alice also presents a relatively simple (limited number of elements) interface with a primarily icon-based drag-and-drop construction mechanisms, and a visual nature of feedback via a 3-D graphic world based program execution.

While the original motivation for the workshops was to raise the inclusivity of females in pursuing a CS or IT career, the applications and instructional materials, in and of themselves, were designed to facilitate learning of fundamental programming concepts. As such, these materials should be useful for both boys and girls in the learning of core computer programming concepts. The design of the workshops, their effects and their evolution towards deployment to a generalised cohort of students, are described below.

IT Girls workshops

For the IT Girls workshops to be effective as an intervention in raising the participation rates of females studying CS/IT, they needed to be delivered before students had committed themselves to subject selections that act as prerequisites to career paths. The subjects that school students select for their senior high school years (Years 11 and 12 in the context of NSW in Australia, where these studies were conducted) define the beginning of such prerequisite pathways, and so female school students currently in Years 9 and 10 were considered to be the most suitable target demographic.

The process began with school visits for career information sessions, delivered to up to 50 male and female students at a time. These sessions emphasised the wide variety of careers available in CS/IT, that CS/IT can be creative and that often CS/IT workers are situated in teams, rather than alone at a keyboard. All presenters talked about education pathways through college and university, and their personal experience with working in or studying CS/IT.

Lego Mindstorms NXT robots were taken to these careers information sessions to spark interest in the students, and to trigger conversations about the use of sensors, programming and other technology in everyday life. For example, the touch sensor sparked conversations about touch screens on smartphones and the careers associated with the development of smartphone software, and the ultrasonic sensor initiated conversations about reversing sensors in cars, and the technology that was used to design, build and maintain cars.

An invitation was provided to the female students to attend a follow-up “IT Girls” workshop at the local university campus, which would present further aspects of CS/IT and programming, including the programming of the Mindstorms robots.

Mindstorms robots

The Lego Mindstorms NXT kits enable students to build and program robots to execute simple programs that demonstrate the full range of primary programming constructs (sequence, selection and iteration). Each kit consists of a central controller “brick” with processing hardware and software, touch, sound, light and ultrasonic sensors and several servo motors coupled with “technics”-style Lego pieces and gears to enable building of several forms of robots. These robots can respond to input through the sensors and can show output by sound, visual display and movement. The robots are programmed using the Mindstorms NXT software, using a USB cable connection to a PC.

The Mindstorms NXT programming environment is highly visual. Programs are created by dragging programming "blocks" – presented as visual icons – to a timeline, and then setting properties of each block by typing in values or selecting options. The programs are executed in sequence along the timeline that can have up to three concurrent branches. More complex programs can be constructed by using loop and switch/decision structures, as well as "wait" blocks that can be likened to event handling in more mainstream languages.

The workshops

The materials for the workshop were presented in the form of a number of small publicly demonstrated worked examples in accordance with CLT application to instructional design (Cooper & Sweller, 1987; Moreno, Reisslein, & Delgoda, 2006; Sweller & Cooper, 1985; Zhu & Simon, 1987), coupled with "free time" where students could create their own programs.

In each case, the public presentation of a worked example was followed by the students attempting to replicate the process just demonstrated, with assistance provided to any student who requested it. After the entire cohort had completed the task in question, the next task would be demonstrated publicly as a worked example, with the students then attempting it. The cyclic process of public demonstration of a worked example followed by students' replication was continued until all the workshop tasks had been completed. In broad terms, this process of public demonstration began with the simplest of programming tasks and moved to those of increasing complexity.

CLT informed the design and delivery of the workshops, because there are several empirically demonstrated benefits that accrue in its use in complex content domains. Worked examples have been proven to be a more effective teaching approach than problem solving for novices in several technical domains (Cooper & Sweller, 1987; Sweller et al., 2011).

The instructor's presentation used verbal explanations at the same time as pictures and processes were demonstrated on the screen. Printed materials were not given to the participants, and instead, only the verbal explanations were given, in accordance with the modality principle (Mousavi, Low, & Sweller, 1995). This states that students will learn better (as measured by both retention and transfer) from graphics combined with narration (Tindall-Ford et al., 1997) and simple animations combined with narration (Mayer & Moreno, 1998) (both techniques using visual and auditory channels) than from graphics combined with printed text (using only the visual channel).

The split attention principle posits that students will learn better when the words and pictures are physically and temporally integrated (Chandler &

Sweller, 1991; Mayer & Anderson, 1991), rather than "split" in either time or place. Therefore, the explanations of each step were provided at the same time as the on-screen demonstrations.

In addition, although printed materials were not given to the girls, the instructor followed a predefined script designed using the segmentation principle (Mayer & Chandler, 2001), which advocates delivering content in small learner-paced segments of delivery, moving from simple examples to more complex ones.

Each student attended a Mindstorms workshop and an Alice workshop. The Alice workshops are not detailed here due to space limitations, but the principles of design, demonstration and delivery follow the same principles as used in the Mindstorms workshops, which are described below.

Mindstorms workshops

In the Mindstorms workshops, students worked either alone or in pairs at one computer with one shared robot (of the three available robots), connected to the computer with a USB cable. The students were initially introduced to the Mindstorms robots in "humanoid form", shown how to handle the robot and identify the motors and sensors. After familiarising themselves with the physical robots, students were introduced to the Mindstorms NXT software and led through a series of programming activities by the instructor.

It was decided that due to the limited time available for each workshop, the concepts of sequence, looping and event handling (using the sensors) would be covered but that decision structures would be omitted. A sequence of activities was devised that would lead the students through using the programming environment, simple block use and setting of properties, sequence, looping, events (sensor triggers) and more complex combinations of these concepts.

For example, the first workshop activity – equivalent to the ubiquitous "Hello World" first program – involved dragging a sound block to the timeline and then choosing a verbal phrase through the property panel. This first program was then saved and downloaded into the robot using the control block and run using the robot's control buttons. The purpose of the first program was to introduce the use of the programming blocks and show how to compile and download the program to the robot.

After completing the set of predefined activities, students were then given free time to complete their own programs. They were encouraged to use their mobile phones to trigger the sound sensor and to explore how the various sensors, programming blocks and combinations worked.

The workshop days began and concluded with a questionnaire, collecting information about age, level of computer knowledge and prior programming experience, and participants' perceptions of their interest and confidence in programming, the perceived difficulty of programming the Mindstorms robots and perceived difficulty of programming in general. The results of the first two of these workshop days are reported below.

Results

The participants ranged in age from 14 to 16 years with the majority being 15 years old (n = 19, M = 15.0, s = .67). All felt that they had an average level of computer knowledge (n = 19, M = 4.8, s = 1.42 on a 9-point Likert scale). Only three students indicated that they had any prior programming experience. Of these, two reported that they had "average" programming experience, and one reported "very low" programming experience. Mean for the group was 1.6 (n = 19, s = 1.35) on a 9-point Likert scale where 1 = "no experience" and 9 = "expert".

There was no statistically significant difference in the attitudes of students towards interest in programming of the Mindstorms robots evident from the pre- and post-workshop questionnaire responses [Wilcoxon Signed Rank test: $W = -18$, $N_{s/r} = 15$, $z = -.5$, $p = .3085$]. This was not an unexpected result given that participants had self-selected to attend the workshops and their initial interest was very high (median of 7). This may represent a ceiling effect.

There was, however, a significant difference in the perceived difficulty of programming the robots [Wilcoxon Signed Rank test: $W = -171$, $N_{s/r} = 18$, $z = -3.71$, $p = .0001$] in the direction of "easier". If respondents had not had any prior experience with robotics programming (as participants had generally indicated), then it would be expected that it may have initially been perceived as a reasonably difficult task (median = 6 on a 9-point Likert scale). On completion of the workshop, the median for difficulty of programming robots had fallen to 3.5; therefore, the aim of the workshop to "demystify" some of the misconceptions surrounding programming has been met.

The mean of responses to the difficulty of programming the Mindstorms robots ($M = 3$, $s = 1.52$) in the post-workshop questionnaire showed that the participants found it generally easy to do.

The participants were asked to indicate their level of confidence with programming by agreeing or disagreeing with the statement "I feel confident with programming" (1 = strongly disagree, 9 = strongly agree). The girls' confidence in programming showed significant improvement after completing the workshop [Wilcoxon Signed Rank test, $W = 178$, $N_{s/r} = 19$, $z = 3.57$, $p < .001$].

Of the 19 participants who completed both questionnaires, a total of 17 shifted their measure on the Likert scale regarding the perceived level of difficulty with respect to programming towards the direction of "easy". This alone speaks for the success of the workshops in presenting an intervention that modifies girls' perceptions of programming in a positive manner. The results of these two workshop days are typical of the results obtained throughout the IT Girls program.

Non-self-selected girls

In the IT Girls Days for the girls of regional High Schools, the girls who participated were self-selected, so were likely to have been eager to learn about programming and IT. This self-selection may have influenced positively upon the effects of the workshop. In order to eliminate a self-selection effect, the workshops were then repeated in the context of a geographically isolated school, within the school grounds, and presented to all female students – not merely those who self-selected. This was done with specific focus on:

- enabling an outreach program for geographically isolated high schools, and
- testing whether the positive outcomes reported by the local self-selected students would be evident among a wider population of female students.

For logistics reasons, students had less time (40-min workshops compared with 1 h) in the Mindstorms workshops than experienced by participants in the local IT Girls workshops. Students worked by themselves with a Mindstorms robot and a computer, and the author demonstrated each small worked example and then each student completed the activity with her robot. As in previous workshops, the activities built from simple activities involving the placement of one programming block and setting of one property, to complex programs with loops, events and longer sequences. Students were given a short time at the end of the instructional period to further explore the programming blocks and to create their own programs.

The remote school workshops were delivered across a set of three days to all female students in the high school, excluding Year 12 girls who were completing final exams. On the first IT Girls Day (Years 10 and 11), 17 girls participated. The second IT Girls Day at the school included participants from Years 8 and 9, and one girl from Year 10. The age range of this cohort was from 13 to 16 years, with the mean age 14.3 (standard deviation .7). On the third IT Girls Day, 16 girls from Years 6 and 7 participated.

As with the local workshops pre- and post-workshop questionnaires were presented to collect responses through Likert scales regarding career aspirations, knowledge of IT, confidence in programming and perceived difficulty

Table 1. Medians of 9-point Likert response questions.

	Career aspirations		Knowledge of IT careers		Confidence in programming		Difficulty of programming	
	Pre	Post	Pre	Post	Pre	Post	Pre	Post
Day 1 year 10/11	1.5	5.5	2.5	6	1	6	6.5	3
	Significant		Significant		Significant		Significant	
Day 2 year 8/9	1.5	5.5	2	5.5	2.5	5	4.5	5.5
	Significant		Significant		Significant		not Significant	
Day 3 year 6/7	3.5	4.5	1	5.5	3.5	9	3.5	1.5
	Significant		Significant		Significant		not Significant	

of programming. Summary results (median scores from 9-point Likert scales) are presented in Table 1.

Overall, there were again positive results indicated from the workshops, with gains in all three groups in both knowledge of IT and career aspiration in IT. All three groups also reported significant rise in confidence in programming, and one group reported a significant decline in the perceived difficulty of programming. Despite these workshops being delivered to all-of-year cohorts, the workshops again displayed positive impacts on participants' self-efficacy in programming.

Controlled experiment

The IT Girls Days and the remote workshops demonstrated positive effects on the students who participated. It was unclear whether these positive results were due to the use of robots, or instructional design, or single-sex format, or student empathy with the instructors (as the instructors were the same gender as participants) or some interaction between these factors. It has been described above how the instructional materials had been designed to raise the effectiveness of learning the introductory programming concepts, and how CLT had been used to inform the instructional design.

It was determined to undertake a controlled study that specifically looked at the effects of the interface based upon CLT. This experiment was conducted with the intent of testing whether cognitive load aspects of design were indeed likely contributors to the success of the instructional materials. To demonstrate the utility of the instruction materials for both genders, males and females were included in this study.

Methodology

This experiment was conducted in a private high school in regional Queensland, Australia, with 32 students in Year 7, aged 11–13 years.

Students completed an introductory Mindstorms programming workshop, followed by a Mindstorms test and then attended a normal school period of classwork. After lunch they attended a CS/IT careers information session followed by an Alice programming workshop. Each student was asked about their knowledge and attitudes towards CS/IT and programming before the IT careers day and at the end of the day, after the Alice workshop. Some demographic information was also collected about each student: age and school year, their level of computer literacy and, programming experience.

Two Mindstorms interfaces were used, both of which are native to the Mindstorms NXT environment. A "Subset" version of the interface, as employed in the previous IT Girls workshops, presented a truncated set of icon blocks suitable to be used for the most common programming tasks. A more "Complete" version of the interface presented a greater number of icon blocks, including the blocks available in the Subset interface, organised into a menu/submenu format. The buttons were the same size in each interface and buttons that had the same functionality had the same appearance.

CLT has previously demonstrated that redundant information may interfere with learning (Mayer, Heiser, & Lonn, 2001). The Complete interface presented icon blocks that were unnecessary to activities undertaken by the students. These additional icon blocks were never referenced in any instructions or activities. It was hypothesised that their mere presence on screen would act as a form of tacit distractor.

It was hypothesised that, as with the IT Girls workshops, students given either interface would experience an increased knowledge of IT, increased intent to pursue IT as a career, decreased perceived difficulty of programming and increased self-efficacy in programming. It was hypothesised that the students presented with the Subset version of the interface would report amplified effects, resulting in higher self-efficacy and lower levels of the perceived difficulty of programming, than those presented with the more Complete interface. It was hypothesised that students given the Subset interface would outperform those given the Complete interface on subsequent knowledge tests (measured by time taken, and score achieved).

It was also hypothesised that students given the Subset interface would be more able to transfer their newly acquired (general) knowledge and skills in computer programming to the Alice computer programming environment.

Treatment groups

Students were divided into two groups/workshops with roughly equal numbers of each gender (Group Subset: 9 males, 8 females; Group Complete: 7 males, 8 females) in each group. Groups were also balanced on student age and abilities, as determined by the teachers. Further details are available below in Results section "Homogeneity of groups". Group Subset used the Subset interface throughout their workshop. Group Complete used the Complete interface throughout their workshop. Both groups used the same programming blocks to complete the same activities and were given the same amount of time. Even though Group Complete had a greater number of blocks available via the palette on-screen, they were not directed to use any of these extra blocks, in any of the worked example activities or free programming time.

Test instrument and post-workshop questionnaire

After completing the Mindstorms workshop, the students were given a timed, written test designed to test recall of the purpose of various programming blocks, the building of schema about the interface used, near transfer of programming construct concepts as well as far transfer of concepts such as sequence, looping and events.

After the Alice workshops, at the conclusion of the day, students were given a post-workshop questionnaire to ascertain their perceived level of knowledge of IT, intention to pursue a career in CS/IT, perception of the difficulty of programming (generally and with both environments) and confidence with programming. Students were also asked about their interest in programming with Mindstorms and with Alice, and how difficult they found each of the programming workshops.

Results

Homogeneity of groups

Before the workshop, there was no significant difference between Group Subset and Group Complete in participant age, computer literacy, programming experience, knowledge of IT and intent to pursue a career in CS/IT. A t-test on age returned no significant difference ($\text{Mean}_{\text{Subset}} - \text{Mean}_{\text{Complete}} = -.08$; $t = -.3$; df = 24; $p = .77$). Mann–Whitney U-tests on computing skill, programming experience, knowledge of IT and intent to pursue a career in CS/IT all returned no significant difference between the two groups at the .05 level (Table 2).

Table 2. Homogeneity of groups.

	U_A	z	p
Computer literacy	87	.1	.46
Prog. experience	97	−.62	.27
IT knowledge	79	.26	.40
Career intent	83.5	.03	.49

Table 3. Results of IT knowledge/career intent.

	n	Median	Mode	Min	Max
Knowledge – pre	24	5	5	1	7
Knowledge – post	24	6	7	2	9
Career – pre	24	4.5	1	1	9
Career – post	24	5.5	7	1	9

Career aspirations and IT knowledge

Although there were 17 participants in Group Subset and 15 participants in Group Complete for the Mindstorms workshops and test, due to timetable clashes, not all participants completed both the pre- and post-workshop questionnaires. Twelve participants in each group completed both questionnaires. The median, minimum and maximum scores for pre-and post-workshop questions are given in Table 3. Results were analysed using the Wilcoxon Signed Ranks test.

There was a significant difference in the perceived IT knowledge of all participants after the workshop than before the workshop [Wilcoxon Signed Rank test: n = 24, W = 179, $N_{s/r}$ = 21, z = 3.1, p = .001] and a significant difference in participants' intent to consider a career in CS/IT after the workshop than before the workshop [Wilcoxon Signed Rank test: n = 24, W = 128, $N_{s/r}$ = 19, z = 2.57, p = .005].

Programming difficulty and self-efficacy

A significant decrease was obtained in participants' perception of the difficulty of programming, from before the workshops to after the workshops [Wilcoxon Signed Rank test: n = 24, W = 128, $N_{s/r}$ = 19, z = 2.57, p = .005]. The median, minimum and maximum scores for perceived level of difficulty of programming are shown in Table 4.

Table 4. Measures of difficulty of programming.

	n	Median	Mode	Min	Max
Pre-workshop	24	5	5	1	9
Post-workshop	24	4	5	1	7

Table 5. Difficulty of programming – groups.

	n	W	$N_{s/r}$	z	p
Subset	12	−55	11	−2.42	.008
Complete	12	−29	10	−1.45	.07

Table 6. Measures of self-efficacy.

	n	Median	Mode	Min	Max
Pre-workshop	24	5	5	1	9
Post-workshop	24	7	9	2	9

It was hypothesised that Group Subset would have a greater shift in perception of difficulty in the direction of "easier" than Group Complete. The pre- and post-workshop answers from the two groups were then analysed individually using Wilcoxon Signed Rank tests. The results are shown in Table 5.

Group Subset did have a significantly different perception of the difficulty of programming after the workshop compared with before the workshop, in the direction of "easier". Although displaying shifts downwards, Group Complete did not experience a significant change in their perception of the difficulty of programming.

In the pre- and post-workshop questionnaires, students were asked to indicate their confidence with programming on a 9-point Likert scale (where 1 = not confident at all and 9 = extremely confident). Although it was hypothesised that both groups would experience an increase in self-efficacy in programming, it was anticipated that Group Subset may have a greater increase than Group Complete.

Answers from both groups were analysed together using Wilcoxon Signed Rank tests, and participant's self-efficacy was found to be significantly higher after the workshops than before the workshops [$n = 24$, $W = 119$, $N_{s/r} = 18$, $z = 2.58$, $p = .005$]. The median, minimum and maximum scores for participants' self-efficacy before and after the workshops are shown in Table 6.

Pre- and post-workshop measures were then analysed separately for the Subset and Complete groups using Wilcoxon Signed Rank tests and the results are shown in Table 7.

Group Subset had a significantly higher self-efficacy in programming after the workshop than before the workshop. Although Group Complete displayed some shifts upwards in self-efficacy, this group did not experience a significant change.

Table 7. Self-efficacy in programming – groups.

	n	W	$N_{s/r}$	p
Subset	12	42	9	$p < .01$
Complete	12	18	9	$p > .05$

Table 8. Test scores.

	Mean	Std. dev	Min	Max
Subset	11.03	2.60	6.5	15
Complete	8.83	1.88	6	14

Test performance

Test completion time. It was hypothesised that if the different interface had an effect on learning then participants from Group Subset would take less time to complete the test than participants in Group Complete. The times from the 17 participants in Group Subset and 15 participants in Group Complete were compared using a *t*-test. Participants from Group Subset were found to take significantly less time to complete the test than Group Complete [$\text{Mean}_{\text{Subset}} - \text{Mean}_{\text{Complete}} = -150$ s; $t = -1.91$, df = 30, $p = .03$].

Test score. There was a maximum possible mark of 17 for the Mindstorms test, allocated between correct description of the purpose of programming blocks (3 marks), correct placements of programming blocks on a blanked interface (6 marks), correct multiple choice answers (near transfer – 5 marks) and correct far transfer answers (3 marks). The results for the two groups for total score, standard deviation, minimum and maximum marks are shown in Table 8.

The test scores from the 17 participants in Group Subset and 15 participants in Group Complete were compared using a t-test, and participants from Group Subset were found to have a significantly higher test score than Group Complete [$\text{Mean}_{\text{Subset}} - \text{Mean}_{\text{Complete}} = 2.20$; t = 2.71, df = 30, $p = .006$].

It is important to note that Group Subset was both faster and higher scoring in the test than Group Complete, thus removing the possibility that their increased speed in completing the test was due to them "skipping" questions that they did not attempt. Their performance is suggestive of having acquired better schemas for the Mindstorms programming environment.

Gender differences. Throughout the program, all participants received instruction from female instructors although single-sex Mindstorms workshops were conducted for both groups. Recall that there was a significant difference between the test scores for Group Subset and Group Complete. The instructional design of the workshops was based on CLT principles, and so if the beneficial effects of the workshop (originally designed for females) were due primarily to the instructional design – rather than merely the use of female instructors – then both females and males should experience a performance difference in the Mindstorms test. Alternatively, if the beneficial effects are a result of the female-only instructor aspects of the workshops, there will be performance differences between males and females. In effect, there are 4 groups, arranged in a 2 × 2 structure – Subset/Complete × Male/Female.

The test results from these groups were analysed using 2 × 2 ANOVA, and there was found to be no significant difference between male and female test scores ($F = .1$, df = 1, $p = .7542$), and no significant interaction ($F = 2.47$, df = 1, $p = .1273$). The beneficial effects of the workshop were due to the instructional design, rather than the gender of the instructors.

Mindstorms environment difficulty

It was anticipated that if having the extra (unused) programming blocks available in the Complete interface were having an effect on working memory load while using the interface, then Group Complete would perceive programming Mindstorms robots as more difficult than Group Subset, even though both groups completed the same activities with the same programming blocks.

The responses of the participants who completed the Mindstorms workshops and completed the post-workshop questionnaire were analysed for differences using a Mann–Whitney U-test. Participants in Group Complete found programming with the Mindstorms robots significantly more difficult than participants in Group Subset [$U_A = 144$, $z = 1.66$, $p = .049$]. The median, minimum and maximum scores are given in Table 9.

This was a hypothesised result. It is argued that the presence of the extra (unused) programming blocks was interfering with the attentional process for the students in Group Complete.

Alice environment difficulty

Group Subset and Group Complete were mixed in a common Alice programming session in the afternoon, after completing the Mindstorms workshops. Both groups completed the same Alice workshop activities, with the same Alice interface and were asked afterwards about the difficulty of programming in Alice (9-point Likert scale where 1 = really easy and 9 = really difficult). As all participants were in the same Alice programming workshop, the only variable was the Mindstorms group in which each participant had participated.

Table 9. Mindstorms difficulty.

	n	Median	Mode	Min	Max
Group Subset	15	2	1	1	5
Group Complete	14	3.5	1	1	9

Table 10. Alice difficulty.

	n	Median	Mode	Min	Max
Group Subset	15	1.5	1	1	7
Group Complete	14	5	5	1	9

It was hypothesised that the difference in difficulty experienced in the Mindstorms workshops as a result of having the subset interface for participants in Group Subset, compared with those in Group Complete, would have a positive transfer effect to the perceived difficulty of Alice programming. That is, participants that had experienced the Subset interface for the Mindstorms workshop, would perceive programming in Alice as less difficult than those who had experienced the Complete Mindstorms Interface because of their heightened level of transfer of programming concepts from Mindstorms to Alice.

The participant responses were analysed for differences using a Mann–Whitney U-test. The novice programming participants in Group Complete found programming with *Alice* significantly more difficult than novice programming participants in Group Subset [$U_A = 158$, $z = 2.29$, $p = .01$]. The median, minimum and maximum scores are given in Table 10.

This result shows a transfer effect from the Mindstorms workshop to the Alice workshop based upon the nature of the Mindstorms interface that participants had received.

Conclusion

Many of the robots used in educational settings have the capacity to take on humanoid form and thus be used at a simplistic level as a doll. These robots, however, also have the capacity to be deployed as a powerful teaching aid, interacted with and used at a far deeper level to be programmed to do realistic human-based tasks such as walking, dancing and talking.

While part of the appeal and perhaps too, even the beneficial effect, of robots in educational settings may lie in their form and perception as "toys" to "play with", it is overly simplistic to argue that this is the primary reason for their effectiveness as a teaching aid for computer programming. Computer programming is a complex problem-solving domain requiring sufficient knowledge and skills in a range of concepts (such as sequencing, selection, iteration, variables, data and functions) that generally need to be integrated into a coherent algorithm that must then be encoded into a specific programming language.

The truncated (simple-form) development environment of the Lego Mindstorms NXT robots provides an interface for program construction that relieves many of the extraneous aspects of language and syntax from the learner. It is argued that this is a primary source of the power of robots to aid learning in the context of the studies reported here. By reducing, and in some cases removing, many of the extraneous sources of cognitive load from the learner, the robots and their development environment, can be used as both a motivator and a conduit to focus attention and cognitive resources upon germane aspects of cognitive load required for schema acquisition and automation, which are the fundamental constructs of expertise. The design and delivery of instructional activities using these robots had also been engineered to be in accordance with cognitively based instructional design principles.

While robots provide a physical object that can be handled easily by students, can be presented in humanoid form that students may identify with, react both instantly and physically to provide meaningful feedback on the execution of their installed program, and can be compared with robots programmed by other students in a form of competitive contrast, a robot that needs to be programmed using a difficult, convoluted, unforgiving line code environment is unlikely to be useful as a pedagogical agent for novice programmers. A primary utility of robots, for novices, is in aspects of their development environments that reduce extraneous cognitive load, which may then be utilised for germane applications. The facilitation of learning core programming concepts and constructs in these "simplified" environments may be subsequently transferred to other programming environments.

References

Anderson, N. (2009). *Equity and information communication technology (ICT) in education*. New York, NY: Peter Lang.

Australian Computer Society. (2011). *Australian ICT statistical compendium 2011*. Retrieved from http://www.acs.org.au/2011compendium/

Barker, L., & Aspray, W. (2000). The state of research on girls and IT. In J. M. Cohoon & W. Aspray (Eds.), *Women and information technology: Research on the reasons for under-representation* (pp. 3–54). Cambridge, MA: MIT Press.

Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and Instruction*, *8*, 293–332.

Chi, M., Glaser, R., & Rees, E. (1982). Expertise in problem solving. In *Advances in the psychology of human intelligence* (Vol. 1, pp. 7–75). Hillsdale, NJ: Erlbaum. Retrieved from http://www.public.asu.edu/~mtchi/papers/ChiGlaserRees.pdf

Cooper, S. (2010). The design of alice. *ACM Transactions on Computing Education*, *10*(4), 1–16. doi:10.1145/1868358.1868362

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology*, *79*, 347–362. doi:10.1037/0022-0663.79.4.347

de Raadt, M., Watson, R., & Toleman, M. (2004). Introductory programming: What's happening today and will there be any students to teach tomorrow? In R. Lister & A. Young (Eds.), *ACE'04 Proceedings of the sixth conference on Australasian computing education* (Vol. 30, pp. 277–282). Darlinghurst: Australian Computer Society.

du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*, 57–73.

Farrell, M. (2007). *What are students' attitudes toward technology?* Presented at the Canada's Association of Information Technology Professionals. Retrieved from http://www.cips.ca/?q=webcasts-http://www.cips.ca/?q=systems/files/MicrosoftCIPSPresentation2007Live.ppt

Kelleher, C., Cosgrove, D., Culyba, D., Forlines, C., Pratt, J., & Pausch, R. (2002). Alice2: Programming without syntax errors. Presented at the 2002 Conference on User Interface Software and Technology. Carnegie Mellon University. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4640&rep=rep1&type=pdf

Kelleher, C., & Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM*, *50*, 58–64. doi:10.1145/1272516.1272540

Klassner, F., & Anderson, S. D. (2003). LEGO mindstorms: Not just for K-12 anymore. *IEEE Robotics & Automation Magazine*, *10*, 12–18. doi:10.1109/MRA.2003.1213611

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from Tower of Hanoi. *Cognitive Psychology, 17*, 248–294.

Lynn, K.-M., Raphael, C., Olefsky, K., & Bachen, C. M. (2003). Bridging the gender gap in computing: An integrative approach to content design for girls. *Journal of Educational Computing Research*, *28*, 143–162. doi:10.2190/79HP-RVE7-3A9N-FV8C

Mayer, R. E., & Anderson, R. B. (1991). Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of Educational Psychology*, *83*, 484–490.

Mayer, R. E., & Chandler, P. (2001). When learning is just a click away: Does simple user interaction foster deeper understanding of multimedia messages? *Journal of Educational Psychology*, *93*, 390–397.

Mayer, R. E., Heiser, J., & Lonn, S. (2001). Cognitive constraints on multimedia learning: When presenting more material results in less understanding. *Journal of Educational Psychology*, *93*, 187–198.

Mayer, R. E., & Moreno, R. (1998). A cognitive theory of multimedia learning: Implications for design principles. Retrieved from http://www.unm.edu/~moreno/PDFS/chi.pdf

Mayer, R. E., & Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. *Educational Psychologist*, *38*, 43–52.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, *63*, 81–97. doi:10.1037/h0043158

Moreno, R., Reisslein, M., & Delgoda, G. (2006). Toward a fundamental understanding of worked example instruction: Impact of means-ends practice, backward/forward fading, and adaptivity. In *Proceedings of the 36th Annual Frontiers in Education Conference* (pp. 5–10). San Diego, CA: IEEE. Retrieved from http://fie-conference.org/fie2006/

Mousavi, S. Y., Low, R., & Sweller, J. (1995). Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology*, *87*, 319–334. doi:10.1037/0022-0663.87.2.319

Peterson, L., & Peterson, M. J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology*, *58*, 193–198. doi:10.1037/h0049234

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education*, *9*, 147–171.

Shiffrin, R., & Schneider, W. (1977). Controlled and automatic human information processing II. Perceptual learning, automatic attending and a general theory. *Psychological Review*, *84*, 127–190.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, *4*, 295–312. doi:10.1016/0959-4752(94)90003-5

Sweller, J. (2005). Implications of cognitive load theory for multimedia learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 19–30). New York, NY: Cambridge University Press.

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, *22*, 123–138.

Sweller, J., Ayres, P., & Kalyuga, S. (2011). *Cognitive load theory*. New York, NY: Springer.

Sweller, J., & Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, *2*, 59–89.

Tindall-Ford, S., Chandler, P., & Sweller, J. (1997). When two sensory modes are better than one. *Journal of Experimental Psychology: Applied*, *3*, 257–287. doi:10.1037/1076-898X.3.4.257

Zhu, X., & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction*, *4*, 137–166.