Chapter 3
Combatting the War Against Machines: An Innovative Hands-on Approach to Coding

Jacqui Chetty

Abstract The twenty-first century has been an era of technological advances that has surpassed previous decades. This is largely due to the level of innovation in the fields of artificial intelligence, robotics and automation. However, learners are often reluctant to choose computer programming (coding) as a subject due to its perceived difficulty. Nevertheless, it is also well known that learners who are introduced to computer programming at a young age become the computer science university graduates of tomorrow. Learners' hesitancy towards computer programming is due to the complex, abstract nature of the discipline. To this end, innovative tools are proving useful for learners to overcome such barriers. This chapter provides effective strategies to teach the fundamental concepts of computer programming using robotics, specifically Lego Mindstorms robots. The approach taken is hands-on, student-centred and visual. Learners develop coding solutions through designing and coding real-world problems, visually correcting their imprecisions. This chapter includes learning activities and practical examples from case studies. The inequality of women in the workplace, especially women in IT, is also addressed. A discussion around effective approaches to teaching girls' coding is included as research indicates that girls' learning requirements for coding are different to those of boys.

Keywords Lego Mindstorms robotics · Innovative tools · Computer programming · Learner-centred learning · Authentic learning

3.1 Introduction

Learners enrolled for a computer programming module for the first time often find it challenging to understand the fundamental concepts surrounding the discipline. Equally, educators find it difficult to teach such learners. Research indicates that

J. Chetty (✉)
University of Johannesburg, Johannesburg, Gauteng, South Africa
e-mail: jacquic@uj.ac.za

M.S. Khine (ed.), *Robotics in STEM Education*,
DOI 10.1007/978-3-319-57786-9_3

traditional pedagogical approaches do not lend themselves towards a positive experience for both learner and educator (Lister 2011). Given that pedagogical approaches are a powerful determinant as to whether a learner will be successful in a programming module, it is worthwhile investigating the variety of pedagogies and related tools available, to determine whether there is an approach that could adequately scaffold learners studying such modules. Research indicates that using games as a pedagogical tool to teach learners, the art of algorithmic problem-solving as well as programming has successfully been implemented to scaffold learners (Badger 2009).

The idea of using games to teaching fundamental computer programming concepts is not new (Lawhead et al. 2002). Using games as a learning tool is advocated as games have the potential to positively contribute to successful learning (Piteira and Haddad 2011). Lego Mindstorms robots is one such game that provides an innovative teaching tool for building learners' computer programming skills. Amongst others, the game provides two necessary elements for learning, namely understanding and motivation (Piteira and Haddad 2011). It provides a platform for learners to build, reinforce and practice fundamental computer programming concepts, while adding an element of fun. Lego Mindstorms scaffolds learners' learning because it uses action instead of explanation; accommodates a variety of learning styles and skills; reinforces mastery skills; provides an opportunity to practice; and affords an interactive, decision-making context.

Lego Mindstorms robots may prove to be an effective teaching tool to scaffold learners as it assists them with refining the art of computational thinking and enables the learning around the design of complex algorithms. The scaffolding becomes an instrument for educators to 'bridge' learners who are navigating a new discipline, as they clasp frantically to fragile knowledge in their minds. Structured scaffolding that is maintained for a reasonable period of time allows the fragile knowledge to solidify and become entrenched into the mind of the learner.

This chapter provides an overview of Lego Mindstorms robots as a tool that can be used to teach or reinforce fundamental computer programming concepts. Guidelines regarding teaching, managing and assessing such lessons are also deliberated. Gender similarities and differences when learning computer programming are also highlighted so that educators are aware of how to ensure that both genders receive appropriate teaching and learning.

3.2 Difficulties Faced by Learners Learning a Computer Programming Language

The skills expected for computer programming are complex, and learners worldwide find it very difficult to solve problems (Mead et al. 2006; Organisation for Economic Co-Operation and Development-OECD 2004). The problem arises as learners need to articulate a problem into a programming solution (Garner et al. 2005; Lahtinen et al. 2005) by combining syntax and semantics into a valid program

(Winslow 1996) through the construction of mechanisms and explanations (Soloway 1986). In order to achieve this, learners need to be able to apply fundamental computer programming concepts (Garner 2005; Robins et al. 2003) and understand abstract concepts (Lahtinen et al. 2005). When learners are faced with trying to absorb and understand too many new concepts at one time, their working memory may become overloaded. Their overloaded working memories make it very difficult for them to understand the concepts taught to them. The idea of working memory and load capacity is also known as cognitive load theory (Mason et al. 2015).

3.3 Cognitive Load

Humans are limited to a working memory capacity that is strictly bounded and relatively small (Mason et al. 2015). This means that due to our limited working memory capacity, our memories can become overloaded and our cognitive performance can decline. This is particularly true when novice learners are faced with the fundamentals associated with computer programming concepts as these concepts are fraught with abstract ideas or higher order thinking skills (HOTS). Such concepts are often layered, one on top of the other, before a learner is able to design and construct computer programs. Given that such learners are new to the discipline (novices), their cognitive load increases exponentially, often exceeding their critical threshold level of cognitive capacity (Mason et al. 2015).

It is therefore not unexpected for research to indicate that the results linked to computer programming modules aimed at novices more often than not have a particularly high failure rate. When such learners are presented with a subject, such as computer programming, they struggle as their cognitive load is pushed to capacity. These learners often cannot adapt as they are expected to learn concepts that require abstract reasoning, also known as computational thinking (Bower and Falkner 2015).

3.4 Computational Thinking

Computational thinking (CT) can be defined as the ability of a learner to develop problem-solving strategies and techniques that assist in the design and use of algorithms and models (Falkner et al. 2015). According to Lister (2011), such thinking needs time to develop. In fact, most learners possess limited CT in the early stages of their lives, but such skills should develop and mature, given that learners are educated and receive formal training (Lister 2011). However, many learners' computational thinking is not developed, and when they are confronted with a discipline such as programming, they are unable to think in a computational manner. Lego Mindstorms robots provides an excellent opportunity to develop CT through the use of problem-solving. Lego Mindstorms and problem-solving are tantamount providing a rich environment that can develop programming skills.

3.5 Problem-solving

Teaching computer programming using a problem-solving approach has become popular over the last decade (Falkner and Palmer 2009; Guillory 2011; Levy and Iturbide 2011; Organisation for Economic Co-Operation and Development-OECD 2004). The idea is that if a student learns how to solve one type of problem using a problem-solving approach, that student should be able to solve other problems of a similar nature, regardless of programming languages (Pears et al. 2009; Winslow 1996). Problem-solving provides an opportunity for students to construct programs using English-like lines of instructions as English is a language that is familiar to them. Expressing algorithmic instruction becomes easier as students are learning on the 'fringes' of what they know. The language is the known and the problem-solving (algorithmically) is the unknown. Lego Mindstorms instructions are English-like, enticing learners to create instructions that are relevant and applicable to the problem at hand.

3.6 Pedagogies Aimed at Computer Programming

Pedagogical approaches to teaching and learning can follow many paths, as there is an abundance of educational paradigms and theories available to teachers and educators alike. Some examples of such paradigms relate to behaviourism, cognitivism and humanism (Knowledgebase 2012). However, social constructivism is a philosophy that is well suited to learning computer programming.

3.7 Social Constructivism and Computer Programming

Social constructivism is a philosophy and a learning theory that has established itself in the last decade (Karagiorgi and Symeou 2005). It is a didactic approach that provides an opportunity for learning to take place in a social, interactive and collaborative manner. Central to this approach is that teaching and learning is an active process of constructing knowledge in a social setting, where students learn collaboratively. The idea is that during the classroom experience, new knowledge is constructed in a social setting. The new knowledge is based on students' past experiences and the hypotheses of the environment in which they learn. An essential aspect concerns how knowledge is being shaped through the use of symbolic tools, such as language (Kozulin and Gindis 2003). The intention is that students develop skills, such as reasoning, problem-solving, the development of higher mental processes and metacognition (Kozulin and Gindis 2003).

Computer programming is a discipline that requires critical thinking, teamwork and the development of software solutions (active process of constructing software

solutions). Social constructivism could prove to be an ideal pedagogical approach in this context. Scholars, such as Vygotsky, Piaget and Bruner, believed learning to be an active contextualised process that requires social negotiation (Bruner 1960; Corney et al. 2012; Vygotsky 1978). The process of developing software solutions emulates the philosophies surrounding social constructivism (Chetty 2016).

Parallels can be drawn between social constructivism and computer programming. Computer programming solutions are often developed through a process of discussion. A team of individuals, in collaboration with one another, are required to solve problems and find solutions (Farrell 2010). For example, a computer program is designed and developed using a formal computer programming language, such as C#. The design and the development of the program is a process that requires analogical reasoning, critical thinking, problem-solving and social negotiation. This process often takes place in collaboration with team members, where the design and the development of a program to be constructed requires team members to discuss and negotiate designs, workload, workflow and optimal solutions.

Although social constructivism is an underlying paradigm that can form a foundation for teaching, authentic learning is a platform that can transform the theoretical constructs of social constructivism into practice.

3.8 Authentic Learning Shaping the Design

Bruner stated that there is a real difference between learning about a domain (such as computer programming) and learning to be a computer programmer (Bruner 1960). While facts and knowledge can be taught, these only take on meaning and relevance when students discover the benefits of actively participating during learning as opposed to passively listening in classes (Lombardi 2007). When students actively participate in learning, they learn to 'do', they collaborate with others and they form communities of practice.

Herrington's Authentic Learning Framework, which comprises nine elements (Herrington and Parker 2013), is an excellent example of how authentic learning can be incorporated into the classroom. Although the framework consists of many elements, this chapter considers two elements that can be included when teaching Lego Mindstorms robots (although others could also be included).

3.9 Authentic Context

An authentic context is a physical environment that reflects or mimics the way in which the constructed knowledge is to be used and produced in the real world (Herrington 2006). Some scholars believe that it is the only context in which learning becomes meaningful. These scholars were the pioneers responsible for developing an authentic learning model that bridged the gap between theoretical

learning in the classroom and real-life application in the work environment (Herrington 2006). The learning environment can be described as one in which the design of the classroom setting preserves the complexity of the real-life setting. Furthermore, such a setting provides purpose and motivation for learners. It also provides an environment where educators and learners alike can discuss and explain ideas within the context of the real-life situation, making no attempts to fragment or simplify the environment (Herrington 2013a).

Given the notion of what an authentic context is, what would constitute a typical real-world setting for a computer programmer? In order to illustrate the setting, it is important to describe the activities and habits of a computer programmer during any given day so that the setting can reflect this. Computer programmers work on an individual basis as well as within a team. In both of these situations, their goal is to solve problems and develop application software solutions (Career One Stop 2008). Therefore, the workspace environment should reflect this. The working space may consist of a personal space that allows programmers a certain amount of privacy, free from interruption (Barker 2012). In this space, they critically analyse, have time to think creatively and develop programming solutions. The need will arise for them to work collaboratively, and a communal space will then be required. This space should allow them to communicate effectively with one another. Whiteboards and other essentials should be available so that they can discuss and visualise solutions in order to find an optimal solution. The idea is to provide an environment for learners to 'to be programmers' by building Lego Mindstorms solutions like they would be in the real world.

3.10 Authentic Task

Researchers and experts worldwide agree that an authentic learning activity represents a problem that has real-world relevance, is ill-defined and needs to be completed over a period of time (Brannock et al. 2013; Herrington 2006, 2013b; Lombardi 2007). Real-world relevance is concerned with problems that match the real-world tasks of professionals in practice. Such problems are normally 'messy' or ill-defined. Ill-defined problems are problems that when described to students are open to interpretation, as opposed to problems that are developed step-by-step. Instead of being highly prescriptive, ill-defined problems provide an opportunity for learners to identify the steps needed to complete the activity (Herrington et al. 2006). As ill-defined problems are more complex, learners need a longer period of time to complete such activities. The longer time period also allows learners to reflect on the choices that they are making regarding the solution, and this enhances their metacognitive skills (Lombardi 2007).

Authentic learning activities provide an opportunity for learners to construct new knowledge instead of reproducing existing knowledge. In order to achieve this, learners are provided with multiple sources from which they can draw information,

examine the problem from many angles, distinguish relevant information from irrelevant information and formulate a product (Lombardi 2007). For example, learners can be asked to develop software solutions, given a real-world problem. These tasks should be completed in collaboration, where learners are given the opportunity to discuss problems, ideas and solutions, thus learning from one another, before completing the task. Authentic tasks can easily be incorporated into a Lego Mindstorms environment providing endless imaginative projects.

3.11 LEGO Mindstorms Robots: A Tool to Scaffold Learning

Lego Mindstorms robots have become a popular pedagogical tool to teach and learn introductory computer programming concepts (Lawhead et al. 2002; Lui et al. 2010). The emphasis is on the word 'tool', where robots create a rich environment that provides a platform for novices as well as experienced educators to implement a laboratory experience for learners to learn programming skills in an interesting, unique and challenging manner. In effect, Stein (1998) challenges the computer science teaching community to move from the premise that computation is calculation to the notion of computation is interaction. Robots would be a natural way to explore such a concept.

Lego Mindstorms robots form part of Lego education and can be bought through a representative responsible for retailing such toys. The Mindstorms consists of building components, a programmable brick, active sensors and motors. There is software for which both graphical user interface (GUI) and command line interfaces are available. The robots, together with their associated interfaces, provide an opportunity for educators to transform classrooms into rich laboratory or software studios, where learners can experience learner-centred learning, collaborative learning and peer-to-peer programming experimentation (Yamazaki et al. 2015). This environment provides an opportunity for learners to 'put their programming skills to the test' as what they program comes to life through the Lego Mindstorms robot. They can visually understand 'what works', 'what does not work' and 'why'. Figure 3.1 shows a robot that is about to perform a task.

Lego Mindstorms robots provides an opportunity for learners to understand fundamental computer programming concepts that are, by their very nature, abstract (deRaadt 2008). These concepts are not analogies with the real world (Piteira and Haddad 2011). However, the Lego Mindstorms programming hides the abstract complexity by providing a fun, click and drag, prompting interface to assist with such analogies.

Introducing Lego Mindstorms robots provides a unique opportunity to transform a classroom environment that can create a degree of motivation in which (Piteira and Haddad 2011):

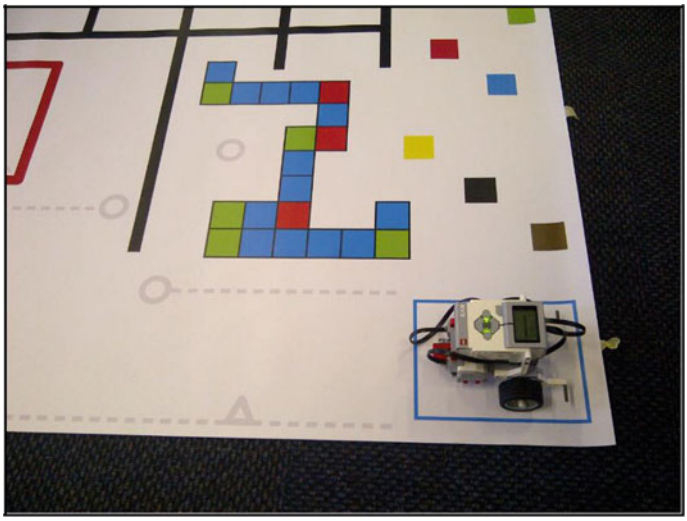

Fig. 3.1 EV3 Lego Mindstorms robot

(a) Learners are given an opportunity to 'grapple' with real-world problems;
(b) The Lego Mindstorms robot becomes a learning tool that can scaffold learners;
(c) Fragile knowledge of abstract programming concepts can be reinforced; and
(d) Learners are given an opportunity to experiment, explore and enjoy programming.

3.12 The Motivation Factor

Research indicates that emotions, such as hope, anger, relief, anxiety and boredom, are significantly related to motivation, learning strategies, cognitive resources, self-regulation, and academic achievement, as well as personality and classroom antecedents (Pekrun et al. 2002). According to Jenkins, motivation in particular is a crucial component related to learners' success. Although motivation is difficult to quantify, Jenkins has identified expectancy and value as two factors, which when multiplied can predict learners' motivation (Jenkins 2001). Expectancy is related to the extent to which learners feel that they are able to succeed. Value is related to what they expect to gain. For example, confident learners who feel that they are able to succeed will attach a value or goal related to high marks. They will most likely score high in the area of motivation as motivation = expectancy $*$ value (Jenkins 2001).

A motivated learner would therefore experience emotions related to hope, enjoyment and pride, whereas an unmotivated learner would experience emotions related to anger, frustration, anxiety and boredom. Lego Mindstorms robots provides an opportunity for learners to experiment and explore. The idea of learning through play is an effective tool to create personal motivation and satisfaction of learning (Piteira and Haddad 2011).

3.13 A Hands-on Approach to Teaching Lego Mindstorms Robotics

Lego Mindstorms robots can be purchased as Lego Mindstorms NXT or the newest version, namely Lego Mindstorms EV3 (lego.com). The EV3 has step-by-step instructions to build a variety of robots, as well as all the components needed to execute a variety of programs so that the robot can perform actions. Figure 3.2 shows some of the components that make up the EV3 robot.

The EV3 brick has a LINUX operating system, memory, ports, USB adaptors and a power supply. There are ports A to D and ports 1 to 4. Each port is an entry/exit point where cables are attached and link to the motors and other sensors. The brick provides the necessary hardware and software for learners to write executable programs. Once learners have built the robot using the Lego pieces, the programmable brick and other components, learners can start writing and executing programs.

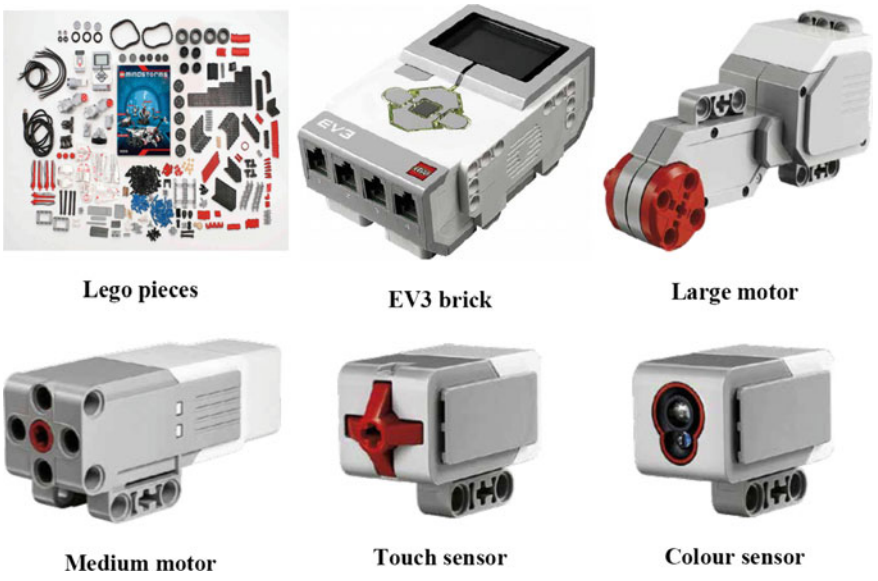

Fig. 3.2 Lego Mindstorms EV3 kit

The programs can be written using the EV3 brick interface or a software application can be downloaded for free (EV3 software). The EV3 software has a powerful interface, as shown in Fig. 3.3, and makes use of a click and drag approach to developing programs. The commands, shown in Fig. 3.3, are grouped by colours, for example green depicts action. All the commands are visible, and learners are not left guessing as to how a program can be built.

The software makes use of a click and drag approach, where learners locate a command and drag it onto the palette. Each command clicks into place, similar to that of puzzle pieces, as shown in Fig. 3.4. Figure 3.4 depicts a robot moving forward at a certain speed over a period of time. The robot then stops once that time period ceases to exist.

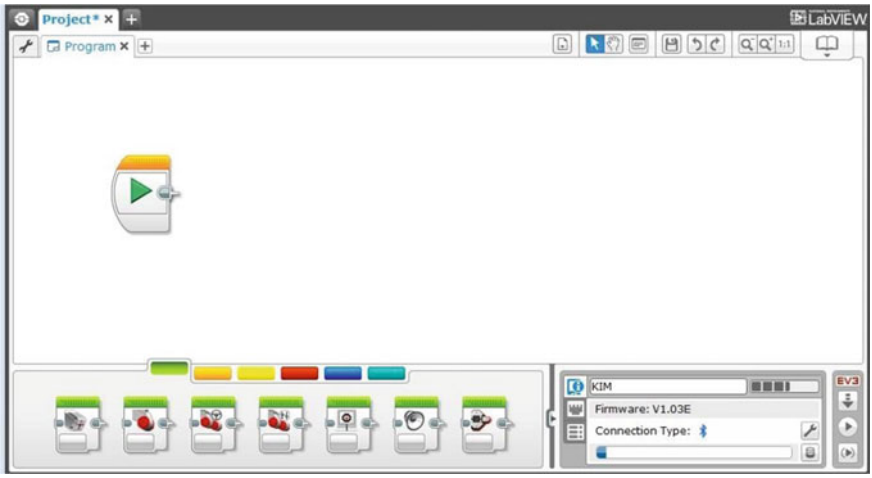

Fig. 3.3 EV3 software interface

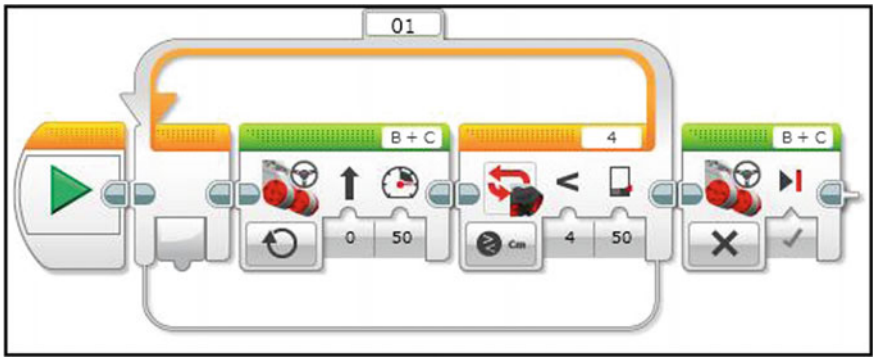

Fig. 3.4 Lego Mindstorms code depicting a loop

The following section provides curriculum guidance to educators that choose to include Lego Mindstorms robotics as part of the classroom learning. The learning discussed next consists of planning, managing and assessing Lego Mindstorms robot's curriculum.

3.14 Planning Lego Mindstorms Lessons

All planning, regarding of subject content, follows a particular pattern (Simmons and Hawkins 2015). For example, lesson plans are developed for a single lesson, collected lesson plans for a number of weeks and long-term plans. This chapter focuses on a single lesson as well as a combination of lessons for a number of weeks as Lego Mindstorms robots is generally taught over a period of a few weeks.

3.14.1 Lesson Plans

A lesson plan is a planning tool that contains all the information as well as the decisions that need to be considered before teaching (Simmons and Hawkins 2015). It consists of learning aims and learning objectives that constitute that lesson. The lesson plan provides the overall context for a lesson as well as activities, outputs and assessment criteria for those outputs. A Lego Mindstorms robots lesson plan should consist of a similar structure.

Lego Mindstorms robots provides a hands-on approach to learning and this means that even the best planned lessons can collapse. This is due to a number of reasons but one reason may be due to the use of technology as the focus of instruction. Many of us have experienced technology failing just as teaching is about to commence, even when many practice rounds have been put in place. For example, a cable can break down or a learner downloads a program but then executes the incorrect program. Teaching programming using innovative tools does increase the risk of lessons failing to achieve the desired outputs, and this can cause much stress to the educator. However, it is important to remember that learners and educators alike learn best when lessons do not go according to plan. A learner that spends a period of time problem-solving why a program does not execute is unlikely to make the same mistake again. An educator that encounters a cable problem more than ten times quickly learns how to solve that particular problem. I have experienced a few common errors that can occur.

Practical Tips

- The EV3 firmware is incompatible with the software application. This is easily rectified by downloading the correct version of firmware;

- Cables used to download programs from the software application to the EV3 brick can malfunction;
- A learner programs a robot to move in a straight line; however, the robot turns in circles. This is normally due to a cable (connected to a sensor on the robot) that is touching a wheel of the robot; and
- The learner has downloaded a program but executes the incorrect program from the EV3 brick. This can be rectified by ensuring that learners name their programs properly.

3.14.2 Planning to Plan

Prior to the development of a lesson plan, it is very important to consider the number of learners in a group as this affects classroom layout, as well as the seating plan. In most instances, it is advisable for learners to work in groups due to, firstly, the expense of the Lego Mindstorms robots. Secondly, Lego Mindstorms robots provides an opportunity for learners to work collaboratively, which is generally appreciated as an excellent approach to learning.

Practical Tip

- A group of two learners is optimal as a group that consists of any more than two learners often means that the other learners in the team do not get to participate —there are not enough tasks for everyone.

3.15 Lego Mindstorms Lesson Plans

3.15.1 Lesson Plan #1

The first lesson plan is relatively straightforward as learners are expected to build the Lego Mindstorms robot. Table 3.1 provides an example of the first lesson plan.

Practical Tips

- The batteries are normally inside the EV3 brick but check;
- Put the EV3 brick on charge while building the robot;
- It may be difficult to establish a timeline for building the robot. It may take from one to two hours; and
- Make sure that the cables are attached to the correct ports/components (i.e. sensors and motors).

Table 3.1 Lesson plan #1

Time	Learner activity	Teacher activity	Learning outcome
1 to 2 h	Provide an opportunity for learners to build the robot in groups where they can discuss and collaborate on their design The building of Lego Mindstorms robots offers learners the opportunity to create their unique bot—ready to be programmed and controlled in the manner that suits them Learners that feel more comfortable with step-by-step, Lego-type instructions can build their bot by choosing a design where the instructions for the building are already given to them Provide them an opportunity to deviate from their model to include a few components of their own	Welcome the class and provide an introductory talk about building the robot Point out the different components that are needed to build the robot Offer learners the opportunity to deviate from the set models by designing and building their own unique bot	The completion of the robot to a satisfactory level

3.15.2 Lesson Plan #2

The second lesson plan involves an explanation of the software application interface, also known as the Programming Canvas. There are a variety of instructions or commands, each grouped and located in a Programming Palette (divided into categories by colour).

The palettes are listed as follows:

Action blocks		Flow blocks		Sensor blocks		Data blocks	
☐	Medium	☐	Start	☐	Brick buttons	☐	Variable
	motor	☐	Wait	☐	Colour sensor	☐	Constant
☐	Large motor	☐	Loop	☐	Infrared sensor	☐	Array operations
☐	Move	☐	Switch	☐	Motor rotation	☐	Logic operations
	steering	☐	Loop interrupt	☐	Timer	☐	Math
☐	Move tank			☐	Touch sensor	☐	Round
☐	Display					☐	Compare
☐	Sound					☐	Range
☐	Brick status					☐	Text
	light					☐	Random

Table 3.2 Lesson plan #2

Time	Learner activity	Teacher activity	Learning outcome
1–2 h	The learners will familiarise themselves with the various palettes that form part of the interface Learners must remember the different palettes, such as Action and Flow. Ask learners to make a chart so that they can categorise the palettes, together with their commands. The chart can be created in such a way that it can be used in a later lesson (colour sensor). The robot can detect a colour and sound out the palette, such as Action and Flow (see lesson 4#)	Point out the different palettes that will be required by learners to develop a set of instructions Assist learners in understanding the different categories of blocks and why they are grouped accordingly	Learners are expected to be able to differentiate between the categories of Programming Palettes Learners should be able to name a few blocks from the Action blocks category
15 min	Learners create their first program by clicking and dragging the Move Tank command to the right of the Start command Learners must make sure that the two commands fit together like puzzle pieces	Perform the activity with the learners and remind learners where to find the Move Tank command Show learners how to correctly click the commands in place to form a program Explain to learners that a program is a set of step-by-step instructions performed in sequence	Learners are expected to have put the two commands together

There are two more blocks, namely the Advanced blocks and the My blocks, each consisting of a number of commands. Table 3.2 provides an example of the lesson plan.

Practical Tips

- Get learners to ask each other which palette holds which blocks;
- Point out to learners the difference between the medium and large motors;
- Make sure that learners understand the difference between move steering and move tank;
- Let learners measure the length when the tank moves forward for x—seconds, degrees and rotations; and
- Point out to learners that the Flow blocks/Wait command can be used to create instructions for sensors instead of the Sensor blocks.

3.15.3 Lesson Plan #3

Lesson plan #3 is a continuation of lesson plan #2 and therefore can be completed as part of the same session. This lesson plan consists of learning about the hardware associated with the robot and how learners can examine the hardware using the Programming Canvas. The hardware is directly linked to the programming commands, such as moving the wheels or instructing the robot to pick up colour. It is therefore important that learners understand how the hardware, cables and ports operate as well as connect with one another.

Figure 3.5 displays information about the EV3 brick. EV3 is the name of the brick, although this can be altered. Learners enjoy developing unique names for their EV3 brick. The brick information tab displays information about the brick, such as the battery level, firmware version and how much memory has been used. The Port view tab shows which sensors and motors are attached to the brick, indicated in Fig. 3.6.

The Available Bricks tab connects the brick to the EV3 software. The download button allows transfer of the program from the computer to the EV3 brick.

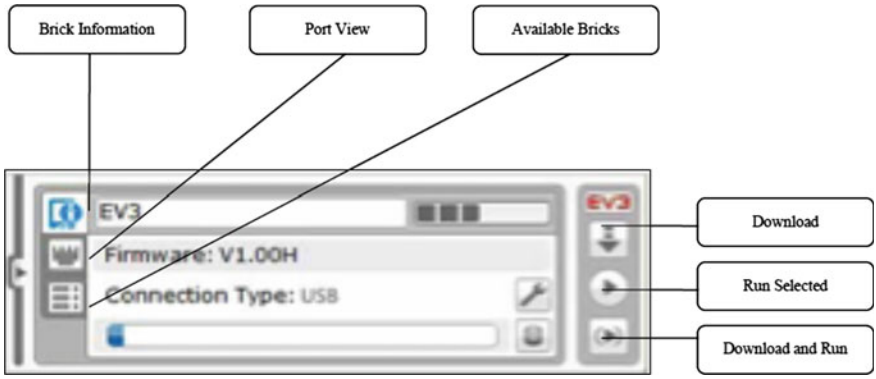

Fig. 3.5 Hardware page tabs on the left and the download/run buttons on the right

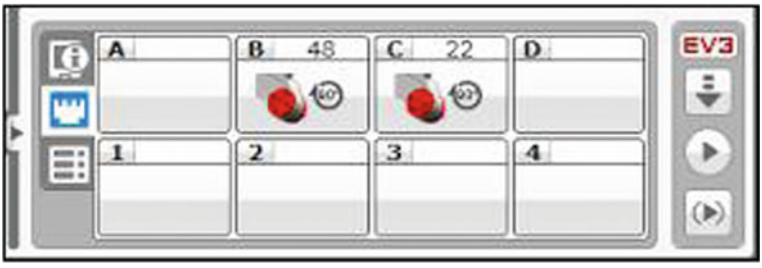

Fig. 3.6 Port view

Table 3.3 Lesson plan #3

Time	Learner activity	Teacher activity	Learning outcome
30 min	Learners observe the various ports, plugging in the different sensors Provide opportunities for learners to collaborate and discuss with one another what the different ports do Allow them to trace the cable leaving an exit point and entering another port Learners download and run a program, observing the activity at the port view	Point out each port associated with a sensor and ensure that learners understand the link between the robot and the port view Provide a small problem for learners to solve to provide an opportunity for learners to download and run a program	Learners are expected to feel comfortable with the various ports Learners should be able to identify which port is connected to which sensor Learners are expected to be able to download and run a program developed in the software application window

This option only allows for a download and does not run the program. The Download and Run option downloads the program and runs the program immediately. The Run Selected will download and run only the blocks that have been selected. This option is useful when fixing problems in a program (Griffin 2014). Table 3.3 provides a lesson plan.

Practical Tips

- Get learners to ask each other which palette holds which blocks—refer them to their charts that they have developed.

3.15.4 Lesson Plan #4

From lesson #4 onwards, the focus of the classroom learning is to provide an understanding of the fundamental concepts associated with computer programming, using Lego Mindstorms robots. These fundamentals[1] are as follows (the completed ones depends on the curriculum):

- Variables;
- Memory;
- Single-line step-by-step statements;
- Selection statement (if … else …);
- Repetition (looping);
- Arrays;
- Methods (functions procedures);

[1]An explanation around the fundamental concepts of programming for non-programmers

Table 3.4 Lesson plan #4

Time	Learner activity	Teacher activity	Learning outcome
Time varies for each programming concept taught. On average, each concept should take about 90 min to teach	Learners must discuss, in a group, the programming concept being taught, by providing examples of how that concept is useful or can be used as part of a software solution	Provide an understanding to learners around the programming concept taught, for example variables Provide much opportunity for learners to practice the programming concepts being taught Introduce real-world problems where learners can work in pairs to solve the problem	Learners are expected to understand each programming concept learnt by developing practical solutions that include the programming concept

- Objects; and
- Classes.

Each lesson plan following on from lesson #4 imprints a similar pattern or structure, the only difference being is the computer programming concept taught. Therefore, the layout for lesson #4 can be seen as a template, Table 3.4, that can be used for further lessons (i.e. lesson #5, lesson #6 and so on).

Practical Tips

- Spend a fair amount of time on each programming concept as learners often 'think' they understand until they have to solve a problem that they have never been exposed to before;
- Each of the programming concepts makes use of a variety of palettes. Other than the common Action blocks, the Flow blocks are prominent combined with the Sensor blocks; and
- The Colour sensor, Touch sensor and Infrared sensor are particularly useful for learning about selection (If…/else…) as well as repetition (For… loop/While… loop).

3.16 Managing Lessons

Classroom management can be one of the biggest sources of uncertainty and anxiety for educators and learners alike (Simmons and Hawkins 2015). Even the best planned lesson objectives can be diluted when the act of learning as well as the behaviour of a group of learners is unpredictable. The focus of managing lessons aimed at Lego Mindstorms robots probably falls into two main categories, namely managing the classroom environment and managing the planned lesson.

3.16.1 Managing the Classroom Environment

Experience suggests that it often becomes tricky to manage the behaviour of learners in the classroom as learners find the experience of building, as well as the development of solutions extremely exciting. The classroom experience is a very physical one, where learners move around, solve problems in pairs and are experiencing the joy of learning by doing. The robot moves around, makes sounds, bumps into objects and other robots, and this provides much entertainment amongst learners. It becomes rather difficult to bring calm to the classroom environment, especially when you need learners to take their seats so that a concept can be discussed or reflected upon. The noise levels within the classroom can be very high.

In order for the classroom environment to remain a positive one, it is important that the educator discusses with learners how learning will take place. Some points of discussion can be:

- **Focus learners' attention**—for example, educators can come to an agreement with learners that when a funny word, such as 'beetle juice', is uttered by the educator the learners respond with 'glug, glug, glug' and they all take their seats.
- **Take turns—**for example, educators can, when a pair is grouped, also be known as pair programming (Preston 2005), by allocating one learner as being the driver (typing in the code) and the other learner as being the navigator (observe/correct the driver). With each activity, the driver and the navigator are swopped around. Proper roles may reduce arguing about whose turn it is.
- **Listen and participate in class discussion**—for example, educators can ask a pair to demonstrate their programming solution and discuss the manner in which the solution was developed. Other pairs can rate the solution being demonstrated.
- **Length of time allocated to lessons**—educators must be aware that although learners are very engaged and the time allocated to participate within a lesson seems to go quickly, learners do tire and small breaks are necessary. It is easy to forget about providing a break when learners and the educator are very absorbed within an activity.

3.16.2 Managing the Planned Lesson

The adage 'failing to plan is planning to fail' cannot be more true, especially within a teaching and learning environment. It is very important, and maybe more so when making use of physical objects within a lesson plan, to be vigilant about planning. Carefully crafted learning objectives, writing lesson plans, developing problems for learners to solve and presentations regarding some part of the curriculum provide a structure to educators and communicate confidence and experience to learners.

- **Learning objectives**—other than developing objectives regarding the programming environment for Lego Mindstorms robots, the learning objectives should reflect fundamental programming concepts, linked to the educators' curriculum. There are a variety of books, such as The Art of Lego Mindstorms EV3 Programming and The Lego Mindstorms EV3 Discovery book, to name a few, that are very helpful.
- **Lesson plans**—these can follow the structure as seen above in Tables 3.2, 3.3 and 3.4.
- **Problems for learners to complete**—there is such a wide variety of problems that can be presented to learners. Regardless of the type of problem, real-world problems can provide an opportunity for learners to develop solutions within their world context. For example, ask learners to develop a solution where the robot behaves like an alarm. When an 'intruder' moves within 100 m of the robot, an alarm bell is activated.

3.17 Assessment

The use of assessment for an ICT subject has traditionally been poor although it is improving (Simmons and Hawkins 2015). Unless the ICT subject was receiving specialist ICT teaching, there is often little proof of assessment. Projects are normally completed by learners, and educators may allocate marks in an arbitrary manner, along with feedback, either verbal or written.

However, all assessment should provide a measure of performance against a target standard. Bloom sought to move away from assessing in such a way that learners were compared with one another, and instead compared to a set of objective criteria. Within the computing discipline, such assessment is more often than not a practical one. This can very much be aligned to Bloom's way of assessing as no two programming solutions may be the same. Learners think differently about a problem and often develop different ways of solving a problem.

Assessment should, of course, be aligned to learning outcomes. However, the type of assessment can be formative or summative assessment. For computing solutions, both types of assessment are useful. Whereby formative assessment provides an opportunity for both educator and learner to discuss what needs to be done to solve a problem as well as how to achieve a desired outcome, summative assessment relies on tests and examinations. For example, learners discussing their programming solutions with other students provide a wonderful learning opportunity as solutions always vary. Learners have opportunities to teach and learn from one another.

Formative assessment can be accomplished by the programming pair being asked to develop a solution. Once the solution is developed, the pair demonstrates the end result. The educator can then ask questions from each individual within the pair to verify that each learner did participate in the learning and that learning did take place for both learners.

Summative assessment can be difficult to realise as no two programming solutions are the same. The idea of a rubric that provides generic criteria, from which marks can be allocated, may be considered as useful. However, within this structure, the educator must provide leeway for out-of-the-box lateral thinking and unique solutions that some learners are bound to produce.

3.18 Reflection upon Lessons

While the manner in which learning takes place is important, it is equally important for learners to plan, manage and reflect on their learning (Laskey and Hetzel 2010), also known as metacognition. The term metacognition was originally associated with scholars such as Flavell, Zabrucky and Brown (Bransford et al. 2000). Metacognition encompasses learners consciously and actively understanding their cognitive aptitude and the ability to apply strategies to control cognitive thought processes. Such internal thought patterns extend into learners' daily lives where cognitive tasks are planned, regulated, coordinated and monitored. Learners with good metacognitive skills therefore have the potential to perform better in an academic environment (Schraw and Dennison 1994).

Within a Lego Mindstorms robot classroom, environment metacognition can be accomplished in many ways. For example, learners can be asked to discuss what they learnt within their pairs, within a group or with an educator. One way of doing this is to include the reflective thinking as part of a fun activity. The use of a Koosh ball shown in Fig. 3.7 can be an effective object to encourage reflective thinking. The educator throws or passes the Koosh ball to a learner, and the learner describes something about their classroom learning experience. The Koosh ball is then passed along to another learner and so on. Each learner is provided with an opportunity to reflect, and nobody within the group is to comment or criticise. From experience, learners find this type of activity enjoyable.

Fig. 3.7 Koosh ball often used for therapy

3.19 Teaching Lego Mindstorms to Both Genders

The teaching and learning of programming has always been a subject of much attention, due to the difficulties that learners encounter when learning this discipline. Over the decades, much research has been conducted so that researchers can better understand the ways in which learners learn to program. Exploration regarding gender and whether this influences learning is also an area that has been researched (Burnett et al. 2010; Carter and Jenkins 1999, 2001). Although there are many factors that influence learning to program, research indicates that gender is a significant factor in determining the way in which students approach learning to program (Funke et al. 2015). This section provides insight into teaching different genders and how to provide a teaching and learning environment that best suits each gender.

3.19.1 Gender Differences in Computer Science

In almost all western countries, women are severely underrepresented in the discipline of computer science. Only 20% of an intake within any given department is female (Funke et al. 2015). Given these statistics what can be learnt so that educators are aware of the situation and provide an environment that encourages female learners to enrol for programming courses.

Female learners tend to be less confident and they underestimate their ability (Carter and Jenkins 1999; Funke et al. 2015). Consequently, female learners often have weaker marks, only do what is required of them and they are less fascinated, adopting a pragmatic approach to programming. However, they are also more enthusiastic to seek assistance and readily attend extra tutorials, preferring smaller groups over larger ones. Interestingly, female learners are more consistent, and when confronted with a problem admit that there is a problem before the problem becomes a larger one. Another aspect that affects the learning process is emotions (Chetty and van der Westhuizen 2013). Research indicates that happiness has a positive effect on learning, and anxiety negatively influences the learning process; and the motivation of female learners (Funke et al. 2015). Communication is very important to female learners. Denner et al. show that girls benefit from collaboration where they work together as a pair, one being the driver and the other the navigator.

Male learners are more confident as they depict the typical role model of the male computer scientist. They seem to have more hands-on experience, try and test things out (scientific curiosity), and have more interest probably due to the gaming industry. The result is that male learners often produce better marks. However, male learners are often less structured and often do not admit when there is a problem until the problem at hand is almost insurmountable (Carter and Jenkins 1999). Additionally, they do not readily attend extra tutorials.

3.20 Teaching Lego Mindstorms Robots to Both Genders

Both genders can benefit from learning programming using Lego Mindstorms robots. Table 3.5 provides some practical ideas of how Lego Mindstorms robots can be used to encourage both genders to enjoy the process of learning to program.

Female programmers dominated the industry in the 1960s up to the 1980s, when a decline of female programmers began. This is unfortunate as female programmers are greatly required in this industry due to their unique abilities that yield excellent

Table 3.5 Learning for both genders using Lego Mindstorms robots

Both genders learning programming using Lego Mindstorms robots		
Lego Mindstorms robots learning	Female learners	Male learners
Scientific curiosity	Needs encouragement and can be through the use of real-world problems that are meaningful to them. Provide examples of female programmers, such as the first programmer was a female	Do not need much encouragement as they are naturally curious about robots
Solving smaller problems	Easily solve smaller problems so this may be a way of retaining interest in robots	Benefit from solving smaller problems due to their inability of admitting when a program has a 'bug' until the problem is very large. Encourage male learners to solve small problems in this manner
Student-centred learning / Collaborative learning	Communication is naturally good so encourage female learners to discuss programming problems and solutions as part of a group discussion. This may build up an excitement and happiness around programming, boosting confidence	Male learners may need to be encouraged to discuss learning. Provide an environment where they can discuss in pairs or small groups problems and solutions related to their learning. Be goal-oriented and specific
Small group learning	Any group learning is suitable for female learners. Provide them with an opportunity to discuss their feelings about solving a problem or working on a solution	Male learners may need encouragement and structure, such as providing a driver and a navigator when solving a problem
Real-world problems	Provide practical problems as they relate to their world, each group of students having a different world perspective	Provide practical problems as they relate to their world, each group of students having a different world perspective
Reflection	As female learners communicate and share well, reflection may be an opportunity to enhance their confidence. For example, allow them to share the successes they have experienced	Male learners may require encouragement. Provide a structure whereby you ask them to reflect on an aspect and prevent open-ended questions

programmers. Skills such as attention to detail, meticulousness and an ability to determine and correct 'bugs' within programs quickly, are just a few that come to mind.

3.21 Conclusion

Learners face many difficulties and challenges when presented with programming concepts. However, many of these challenges can be addressed when educators investigate and understand the ways in which learners learn programming best. Although not a silver bullet, good teaching enables learners and provides an environment that encourages learning.

Lego Mindstorms robots may be an effective tool in which the fundamental concepts related to programming can be presented to learners. Mindstorms provides an opportunity to encourage innovative learning styles, such as student-centred collaborative learning. These styles of learning are often successful with learners. Furthermore, Lego Mindstorms robots provides the much needed scaffolding required when teaching programming by presenting difficult programming concepts in a visual, step-by-step way that learners may find easier to grasp. The fun interactive manner in which learning occurs means that students learn from one another.

This chapter provides an overview of how teaching and learning can be accomplished through the use of Lego Mindstorms robots. These robots provide a wonderful opportunity for educators to include much needed scaffolding for an otherwise very difficult discipline such as programming.

References

Badger, M. (2009). *Scratch 1.4 Learn to program while creating interactive stories, games, and multimedia projects using Scratch Beginner's Guide*. Birmingham, Mumbai: PACKT.

Barker, E. (2012). *What do the best programmers have in common?* Retrieved from http://www.bakadesuyo.com/2012/08/what-do-the-best-computer-programmers-have-in/

Bower, M., & Falkner, K. (2015). *Computational thinking, the notional machine, pre-service teachers, and research opportunities.* Paper presented at the Australasian Computer Education (ACE) Conference, Sydney, Australia.

Brannock, E., Lutz, R., & Napier, N. (2013). *Integrating authentic learning into a software development course: An experience report.* Paper presented at the SIGITE'13, Orlando, Florida, USA.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn: Brain, mind, experience, and school: Expanded Edition.* Committee on Learning Research and Educational Practice (Ed.) Retrieved from http://www.nap.edu/catalog/9853.html

Bruner, J. (1960). *The process of education*. Cambridge: Harvard University Press.

Burnett, M., Fleming, S., Iqbal, S., Venolia, G., Rajaram, V., Farooq, U., et al. (2010). *Gender differences and programming environments: Across programming populations.* Paper presented at the ESEM '10, Italy.

Career One Stop. (2008). *A day in the life—Computer programmer*. Retrieved from http://www.youtube.com/watch?v=RQ_HdHSpDEg

Carter, J., & Jenkins, T. (1999). *Gender and programming: What's going on?* Paper presented at the ITiCSE'99, Poland.

Carter, J., & Jenkins, T. (2001, September). *Gender differences in programming?* Paper presented at the ITcSCE, Canterbury, UK.

Chetty, J. (2016). *An emerging pedagogy for teaching computer programming: Attending to the learning needs of under-prepared students in university-level courses* (Ph.D. Education Information Systems), University of Johannesburg, Johannesburg, South Africa.

Chetty, J., & van der Westhuizen, D. (2013). *"I hate programming" and other oscillating emotions experienced by novice students learning computer programming*. Paper presented at the EdMedia'13, Canada.

Corney, M., Teague, D., Ahadi, A., & Lister, R. (2012). *Some empirical results for Neo-Piagetian reasoning in novice programmers and the relationship to code explanation questions.* Paper presented at the Australasian Computing Education Conference, Melbourne, Australia.

deRaadt, M. (2008). *Teaching programming strategies explicitly to novice programmers.* (Doctor of Philosophy), University of Southern Queensland.

Falkner, K., & Palmer, E. (2009). *Developing authentic problem solving skills in introductory computing classes.* Paper presented at the SIGCSE'09, Tennessee, USA.

Falkner, K., Vivian, R., & Falkner, N. J. G. (2015). *Teaching computational thinking in K-6: The CSER digital technologies MOOC.* Paper presented at the Australasian Computer Education (ACE) Conference, Sydney, Australia.

Farrell, J. (2010). *Java™ programming* (5 edn). Course Technology, Cengage Learning.

Funke, A., Berges, M., Muhling, A., Hubwieser, P. (2015). *Gender differences in programming: research results and teachers' perception.* Paper presented at the Koli Calling '15, Finland.

Garner, S., Haden, P., Robins, A. (2005). *My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems.* Paper presented at the Australasian Computing Education Conference, Newcastle, Australia.

Griffin, T. (2014). *The Art of Lego Mindstorms EV3 Programming*.

Guillory, B. A. (2011). *Teaching through problem solving*. Retrieved from http://www.slideshare.net/bbieniemy/teaching-through-problem-solving1

Herrington, J. (2006). *Authentic e-learning in higher education: Design principles for authentic learning environments and tasks.* Paper presented at the AACE.

Herrington, J. (2013a). *Authentic contexts set the scene*. Retrieved from http://authenticlearning.info/AuthenticLearning/Authentic_Context.html

Herrington, J. (2013b). *Its the task that matters most*. Retrieved from http://authenticlearning.info/AuthenticLearning/Authentic_Task.html

Herrington, J., & Parker, J. (2013). Emerging technologies as cognitive tools for authentic learning. *British Journal of Educational Technology, 44*(4), 607–615.

Herrington, J., Reeves, T. C., & Oliver, R. (2006). Authentic tasks online: A synergy among learner, task and technology. *Distance Education, 27*(2), 15. doi:10.1080/01587910600789639

Jenkins. (2001). *Teaching programming—A journey from teacher to motivator.* Paper presented at the 2nd Annual LTSN-ICS Conference, London.

Karagiorgi, Y., & Symeou, L. (2005). Translating constructivism into instructional design: Potential and limitations. *Educational Technology & Society, 8*(1), 11.

Knowledgebase, L. T. (2012). Learning-Theories.com. Retrieved from http://www.learning-theories.com/

Kozulin, A., Gindis, B., Ageyev, V. S., & Miller, S. M. (Ed.) (2003). *Vygotsky's educational theory in cultural context*. Cambridge.

Lahtinen, E., Ala-Mutka, K., & Jarvinen, H. (2005). *A study of the difficulties of novice programmers.* Paper presented at the ITiCSE '05, Monte de Caparica, Portugal.

Laskey, M. L., & Hetzel, C. J. (2010). *Self-regulated Learning, metacognition and soft skills: The 21st century learner*. Retrieved from http://www.eric.ed.gov/PDFS/ED511589.pdf

Lawhead, P. B., Bland, C. G., Barnes, D. J., Duncan, M. E., Goldweber, M., Hollingsworth, R. G., et al. (2002). *A road map for teaching introductory programming using LEGO Mindstorms Robots.* Paper presented at the ITiCSE-WSR.

Levy, R. B., & Iturbide, J. A. V. (2011). *A problem solving teaching guide based on a procedure intertwined with a teaching model.* Paper presented at the ITiCSE'11, Darmstadt, Germany.

Lister, R. (2011). *Concrete and other Neo-Piagetian forms of reasoning in the novice programmer.* Paper presented at the Australasian Computer Education Conference, Perth, Australia.

Lombardi, M. M. (2007). Authentic learning for the 21st century: An overview. Retrieved from http://net.educause.edu/ir/library/pdf/ELI3009.pdf

Lui, A. K., Ng, S.C., Cheung, H.Y., & Gurung, P. (2010). Facilitating independent learning with Lego Mindstorms Robots. *acmInroads, 1*(4), 5.

Mason, R., Cooper, G., Simon, & Wilks, B. (2015). *Using cognitive load theory to select an environment for teaching mobile apps development.* Paper presented at the Australasian Computer Education (ACE), Sydney, Australia.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., St. Clair, C., et al. (2006). *A cognitive approach to identifying measurable milestones for programming skill acquisition.* Paper presented at the ITiCSE'06, Bologna, Italy.

Organisation for Economic Co-Operation and Development-OECD. (2004). Problem solving for tomorrow's world—First measures of cross-curricular competencies from PISA 2003. Retrieved from http://www.oecd.org/dataoecd/25/12/34009000.pdf

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., … Paterson, J. (2009). *A survey of literature on the teaching of introductory programming.* Paper presented at the ITiCSE-WGR'07.

Pekrun, R., Goetz, T., Titz, W., & Perry, R. P. (2002). Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational Psychologist, 37*(2). doi:10.1207/S15326985EP3702_4

Piteira, M., & Haddad, S. R. (2011). *Innovate in your program computer class: An approach based on a serious game.* Paper presented at the OSDOC'11, Lisbon, Portugal.

Preston, D. (2005). *Pair programming as a model of collaborative learning: A review of the research*. Paper presented at the CCSC: Central Plains Conference.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Educational Journal, 13,* 137–172.

Schraw, G., & Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary Educational Psychology, 19,* 16.

Simmons, C., & Hawkins, C. (2015). *Teaching computing*. London: Sage.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 9.

Stein, L. (1998). What we've swept under the rug: Radically rethinking CS1. *Computer Science Education, 8*(2), 118–129.

Vygotsky, L. (1978). *Mind and society*. Cambridge, MA: Harvard University Press.

Winslow, L. E. (1996). Programming pedagogy—A psychological overview. *SIGCSE Bulletin, 28*(3), 6.

Yamazaki, S., Sakamoto, K., Honda, K., Washizaki, H., & Fukazawa, Y. (2015). *Comparative study on programmable robots as programming educational tools.* Paper presented at the Australasian Computer Education (ACE) Conference, Sydney, Australia.