The use of learning tools for active knowledge construction to develop coding skills

Abstract: Research indicates that providing students with learning tools to scaffold them during a CS1 course can positively affect the manner in which they develop programming skills. Some examples of learning tools are Scratch, Lego Mindstorms and programming strategies. These learning tools provide an opportunity for students to actively learn, as the tools' fundamental design focuses on multi-dimensional, tangible objects. This paper presents the qualitative and quantitative results which formed part of a study that focused on many learning principles for CS1 courses. Active learning, discussed in this paper, was one of the learning principles. The results to date are promising.

Introduction

Tools provide a foundation on which programming can be taught. The purpose of such tools is to provide a graphical development environment that delivers a visual interpretation of how to develop a computer program (Ford, 2009). The idea of interactive tools, such as games, as a pedagogical approach to teaching-and-learning computer programming is not new (Lawhead, 2002). This said, using games as a learning tool is advocated as games have the potential to positively contribute to successful learning (Piteira, 2011). Amongst others, tools provide two necessary elements for learning, namely understanding and motivation (Piteira, 2011, Yamazaki, 2015). Robots, in particular, provide an innovative teaching tool for building students computer programming skills.

The tools provide a platform for students to build, reinforce and practice fundamental computer programming concepts, while adding an element of fun. They can scaffold students learning because they use action instead of explanation, accommodate a variety of learning styles and skills, reinforce mastery skills, provide an opportunity to practice, and afford an interactive, decision making context. More importantly, tools provide students with an opportunity to 'tinker' with varying possible code combinations. This paper presents the results of students that were provided with opportunities to make use of tools when learning CS1.

Background

According to Kagan (2009) the domain-based knowledge that instructors transfer to students is one of the least important aspects of teaching-and-learning as information is not instruction. In reality, each student brings their own knowledge framework into the teaching environment and unless students are given an opportunity to actively engage with others and construct knowledge, the learning is likely to fail (Ben-Ari, 1998). The term 'active learning' has its roots deeply entrenched within authentic learning as well as social constructivism. MacDonald (2005) provides a useful definition of what makes learning active by saying that:
Active learning is a process whereby students are actively engaged in constructing knowledge in a meaningful, realistic context through exploration reflection and social discourse with others, rather than passively receiving information (MacDonald, 2005).

Instructors can make use of knowledge construction tools that confront the student with a phenomenon, or include a construction kit which encourages investigation and 'tinkering' (Karagiorgi, 2005). For example, a phenomena can be viewed as a real-world simulation where investigation can take place, and a construction kit could involve the use of Lego or Logo (Karagiorgi, 2005). These construction kits can be seen as being part of Vygotsky's theory regarding the 'zone of proximal development'. Students form partnerships with the tools, which provide strategies that experts use to solve problems, thus providing scaffolding relevant to the student's ability level (Mercer, 1992).

The design of the pedagogical intervention for the computer programming module made use of the following construction kits:

Table 1: Construction Kits

Semester 1 **problem solving knowledge construction kits**	**Semester 2** **computer programming construction kits**
• Puzzle-based learning (Kawash, 2012, Allan, 1997, Gonzalez, 2004) • Notational machine (Sorva, 2013) • Input-process-output (IPO) charts • Pseudo code • Programming strategies explicitly taught (deRaadt, 2008) • Interactive development environments, such as Scratch (Meerbaum-Salant, 2010)	• Notional machine (Sorva, 2013) • Input-process-output (IPO) charts • Pseudo code • Programming strategies explicitly taught (deRaadt, 2008) • Lego® construction (Mason, 2013, Yamazaki, 2015)

Aims, objectives and research questions

The aim of the paper is to show how construction kits or tools can assist CS1 students to understand programming concepts taught to them. The objectives of the paper are as follows:

- To describe how tools can be used to improve teaching and learning of CS1;
- To analyse quantitative results that were collected related to tools, namely programming strategies and the Lego Mindstorms questionnaires respectively; and
- To analyse qualitative results that were collected related to tools, as listed in the previous bullet point.

The research question that was answered was "Do students feel that tools scaffold the learning of a CS1 course?". A sub-question was "To what extent did the tools scaffold learning?".

Method

Research design

DBR was a vehicle used to address a complex real-world educational problem. Using iterative cycles of analysis, design and reflective inquiry a final set of educational design principles was produced. During the first iteration the educational design principles were developed and then refined over the second iteration to reflect the final eight design principles. This paper presents the results related to one of the design principles, namely to "provide learning tools that require students to be actively involved during knowledge construction while learning by 'tinkering' and so develop algorithmic programming skills". The study tested a proposed solution during iteration one (pilot study) and iteration two (main study).

Participants

For iteration one, there were 108 students enrolled for the CS1 course, comprising of module A and module B (over a year)..For iteration two, 135 students. The participants represented the diversity of the population in terms of gender, however there were many more male students than female ones. In terms of ethnicity the majority of students were black (83) and the remainder were from other groups (Coloured, Indian and White). Although the design was weaved into the teaching-and-learning students were provided with an explanation as to why the teaching-and-learning approach was different to that of other modules. Students were aware that research was being conducted into the teaching-and-learning of computer programming. Ethical considerations were implemented accordingly.

Instruments

The data that was collected in response to each of the research questions posed in this study illuminated the role that each design principle played in teaching- and-learning during the two iterations of the computer programming modules. For this paper qualitative results, as well as other questionnaires that focus on learning tools, are presented.

Data analysis

Questionnaires that made use of a Likert-type scale were developed. Scores, means and standard deviations calculated useful statistical data. A factor analysis was applied to validate results. The use of an electronic computerised software package, namely SPSS was applied.

Results

The results of each are now presented by design principle.
Data for exploring this design principle was generated by using two pedagogical intervention questionnaires, a focus group interview, a Lego Mindstorms questionnaire, programming strategies questionnaire, and an evaluation of students programming strategies. The results are discussed in the following sections.

Focus group interview

The analysis yielded two codes. Firstly, a code 'games are useful for critical thinking' with a code frequency of 50% emerged. An example of text that supported this code is:

> *I learnt how to build a robot from the scratch and write a program of how it must function, for example, how to turn when it reaches a certain point.*

Secondly, a code 'Scratch is useful for programming' with a code frequency of 70% also emerged. An example of text that supported this code is:

> *I believe the games that we played before hand - they were beneficial in a way. I believe they allowed us to firstly just open up with the rest of your classmates and - they also allowed - they also make you start thinking out of the box, because that was what mostly you were firstly taught or stressed upon that we need to think out of the box when it comes to programming.*

As qualitative analysis perceived as a ladder of analytical abstraction where data is firstly summarised and packaged, and then repackaged and aggregated, further analysis yielded categories and themes that identified relationships in the data (Troskie-de Bruin, 2013). For example, the code that the researcher suggests be labelled 'games are useful for critical thinking' and 'Scratch is useful for programming'.

Programming strategies questionnaire

During the second iteration students completed a questionnaire regarding the use of programming strategies that was used as a tool when learning computer programming skills. Strategies can assist students when they were faced with the task of developing a software solution. The responses, tabulated in Table 2, relate to whether or not students felt that strategies promote learning.

Table 2 indicates that as students have no prior experience of programming and, at best, demonstrate an ability to hold only fragile knowledge related to computer programming concepts, strategies do assist them with building algorithmic solutions. The percentage of students who agreed that computer programming strategies were very helpful to them was 83.6%, with 16.9% of students in disagreement. It is interesting to note that in question 6 of Table 1 many students struggled to adapt the strategy from one problem solving context to another. This means that these students have not developed sufficient higher order thinking skills (HOTS) (King, 2000). This is problematic as higher order thinking skills are directly related to the ability of students to solve a problem in one context and develop a solution in another context, namely at an algorithmic level (Levy, 2011).

Table 2: Students' responses to explicit teaching of computer programming strategies

Questions	**% Yes**	**% No**
Q1: I understand that a computer programming strategy is a specific guideline (algorithm) to solve a general problem.	96	4
Q2: I have read and understand most of the computer programming strategies provided to me.	78	22
Q3: I am able to identify what a computer programming strategy does when I see it.	81	19
Q4: I am able to make use of the correct computer programming strategy when a problem is presented to me.	76	24
Q5: I found computer programming strategies useful when solving a problem.	91	9
Q6: I find it easy to take a strategy and change/adapt the strategy to the problem that I need to solve.	67	33
Q7: I find that a computer programming strategy can assist me in solving a problem faster, with more accuracy.	85	15
Q8: In the last semester test I made use of the strategies given to me as an appendix at the end of the question paper.	86	14
Q9: A computer programming strategy can improve my programming abilities because I can learn from the strategy and have better insight into the problem.	92	8
Q10: I feel that computer programming strategies can assist me with learning programming skills.	92	8
Q11: When I am reading through a problem I very quickly start getting a mental picture of which computer programming concepts are needed to solve the problem.	70	30
Average	**83.16%**	**16.9%**

Evaluation of students' programming strategies results

During the second iteration students' end-of-semester assessment was analysed to determine the presence or absence of strategies. There were two questions in the assessment that were analysed, namely Question 4 and Question 5 (Table 4 shows the skill set, context, level of programming concepts and abstract thought required for each question).

Table 3: Presence or absence of a computer programming strategy

Question	**Strategy absent**	**Strategy present**
4	55%	45%
5	11.5%	88.5%

The presence or absence of a strategy was analysed by the researcher, the results of which are tabulated in Table 3. It appears that the difficulty level associated with an assessment question determines the presence or absence of a strategy, as shown in Table 3.

Strategies for novice programmers are seemingly only useful if the transfer of learning is not too wide. This is confirmed by Vygotsky's theory regarding the ZPD. The use of a strategy as a tool to assist students when learning computer programming is seemingly only beneficial if students are 'learning on the edge of what they know'.

Table 4: Computer programming skill set required to solve problems

Computer Programming skill set	**Question 4**	**Question 5**
Level of understanding required when reading the question (easy, moderate, difficult).	Difficult	Easy
Amount of 'story telling' information provided that could interfere with students thought processes.	Considerable	Little
Computer programming constructs required to solve the problem.	Input, processing, output, selection, looping	Input, processing, output, looping, arrays
The degree to which the strategy needs to be adapted from one context to another.	Hard	Moderate

Lego Mindstorms questionnaire

The analysis yielded the category ‘fun way to develop programming skills’ with a code frequency of 100%. An example of text that supported this code is:

I learnt to think about solving problems in different ways and just have fun while learning.

As a tool, the Lego Mindstorms provided students with an opportunity to further enforce the fundamental skills (Table 5) needed to write programming solutions. As indicated in Table 5 there was a high response rate (yes) for Lego Mindstorms being an effective tool to reinforce programming concepts.

Table 5: Responses regarding Mindstorms reinforcing learning

Programming concept	**‘yes’**	**‘no’**
Problem solving	21	
Breaking problems into small steps	20	1
Practice algorithmic skills	20	1
Assist in understanding how to solve problems better algorithmically	21	
Assist in coding programs that require input-processing-output	19	2
Assist in coding programs that require methods	21	
Assist in coding programs that require repetition	21	

Conclusion

The analysis of the focus group interview indicates a code frequency of 50% and students Agreed (3) that classroom activities included the use of tools. Additionally, these tools promoted learning, as indicated by a code frequency of 70%.
The researcher noted that 83.16% of students felt that the programming strategies promoted learning as these strategies scaffold students. Only 16.9% of students felt that the strategies did not promote learning.
Students programs were evaluated for the presence or absence of a programming strategy. The presence or absence of a strategy depended on the difficulty of the problem that students needed to solve. A total of 55% of students made use of a strategy when problems that ranged between ‘Easy’ and ‘Moderate’ were presented to them. It was noted that 5% of students did not make use of a strategy. However, if the problem was deemed to be categorised as ‘Difficult’, only 11.5% of students made use of a strategy. The gap between the ‘known’ and the ‘unknown’ was too wide and 88.5% of students could not identify the relationship between the problem and the strategy.

The analysis of the Lego Mindstorms questionnaire indicates a code frequency of 100% where students felt that the Lego Mindstorms provided an opportunity to actively engage with tools as well as that these tools promoted learning.

References

ALLAN, V. H., KOLESAR, M.V. 1997. Teaching Computer Science: A Problem Solving Approach that Works. *SIGCUE Outlook,* 25**,** 9.
BEN-ARI, M. Constructivism in Computer Science Education. *In:* SIGSCE, ed. SIGSCE'98, 1998 Atlanta GA USA. ACM.
DERAADT, M. 2008. *Teaching Programming Strategies Explicitly to Novice Programmers.* Doctor of Philosophy, University of Southern Queensland.
FORD, J. L. J. 2009. *Scratch programming for teens,* USA, Course Technology.
GONZALEZ, G. 2004. CONSTRUCTIVISM IN AN INTRODUCTION TO PROGRAMMING COURSE. *Consortium for Computing Sciences in Colleges,* 19**,** 7.
KARAGIORGI, Y., SYMEOU, L. 2005. Translating Constructivism into Instructional Design: Potential and Limitations. *Educational Technology & Society,* 8**,** 11.
KAWASH, J. Engaging Students by Intertwining Puzzle-Based and Problem-Based Learning. SIGITE'12, 2012 Calgary, Alberta, Canada. ACM.
KING, F. J., GOODSON, L., ROHANI, F. 2000. Higher Order Thinking Skills. Publication of the Educational Services Program.

LAWHEAD, P. B., BLAND, C.G., BARNES, D.J., DUNCAN, M.E., GOLDWEBER, M., HOLLINGSWORTH, R.G., SCHEP, M. A Road Map for Teaching Introductory Programming Using LEGO Mindstorms Robots. ITiCSE-WSR, 2002.

LEVY, R. B., ITURBIDE, J.A.V. A Problem Solving Teaching Guide based on a Procedure Intertwined with a Teaching Model. ITiCSE'11, June 2011 2011 Darmstadt, Germany. ACM, 1.

MACDONALD, J. Rules of engagement: Fostering active learning for performance improvement. Interservice/Industry Training, Simulation, and Education Conference, 2005. 9.

MASON, R., COOPER, G. 2013. Mindstorms robots and the application of cognitive load theory in introductory programming. *Computer Science Educational,* 23**,** 296–314.

MEERBAUM-SALANT, O., ARMONI, M., BEN-ARI, M. Learning Computer Science Concepts with Scratch. ICER 2010, 2010 Aarhus, Denmark. ACM, 8.

MERCER, N., FISHER, E. 1992. How do teachers help children to learn? An analysis of teachers' interventions in computer-based activities. *Learning and Instruction,* 2**,** 17.

PITEIRA, M., HADDAD, S.R. Innovate in Your Program Computer Class: An approach based on a serious game. OSDOC'11, 2011 Lisbon, Portugal.

SORVA, J. 2013. Notational Machines and Introductory Programming Education. *ACM Transactions On Computing Education,* 13**,** 31.

TROSKIE-DE BRUIN, C. 2013. Qualitative research: Requirements for post-graduate study projects. *In:* (ASEV), R. A. D. C. (ed.). Stellenbosch.

YAMAZAKI, S., SAKAMOTO, K., HONDA, K., WASHIZAKI, H., FUKAZAWA, Y. Comparative Study on Programmable Robots as Programming Educational Tools. *In:* FALKNER, D. D. S. A. K., ed. Australasian Computer Education (ACE) Conference, 2015 Sydney, Australia. Australia Computer Society Inc.