Learning to code: from procedural puzzle-based games to creative programming

Romero, Margarida
Université Laval
Département d'études sur l'enseignement et l'apprentissage / Faculté des Sciences de l'Éducation
2320 rue des Bibliothèques, local 1112. G1V 0A6. Québec. Canada.
margarida.romero@fse.ulaval.ca

Davidson, Ann-Louise
Concordia University
Department of Education / Faculty of Arts and Science
Faubourg Ste-Catherine Building, 1610 St. Catherine W., Montréal, Canada
ann-louise.davidson@concordia.ca

Cucinelli, Giuliana
Concordia University
Department of Education / Faculty of Arts and Science
Faubourg Ste-Catherine Building, 1610 St. Catherine W., Montréal, Canada
giuliana.cucinelli@concordia.ca

Ouellet, Hubert
Université Laval
Département d'études sur l'enseignement et l'apprentissage / Faculté des Sciences de l'Éducation
2320 rue des Bibliothèques. G1V 0A6. Québec. Canada.
hubert.ouellet@fse.ulaval.ca

Kate Arthur
Kids Code Jeunesse
51 Sherbrooke Ouest, H2X 1X2, Montréal, Canada.
kate@kidscodejeunesse.org

1. **ABSTRACT:** Learning to code is integrated in a growing number of schools worldwide. However, the learning to code activities shows important differences according to the creative engagement of the learners in the activity. We identify five levels of learning to code activities: (1) teacher-centered explanations or tutorials; (2) procedural, step-

ISSN: 2385-6203

by-step programming; (3) creative individual programming; (4) co-creative programming and (5) participatory co-creation of knowledge through programming.

2. **ABSTRACT:** Learning to code is integrated in a growing number of schools worldwide. However, the learning to code activities shows important differences according to the creative engagement of the learners in the activity. We identify five levels of learning to code activities: (1) teacher-centered explanations or tutorials; (2) procedural, step-by-step programming; (3) creative individual programming; (4) co-creative programming and (5) participatory co-creation of knowledge through programming.

3. **KEYWORDS:** Computational thinking, Code, Programming, Creative Programming, Creativity, Innovation.

KEYWORDS: Computational thinking, Code, Programming, Creative Programming, Creativity, Innovation.

4. **DEVELOPMENT:**

Learning to code as a 21st century skill

ICT skills, collaborative problem solving, (co)creativity and critical thinking are some of the skills needed to cope with the complexity of the 21st century (Griffin, McGaw, & Care, 2012; Voogt & Roblin, 2012). The concept of digital literacy or ICT skills has evolved over the last three decades from a techno-centered approach of using the computers and their programs towards the emergence of socio-constructivist, (meta)cognitive and co-creative uses of ICT (Azevedo, 2005; Katz, 2013; Romero, Hyvönen, and Barbera, 2012). However, it is not enough to know about information search, which constitutes the first stage of the ICT skills in education (UNESCO, 2011). In a lifelong learner posture, we must achieve the levels of deepening knowledge (step 2), knowledge creation (step 3) and develop computational thinking (Grover & Pea, 2013; Minichiello, 2014) as a new literacy that uses the process of abstraction, automation and problem solving (Qin, 2009; Wing, 2006). Computational thinking (CT) is a way to develop new thinking strategies to analyze, identify and organize relatively complex and ill-defined tasks (Rourke & Sweller, 2009). Computational thinking is a “set of cognitive and metacognitive strategies linked to the knowledge and process modelling” such the identification, decomposition and structural organisation of items into logical sequences, the capacity of abstraction, pattern identification and the understanding and creation of algorithms (Romero, 2016, p. 4). Tchounikine (2016) analyze the CT concept from a computer science perspective and propose to define it as the set of "concepts and process which are explicitly related to computer science" (p.2). He highlights the importance of the algorithm concept within CT and discusses its relation to the concept of programming.

In order to develop CT from elementary education, some educational initiatives have called for establishing programming as a compulsory activity. Estonia, France and UK are among some of the countries that have introduced programming in their curricula. These initiatives rely on the potential of programming to mobilize the different CT strategies and, in some cases, introduce assessments tools for the CT skill (e.g. code.org). While programming has a potential to develop the CT skills, the way the programming activities are proposed to the learners shows a high diversity in the level of engagement in a creative programming approach. In some cases, programming has been approached as a procedural task, with little pedagogical interest beyond learning to code. Some authors has been very critical to the introduction to programming in schools because they consider initiatives that has been focused on learning to code as an objective per se, "motivated by a shortage of programmers and software developers in industry, focus especially on preparing students for computer science degrees and careers, and they typically introduce coding as a series of logic puzzles for students to solve" (Resnick & Siegel, 2015, para. 2). The authors of this criticism are no less than two of the founders of the visual programming tool Scratch, working at the Massachusetts Institute of Technology (MIT). The Resnick and Siegel is not about learning to code per se as a technical tool, but how coding is a creative tool, a "new type of literacy and personal expression, valuable for everyone, much like learning to write. We see coding as a new way for people to organize, express, and share their ideas" (para. 4). In this paper, the type of programming we are referring to goes beyond the simple teaching of codes and the memorization of these codes; we favor teaching coding as a participatory process that could be, in and by itself, a learning tool or a "mindtool" in terms of Jonassen (1996). Creative programming goes beyond the consumer approach of technology and coding. We also argue that coding could be used to (re)assess intergenerational learning. With the variety of readily available tiny and affordable computers that can be dedicated to projects (e.g. Rasperry Pi), Open-source software and programming tools (e.g. Scratch, Blockly, etc.), the act of programming not only reaches unprecedented levels today, but is also presenting occasions for studying innovative collaborative and progressive pedagogies and develop a new relation to technologies as creative agents.

A growing number of countries are introducing computer programming education courses in the schools. In Europe, twelve countries have already integrated programming in the curriculum, including Estonia and the United Kingdom, and seven are in the process of integrating it, such as France and Finland. In the US, the initiative Hour of Code (#hourofcode, #heureducode) has a growing popularity at the national and international level, with more than 9 million of registered users worldwide. The Hour of Code self-describes the initiative as "is a global movement reaching tens of millions of students in 180+ countries. Anyone, anywhere can organize an Hour of Code event. One-hour tutorials are available in over 40 languages" (Code.org, 2015). The website Hourofcode.com offers tutorials mainly based on a step-by-step procedural learning approach to programming. Like Scratch and other visual programming interfaces, the Hour of Code programming is done by dragging and dropping pieces of codes in a jigsaw puzzle.

In Quebec, the number of declared Hour of Code events were inferior to other Canadian provinces and other American and European countries (Romero, 2015). However, there are an increasing number of innovative teachers and organizations, such the Squeaki RÉCIT team and Kids Code Jeunesse who are actively engaged towards the integration of programming at school.

From procedural learning to code to creative programming

Programming is a knowledge modeling tool (Jonassen, Strobel, & Gottdenker, 2005) with a huge creative, cognitive (Lajoie, & Derry, 1993) and metacognitive potential (Azevedo, 2005). However, like any other technology, it must be pedagogically integrated in the classroom activities as a mindtool, and not only a technical tool, to deploy its potential. While some uses of technologies engages the learner in a passive or interactive situation where there is no little room for knowledge creation, other uses engages the learner in a creative knowledge building process where the technology aims to enhance the cocreative learning process (Romero, Laferriere, & Power, 2016). As shown in the figure below, we distinguish five levels of creative engagement in computer programming education education according to the creative learner engagement in the learning to program activity: (1) passive exposure to teacher-centered explanations, videos or tutorials on programming; (2) procedural -step-by-step- programming; creating new medias through individual programming (3) or team-based programming (4), and finally, (5) participatory co-creation of knowledge through programming.

INSERT FIGURE 1

Fig 1. Five levels of learning to code activities and examples.

We refer to these five components as levels because they corresponds to a progression on a scale related to the learner engagement on the activity —from teacher-centered activity to student-centered activity, or even better from teacher-centered activity to student co-creation of knowledge. These five levels are in this study applied to the learning to code activities but could be considered in any

The first two steps aims to learn to code, where learning to code is an objective per se and is decontextualized from the curriculum. While they might be seen as low levels of student engagement or too didactic because they focus on the activity of the teacher, for learners who are not familiar with programming language, they are essential steps. In fact, these two first steps are probably one of the only ways to learn the basics of programming before engaging the learners in the (co)creative programming activities of the last three levels. Next, we introduce each of the five levels with examples.

Level 1. Passive exposure to programming lectures or resources is the lowest level of creative engagement of the learner in computer programming education. This transmissive approach of the lectures given by the teacher or the readings or videos displayed allows to transmit

information, but the learner is not engaged in any type of interaction. This level has been described under the term Web 1.0.
Level 2. Procedural (step-by-step) programming engages the learner in a procedural approach similar to a 'code recipe' or step-by-step construction manual (e.g. code.org/flappy). Despite that this approach shows a limited educational value and is decontextualized from the curriculum, procedural learning programming could sometimes be an easy first step before engaging the learners in creative programming activities pegged to the curriculum (Romero & Lambropoulos, 2015).

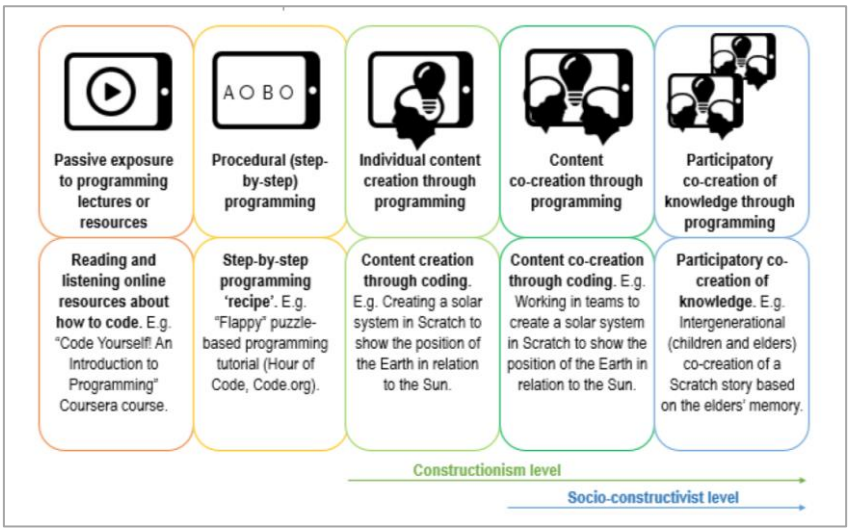
Level 3. Individual content (co)creation through programming engages the learner in an individual creative activity, where he should develop an original solution. For example, learners could be individually engaged in creating a solar system in Scratch to show the position of the Earth in relation to the Sun.
Level 4. Team-based content (co)creation through programming engages a group of learners in a collaborative creative activity, where he should develop an original solution.
Level 5. Participatory co-creation of knowledge through programming engages not only a group of learners but also persons from outside the classroom (other pupils in the school, family members, etc.) in a participatory process where decision are supposed to be democratic. In democracy, all users are engaged in the creation process which allows to take advantage of the benefits of the inclusive design (Clarkson, Coleman, Keates, & Lebbon, 2013).

In the last three levels, the creative programming engages the learner in the process of designing and developing an original work through coding. In this approach, learners are encouraged to use the program as a knowledge co-construction tool. For example, they can (co)create the history of their city at a given historical period or transpose a traditional story in a visual programming tool like Scratch (https://scratch.mit.edu/). In such activities, learners must use skills and knowledge in mathematics (measurement, geometry and Cartesian plane to locate and move their characters, objects and scenery), Science and Technology (universe of hardware, transformations, etc.), language (narrative patterns, etc.) and social studies (organization in time and space, companies and territories). It is through these three highest levels of programming that the development of new ideas is possible.

Code learning does not replace or dilute the time allowed in class for the traditional disciplines ; instead, it offers an interdisciplinary development opportunity when its integration is situated within the three last levels. Despite the pedagogical relevance of those last three approaches, it still is easier to find educational resources located in the first two levels ; code.org is especially ripe with pedagogical material. The learning to code per se is mostly decontextualized from the curriculum, which makes it easier to create global educational resources to learn to program. These limits can be overcome by the teachers if they integrate the learning to code activities within an educational situation where pre and post activities allows the development of learning activities that are more socio-constructivist, creative and curriculum oriented.

The creative programming helps develop computational thinking and enhance 21st century skills, including (co)creativity and problem solving. In matters related to science, technology, engineering and mathematics (STEM), it was observed that students with learning difficulties were more committed when they were engaged in digital game activities and programming of robots (Yasar, Maliekal, Little, & Jones, 2006). In addition, these activities provide the opportunity to develop computational thinking through the programming and are of crucial importance if we want to help reduce inequalities between girls and boys face of scientific and technological careers. As citizens with increasingly important and pervasive digital lives, thinking critically about ICTs is crucial. Developing skills in computational thinking and coding appears to be a valuable learning strategy to introduce questions about our usage of technologies and their limitations. Through the process of understanding coding, learners emerge as more active and creative in their society; no longer only consumer of media, they are informed citizens making informed ICTs choices where they better comprehend the ramifications.

4.1. FIGURE



5. REFERENCES

Azevedo, R. (2005). Computer environments as metacognitive tools for enhancing learning. Educational Psychologist, 40(4), 193–197.

Clarkson, P. J., Coleman, R., Keates, S., & Lebbon, C. (2013). Inclusive design: design for the whole population. Springer Science & Business Media.

Code.org. (2015). Hour of Code. Récupéré du site https://hourofcode.com/me/fr

Lajoie, S. P., & Derry, S. J. (Eds.). (1993). Computers as cognitive tools. Routledge.

Griffin, P., McGaw, B., & Care, E. (2012). Assessment and teaching of 21st century skills. Springer.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12. A Review of the State of the Field. Educational Researcher, 42(1), 38–43.

Jonassen, D. H. (1996). Computers in the classroom: Mindtools for critical thinking. Prentice-Hall, Inc..

Jonassen, D., Strobel, J., & Gottdenker, J. (2005). Model building for conceptual change. Interactive Learning Environments, 13(1-2), 15-37.

Katz, I. R. (2013). Testing information literacy in digital environments: ETS's iSkills assessment. Information Technology and Libraries, 26(3), 3–12.

Minichiello, F. (2014). L'enseignement du code à l'école. Revue internationale d'éducation de Sèvres, (67), 12–16.
Qin, H. (2009). Teaching computational thinking through bioinformatics to biology students. In ACM SIGCSE Bulletin (Vol. 41, pp. 188–191). ACM.

Resnick, M., & Siegel, D. (2015, November 10). A Different Approach to Coding. Retrieved from https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.b9jcw2js5

Romero, M. (2015). L'heure du code à Québec: introduction de la programmation à l'école. Québec Numérique. Retrived from : http://www.quebecnumerique.com/lheure-du-code-a-quebec-introduction-de-la-programmation-a-lecole/

Romero, M. (2016). Cinq compétences pour le 21e siècle. In Romero, M., & Vallerand, V. (eds.). Guide d'activités technocréatives pour les enfants du 21e siècle. Québec, QC: Livres en ligne du CRIRES. Retrived from : http://lel.crires.ulaval.ca/oeuvre/guide-dactivites-technocreatives-pour-les-enfants-du-21e-siecle

Romero, M., Hyvönen, P., & Barberà, E. (2012). Creativity in Collaborative Learning across the Life Span. Creative Education, 3(4), 0–0.

Romero, M., Laferriere, T., & Power, T. M. (2016). The Move is On! From the Passive Multimedia Learner to the Engaged Co-creator. eLearn, 2016(3), 1.

Romero, M., & Lambropoulos, N. (2015). Digital game creation as a creative learning activity. In Interactive Mobile Communication Technologies and Learning (IMCL), 2015 International Conference on (pp. 338–342). IEEE.

Rourke, A., & Sweller, J. (2009). The worked-example effect using ill-defined problems: Learning to recognise designers' styles. Learning and Instruction, 19(2), 185-199.

Tchounikine, P. (2016) Initier les élèves à la pensée informatique et à la programmation avec Scratch. Retreived from http://lig-membres.imag.fr/tchounikine/PenseeInformatiqueEcole.html

UNESCO. (2011). TIC UNESCO: un référentiel de compétences pour les enseignants. Paris: UNESCO. Retrieved from http://unesdoc.unesco.org/images/0021/002169/216910f.pdf

Voogt, J., & Roblin, N. P. (2012). A comparative analysis of international frameworks for 21st century competences: Implications for national curriculum policies. Journal of Curriculum Studies, 44(3), 299–321.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.
Yasar, O., Maliekal, J., Little, L. J., & Jones, D. (2006). A computational technology approach to education. Computing in Science & Engineering, 8(3), 76–81.