Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study

Jennifer Jenson[1] and Milena Droumeva[2]
[1]York University, Canada
[2]Simon Fraser University, Vancover, Canada
jjenson@edu.yorku.ca
milenadroumeva@gmail.com

Abstract: While advances in game-based learning are already transforming educative practices globally, with tech giants like Microsoft, Apple and Google taking notice and investing in educational game initiatives, there is a concurrent and critically important development that focuses on 'game construction' pedagogy as a vehicle for enhancing computational literacy in middle and high school students. Essentially, game construction-based curriculum takes the central question "do children learn from playing games" to the next stage by asking "(what) can children learn from constructing games?" Founded on Seymour Papert's constructionist learning model, and developed over nearly two decades, there is compelling evidence that game construction can increase student confidence and build their capacity towards ongoing computing science involvement and other STEM subjects. Our study adds to the growing body of literature on school-based game construction through comprehensive empirical methodology and evidence-based guidelines for curriculum design. There is still debate as to the utility of different software tools for game construction, models of scaffolding knowledge, and evaluation of learning outcomes and knowledge transfer. In this paper, we present a study we conducted in a classroom environment with three groups of grade 6 students (60+ students) using Game Maker to construct their own games. Based on a quantitative analysis and a qualitative discussion we organize results around several core themes that speak to the field of inquiry: levels of computational literacy based on pre- and post-tests; gender-based attitutdes to computing science and programming based on a pre- and post-survey; and the relationship between existing media literacy and performance in programming as part of the game construction curriculum. Significant results include some gender differences in attitudes towards computers and programming with boys demonstrating slightly higher confidence and performance. We discuss the complex reasons potentially contributing to that, particulaly against a diverse ecology of overal media use, gameplay experience and access to technology at home. Finally, we theorize game construction as an educational tool that directly engages foundational literacy and numeracy, and connects to wider STEM-oriented learning objectives in ways that can benefit both boys and girls in the classroom.

Keywords: game making, STEM, coding, Game Maker, digital literacy

1 In pursuit of "21st century skills"

An ongoing challenge of the 21st century is ensuring everyone has the requisite skills to participate in a digital, knowledge-based economy. This is increasingly difficult under conditions of austerity in both K-12 and higher education, at a time when there is significant need for skilled labour in technology and computing fields in particular. Despite widespread enthusiasm for "21st century learning," researchers and policy makers around the globe are still trying to articulate exactly what constitutes this term (Media Awareness Network 2010), while public education generally is being criticized for not adopting it (Francis 2012; Lynch 2013). There is, for example, no specific curriculum provision regarding what 21st century learning should entail and how that should inform K-12 schooling, though there is widespread and growing agreement that digital games figure somewhere in that landscape (Gee 2005; Salen 2007; Squire 2011). Digital games are increasingly at the forefront of conversations about ways to address student disengagement (Gee 2003; Rieber, LP 1996; Rupp et al. 2010) and ways to foster 21st century learning and skills (Barab & Dede 2007; Steinkuehler 2008; Squire 2011). That research concentrates on *playing* digital games, whether those are commercially made or made especially for education. Less prominent has been research focused on the *design and development* of games as a means to support critical competencies like creative problem solving, collaboration, and programming skills (Carbonaro et al. 2010; Denner 2011; Denner & Wenner 2007; Papert 1993). Designing and making digital games, this prior research suggests, can provide an ideal framework for operationalizing 21st century learning: creating digital artifacts entails technical, computational and aesthetic forms of activity whose success depends on bridging between arts and sciences—an intersection increasingly characteristic of the contemporary job market *and* effective participation in social life.

Reference this paper as Jenson J and Droumeva M, "Exploring Media Literacy and Computational Thinking: A Game Maker Curriculum Study" *The Electronic Journal of e-Learning Volume 14 Issue 2 2016*, (pp111-121) available online at www.ejel.org

One of the main motivations for bringing game design and development into the fold of STEM curriculum planning concerns the need to introduce and familiarize youth to the principles of computation, design thinking and procedural logic, from an earlier age than is currently practised. The context for this is a growing acknowledgement among educational researchers, computer scientists and teachers that 'computational thinking' and algorithmic logic ought to be considered a kind of 'core literacy' that needs to be incorporated into the school curriculum alongside numeracy, textual literacy and scientific thinking (diSessa 2000; Wing 2006). Computational thinking can also be located alongside a range of other competence-based technological 'literacies' discussed in popular education blogs, that include 'making' or tech prototyping, fostering of applied 'creativity,' as well as 'design thinking'. While Papert's work in the 1980s saw the emergence of the first user-oriented *Logo* coding language developed specifically with educational goals in mind, it has only been in the last five to ten years that a plethora of drag-and-drop programming environments for children have become readily and easily available. During that time there have been numerous improvements to the user interface and functionality of these programs, targeting specific age groups and in many cases making tools available on the web as part of online sharing communities of practice[1]. One of the most pertinent underpinnings of contemporary education research into using game construction software in the classroom is addressing the systemic problem of girls' impoverished representation in computing science and technical fields. Such studies aim to deliberately engage girls and other marginalized youth groups in coding activities, and counter negative associations and lack of confidence that might hold them back from approaching and benefiting from ongoing computer programming instruction. A central pedagogical concern with regard to teaching game construction as an entry-level form of computer programming centres around defining and operationalizing "computational thinking" as a core curricular concept, and identifying how and when to introduce it into the classroom. Additional concerns involve what type of instruction is required and which tools are best suited to achieve these cognitive objectives.

2 Definition and 'cognitive objectives' of computational thinking (CT)

Wing (2006) defines CT as "reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation." Yadav et al. (2014) define CT as a "mental activity for abstracting problems and formulating solutions that can be automated" while Cuny et al. (2010) define it as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent". According to Denner, Werner and Ortiz (2011), "algorithmic thinking involves defining a problem, breaking it into smaller yet solvable parts, and identifying the steps for solving the problem." As part of this, students must model the essential characteristics of the problem while suppressing unnecessary details. In the process, "finite sequences of instructions are coded to operationalize the modeled abstractions." From a review of the field in Grover and Pea (2013), the following is a standard list of learning objectives or computational constructs that ought to be covered in some form in instructional designs of entry-level computing:

- Abstractions and pattern generalizations (including models and simulations)
- Systematic processing of information (proceduralization)
- Symbol systems and representations
- Structured problem decomposition (modularizing)
- Iterative, recursive, and parallel thinking
- Conditional logic
- Debugging and systematic error detection (pp. 39-40)

A major challenge here is translating these constructs to both affordances of existing game development tools and to specific instructional designs and learning objectives. A number of contemporary studies, for instance, have published extensive breakdowns of tool-specific available actions, modifications, as well as procedural and conditional logic sequences that correspond to top-level computational constructs (Carbonaro et al. 2010; Denner 2011; Denner & Wenner 2007). However, less explicit are the particular pedagogical underpinnings of instructional design by which game construction is introduced and implemented in the classroom within the larger context of mathematics and science (STEM) instruction. Specifically, some of the concerns that need to

[1] Arguably two of the oldest and most widely used drag-and-drop coding environments for early school education are Scratch https://scratch.mit.edu/ and Alice http://www.alice.org/

addressed include the scaffolding, assessment and transfer of both computational terminology and applied coding skills.

3 Assessment, scaffolding and transfer of conceptual skills

A standard approach in establishing the efficacy of a particular curricular program is using a pre- and post-test. Additional study measures include implementing different scaffolding designs (e.g. written materials or direct instruction prior to game construction activity), and exercises that specifically evaluate domain transfer and cross-domain transfer of particular learning content or skill. Depending on the research objectives related to computational thinking instruction (CT) different studies adopt different pre- and post-test measures. For instance, given existing evidence that programming performance is related to confidence and attitudes to computing science (CS), there are several instruments that specifically test (using Likert-scale questions) confidence and attitude constructs related to CS (Hoegh & Moskal 2009; Heersink & Moskal 2010). Additionally, Seaborn and colleagues (2012) implement a programming-specific pre/post-test that not only evaluates domain-level knowledge related to computational terminology, but also responses to semantically correct programming language (commands written in programming code). A large part of assessment is of course analyzing and evaluating the game artefacts that students create in terms of complexity and the incorporation of specific computational elements. The problem of assessing the transfer of computational knowledge was addressed as early as 1988 (Klahr and Carver). Werner, Denner and Campe (2012) propose a de-bugging game as a gamified form of assessment that not only looks at correct solutions, but the process of troubleshooting and alternative approaches.

Assessment is also a function of the study design including the overall timeframe of instruction and scaffolding activities (Lye & Koh 2014). In this sense, and given the practical difficulties of securing 'classroom time' as part of regular school curricula, there is quite a lot of variation in the structure and framing, as well as choice of programming environment, in educational research that takes up game construction as a way of teaching computational thinking (CT). Examples of this include Carbonaro et al. (2010), who used ScriptEase, a module-based[2] drag-and-drop game construction program, for two 6-hr direct instruction workshops at University of Alberta, in addition to 6 more hours at school for kids to finish their games. Denner, Werner and Ortiz (2011) worked with girls in an after-school club setting for over 14 months (1-2hrs a week), designing a series of six different genre games in Stagecast Creator, another module-based drag-and-drop environment. Seaborn et al. (2012) adopted the structure of a 'design camp' utilizing Game Maker with high school teams in six modules each lasting several months; their study measured self-efficacy, perception of helpfulness of classroom activities, and understanding of computational concepts.

4 Teaching coding with Game Maker: A case study

This project is a research-based challenge to the now widely questioned but surprisingly persistent presumption that students in today's classrooms are all by default 'digitally native' (Prensky 2001), and that those 'digitally native' children are learning just by playing digital games (Prensky 2005). In actuality, 'digital nativity' is looking markedly gendered, raced and classed (de Castell, Boschman & Jenson 2008), and the educational use of digital games to advance 21st century learning is turning out to demand a lot more than just playing them. Just being *familiar* with digital technologies and using them in one's everyday life does not necessarily translate into skillfully using them for learning (Livingstone, 2010). This study recognizes that in the contemporary media landscape, familiarity with digital gameplay can represent for many young people their entry point into acquiring the foundational digital skills demanded by a global knowledge economy. It builds on the familiar game medium as a 'gateway' to study the development of critical digital literacies not through digital game play on its own, but through a 'production pedagogy' in which gameplay is integrally co-engaged with the design and development of digital games.

One of the aims of this study is to investigate the question of whether gameplay experience and general media literacy in childrens' lives relate to their ability to participate and benefit from game construction activities in the classroom. This question is inextricably linked to the larger context of STEM instruction and in that sense this study will contribute to research on game design as a 'gateway' to STEM. This might, moreover, be a way to effectively *re-fuse* the digital divide which the survey will document and track for the duration of the project. Finally, we set out to explore, beyond simply celebrating the introduction of game construction in the

[2] Typical interactions, settings and game mechanics are pre-programmed in the environment; they are selected as drag-and-drop elements and customized to fit a game-specific situation and purpose.

classroom, specific instructional designs that can help and support students, not only in overcoming confidence-related barriers to entry into computing science later in their education, but also in supporting and supplementing their grade-specific STEM knowledge through its application in the domain of game-making.

4.1 Study Design

This study took place in a very large elementary school (with over 750 children) in Ontario, Canada. Ontario does not currently have any mandatory computer science related curricula at the grade 6 level. We chose to work with Grade 6 students as much of the work done previously (see Carbonaro et al. 2010; Denner 2011) suggests that grades 6 to 7 is the time many students begin to make choices about what courses they will take at the high school level and beyond. Because there is currently no equivalent curriculum in Ontario, we had to negotiate classroom time with the participating principal and teachers, meaning that in this case we used time that otherwise would have been designated for Language Arts. Our rationale for this is that we were concentrating on learning a new piece of software that also meant students learned new vocabulary and new concepts related to programming. In the end, we were able to negotiate working with the full grade 6 complement in the school (3 classes, 67 students), replacing their curriculum for a period of 1.5 hours over 6 consecutive days, in addition to a full day of curricular programming in a fieldtrip to a local university. In total, the participating students had approximately 15 hours using Game Maker and of that, approximately 4-5 hours were direct instruction. Nearly all students worked in pairs to create their games. Peer-based programming instruction has been shown in previous studies to be positively correlated with the retention and application of new material (Peppler & Kafai 2007). We also wanted to scaffold peer support for students so that they did not have to rely on the researchers to answer questions and to help move their games along.

4.2 Operationalizing Computational Constructs

In order to create a usable instructional design for grade-appropriate computational literacy curriculum we had to translate higher-level frameworks of computational thinking such as 'decomposition,' 'parallelism' and 'abstractions and pattern generalization' constructs into operational computer science vocabulary and operations. In particular, amidst increasing critiques of drag-and-drop game design as a form of computational literacy instruction (Duncan, Bell & Tanimoto 2014) we wanted to depart from bottom-up 'sandbox' environments such as *Scratch* or *Alice* and attempt grade-appropriate instruction directly using code-window semantic programming. Since *Game Maker Studio* provides both drag-and-drop and semantic coding (though, arguably it is skewed towards coding) we landed on using this tool as one of the more versatile products that offer low/mid-entry and high ceiling opportunities for game development. It is a tool that relates more transparently to computational constructs and the practice of object-oriented programming, and can be adapted for computational instruction at a variety of (upper) grade levels as well. The following table represents our instructional framework across the specific software domain of Game Maker as they link to higher-level computational constructs and vocabulary.

CT constructs	Definition / Domain knowledge	Game Maker syntax examples	Computational Vocabulary
Variables	Containers for storing values so that values can be used and modified in other parts of the program	**direction = 180;** **speed = 4;**	Variable, value, object, instantiation, syntax, rate of movement, direction
Operations	Mathematical operations with variables or other parts of the program that cause game state changes	**score = score + 1;** **x = x – 7;**	Mathematical operations, Cartesian (x/y) coordinates, syntax
Functions	Built-in computational objects, modifiable constructs that cause specific game actions and state changes	**instance_destroy();** **move_bounce_solid(false);**	Function, Boolean logic (true/false), syntax, attributes, parameters, nested operations, placeholder
Conditionals	Statements that evaluate a game state and cause other game actions, operations, variable changes etc. to take place	**if place_meeting (x, y + 1)** **{** **gravity = 0.01;** **}** **else** **{** **gravity = 0;** **}**	Boolean evaluation (if/then/else), conditional logic, branching and nesting, truth value, queries

Table 1. Computational instruction framework

4.3 Instructional design: scaffolding and facilitation

In previous iterations of this type of study – game design camps with grade school students – we typically relied on heavy facilitation and one-on-one work with students or project peer groups to help students complete a functional and polished version of their game idea (see Fisher & Jenson, Forthcoming). We still used *Game Maker*, however our curriculum was based on drag-and-drop commands and the instructional format was after-school clubs rather than classroom-based instruction. The system of instruction we used formerly comprised of one extensive follow-along tutorial of Game Maker's sandbox game the *Brick Breaker*[3] followed by unstructured game design time for kids to work on their own game ideas. For this iteration of research, particularly given that it took up valuable classroom time, we scaffolded coding instruction with a series of direct follow-along lessons where participants learned new vocabulary and practiced applying new programming constructs in appropriate chunks of material. The rest of each session was spent working on their own game, adapting and modifying elements that were just covered in their own design. Key to our curriculum design in this study was the pacing of instruction and material, and the incremental introduction of computational concepts. First we introduced the concept and application of variables, then the role and syntax of operations, followed by the concept and use of functions (including in-depth self-help strategies using Game Maker's reference guide for game programming). Finally we introduced the syntax and function of conditional statements, all the while reinforcing previously learned vocabulary, reiterating the logic and relationship between game events and game actions (input and output).

As part of the curriculum we built in the option for participants to look into several developed example games and copy and adapt code from them as a kind of 'ecological' approach to coding instruction, given that copying and adapting code is foundational to efficiency in programming in the workplace (Duncan, Bell & Tanimoto 2014). In this much more structured and scaffolded game construction curriculum, our vision was of research facilitators assisting with software/interface issues (since Game Maker has a bit of a learning curve), and helping to guide participants in design and programmatic challenges through case-by-case directed instruction, rather than dictating or writing code for them. To enable this model of self-directed learning we enforced an "Ask 3 before you ask Me" rule where kids had to look up a question they had in the Game Maker help, or ask a peer, before they turned to a research facilitator.

4.4 Data Collection

Prior to the study, every participant was given a media literacy and attitudes questionnaire, as well as a pre-test designed to evaluate their existing knowledge of computer science concepts such as variables, operations and functions. Following the study, in order to determine what if any attitudinal changes might have occurred, a post-test was administered that was identical to the pre-test, and a short questionnaire that repeated the same attitudinal questions from the medial literacy and attitudes questionnaire. In addition, daily field notes were taken by at least two researchers who were on hand for the duration of the study, that included short video clips and photos as students worked on their games. To capture the progress that participants were making daily as well as to gauge how much and what type of help participants were receiving from researchers, we used Chronolapse, a software that records an image of the computer's screen along with a webcam image every 15 seconds. In total, we generated 256 Chronolapse videos of approximately 1.5 hrs duration for each and recorded 36 qualitative fieldnotes of each classroom session day.

5 Results and Discussion

Given that the study and data collection are still ongoing, in this paper we report preliminary results in relation to areas of critical discussion related to the issues raised earlier. As well, we discuss some preliminary correlations, which are evaluated on an ongoing basis with a statistical analysis using the chi-square test of independence for binomial variables and paired t-tests for continuous variables. Overall, the classroom-based instructional model seemed to function well for grade 6 students working in pairs, who were able to create playable complete games using Game Maker within the 6 classroom sessions + 1 extended university-based field trip. Not only did students design and code their games with minimal facilitation, but their content knowledge of basic computational terminology, as well as Game Maker domain knowledge, improved from an average of 6.7 to an average of 9.3 (out of 16). In the following sub-sections we discuss additional preliminary data organized around several critical areas: assumptions about pre-existing video game competence and computer-based (and computational) knowledge; the relationship between gender, confidence, and attitudes

[3] http://sandbox.yoyogames.com/games/120704

towards computer programming instruction; and preliminary results about gender differences in computer programming performance in the context of game construction. Finally we comment on some initial impressions related to the study design, facilitation and classroom-based context utilized here.

Figure 1. Classroom set up and kids working on game design and game programming

5.1 Digital Know How: Surveying Media Use and Playing Games

In terms of general media use (based off the media questionnaire), we made sure to cover a wide array of media practices and technologies that students might use: personal mobile technologies; different types of computers; gaming devices; broadcast media; social media portals; game genres; and electronic communication platforms. The results are as follows: boys and girls are similarly likely to use social media, and almost everyone reported YouTube as one of their top websites/social media sites to visit at home, closely followed by Facebook and Instagram. Girls are slightly more likely to report that they use online communication tools such as Skype or FaceTime, as well as more likely than boys to frequent the micro-blogging platform Tumblr (non-significant result). While most respondents reported regularly playing videogames, boys are significantly more likely to play online multiplayer and high-end console games, and girls a bit more likely to play puzzle or role-playing games on the Wii platform, have access to tablets, and play mobile games. In terms of significance, a chi-square test of independence indicated that the relationship between gender and daily game console use was significant, with boys much more likely to use gaming consoles, X^2 (1, N=65) = 10.76, $p < .01$. Some of these gender differences in access to and use of computers and gaming consoles likely speaks to cultural advertising that targets boys for high-end consoles such as XBOX and PlayStation, while establishing a wider audience for 'educational' tools such as the Wii Series or the iPad[4]. Of the participants who reported that they did not own gaming platforms or played games, the majority were girls. These nuanced gender differences were also noted by researchers in the classroom during instruction, with boys much more likely to raise their hand to answer questions related to gaming and computers, and girls much more likely to volunteer answering questions about mathematics, language and other STEM content that figured into computational concepts.

In terms of device use at home, which relates to overall media literacy and technological ease and competence, there was only one statistically significant difference in the relationship between gender and daily laptop use with a chi square of X^2 (1, N=65) = 4.53, $p < .03$, indicating that female students reported that they were more likely than male students to use a laptop daily at home. This means that the majority of access and frequency of use of everyday technologies such as broadcast media, personal devices, popular websites and social media, are more or less equally distributed among male and female students, with the more interesting differences being in the content consumed/interacted with: more high-end, strategy and virtual reality-based games in the case of boys, and more versatile, casual puzzle and motion-based games in the case of girls. Potentially related to these statistics of media and video game use (however, non-statistically significant in this study), girls had a slightly lower average score on the computational literacy pre-test compared to boys, and boys' post-test scores improved significantly more than girls' scores (see Figure 2). That said, a more nuanced look into the data reveals that girls tended to have more consistent average scores,

[4] http://www.edutopia.org/blog/ipad-teaching-learning-apps-ben-johnson

whereas boys' competence was split between those who had very little knowledge and those who had extensive prior experience with computing and gaming.

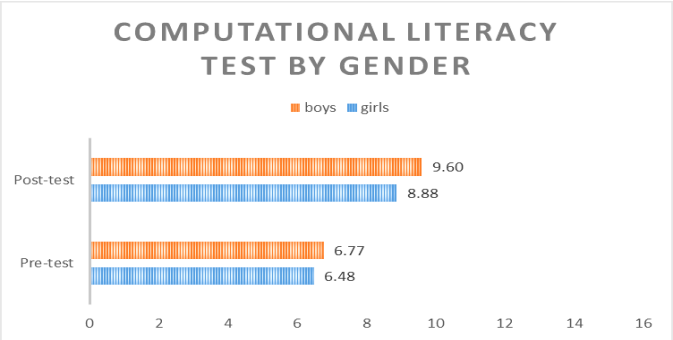


Figure 2. Computational literacy pre- and post-test results by gender

What is interesting here in terms of the oft-assumed relationship between an allegedly tech-savvy generation and an aptitude for computer science, is that reported frequent gameplay activity or specific media technology use did not correlate significantly with either a high score on the computational knowledge pre-test, or with an overall high confidence about using computers and learning programming. A few exceptions, annecdotally, are that playing Minecraft seemed to correlate with higher pre-test scores and overall higher confidence about using/learning about computers; not playing games at all correlated with lower confidence about using and learning about computers and lower pre-test scores. This context both confirms and questions some of the game-based learning assumptions pointed out in past work – namely, that playing video games creates base computational knowledge and confidence about technical computer skills. Our work suggests that playing video games is one important ingredient to creating the conditions for computer programming instruction and computational literacy, however, it can also be said that access to and encouragement to play games and use technology is a core part of home and family socialization, bringing us back to the 'origins' question of gender-based differences in technology performance.

Clearly, there is more at play here. For instance, when we look at performance on the computational knowledge pre-test, on average, kids who had higher scores reported less confidence about their ability to learn new computer programs, computational concepts, and troubleshoot computer programs. Conversely, kids who displayed some of the lowest pre-test scores reported some of the highest confidence about working with computers and being able to use and learn new computer programs. This finding is of critical pedagogical importance because it suggests that procedural 'content' knowledge about mathematics and computer programming does not necessarily translate into confidence or perceived ability to contend with computational instruction. There are more issues here that warrant further research to better our understanding, specifically around children using computers and engaging in gaming and social media, and if and how these activities support "learning."

5.2 Gender-based Attitudes to CS in Computer Programming Instruction

To develop and implement a school-wide computational literacy program based in game construction, it is necessary to first examine and understand some of the underlying context of STEM education at the grade 6 level, as well as some of the persistent gender differences in confidence and preparedness in relation to computer work in general. Confidence and attitudes has already been linked in numerous studies (Carbonaro et al. 2010) to actual classroom performance and the ability to learn computer programming, as well as the motivation to continue on this educational track. Given this, an important part of the pre-study questionnaire was gauging self-reported confidence around using computers and learning computer programming, including potentially 'gendered' attitudes towards computational knowledge in general. Results collected so far suggest that while both sexes think the other is worse at computer programming, boys were significantly more likely to assess girls' computer skills as low, whereas girls had mixed evaluations of boys' capabilities in programming. This trend translates into self-reported attitudes and confidence with regard to computer skills in general and one's capacity to learn programming (see Figure 3). Girls consistently scored lower in confidence levels than boys, and in particular, they scored significantly lower on confidence in their abilities to troubleshoot computer programs as well as general self-confidence when it comes to computer programming (but not computer use).

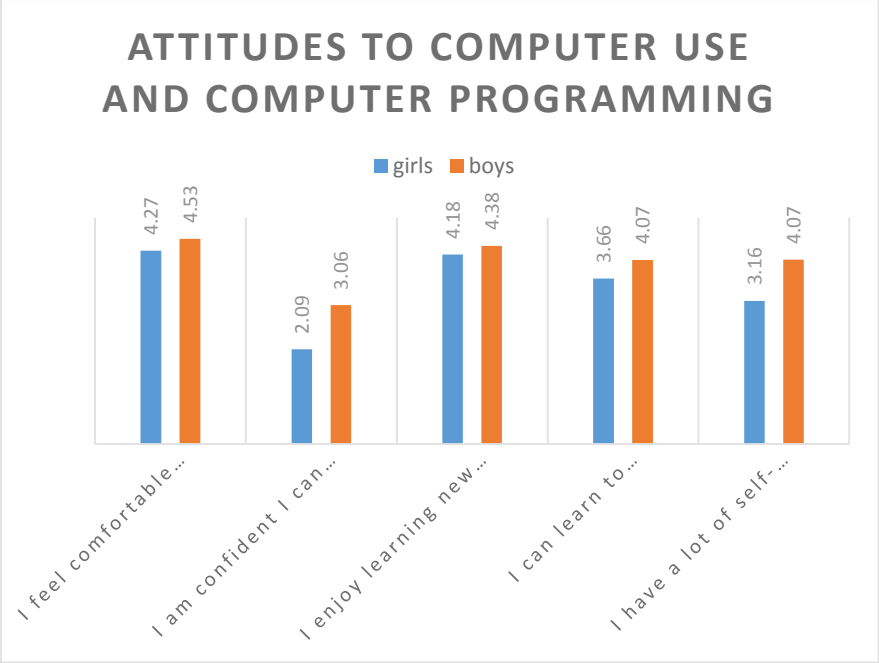
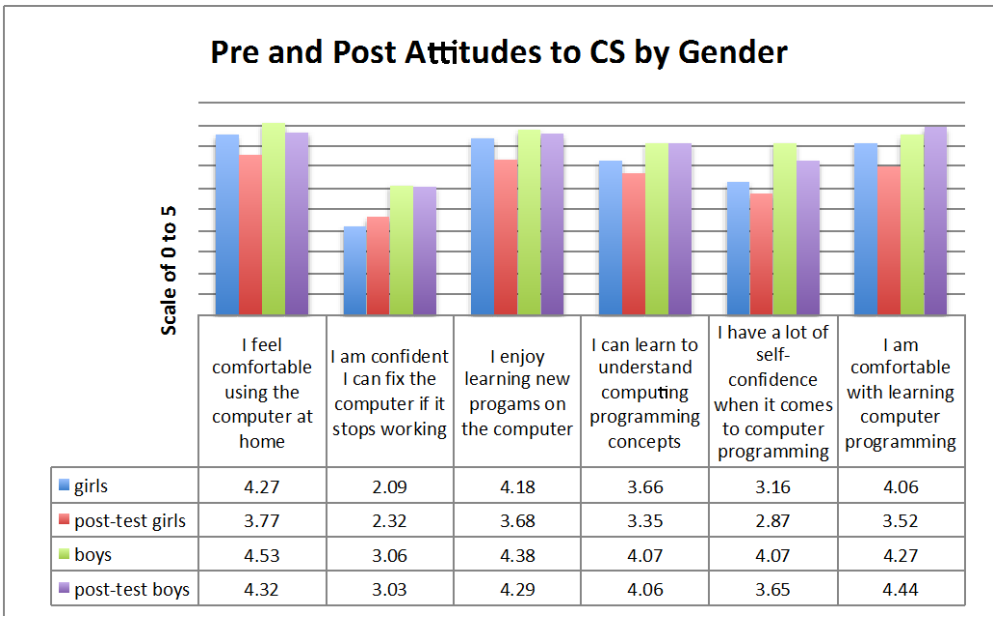


Figure 3. Attitudes to computer use and computer programming by gender

Some of the positive findings around attitudes to computer science were that neither girls nor boys reported any social stigma for 'being good with computers', and on the whole everyone gave positive scores to the idea of studying computer programming at school, being interested in computers, pursuing computer science further, and having a future career that includes computer work and coding. In terms of pedagogical efficacy, we wanted to ask students what they thought can be learned from games and from programming in the classroom. When asked why they thought computer programming would be good to teach at school, most kids emphasized experiential learning, trial and error, and learning about computers. Girls were significantly more likely to list 'collaborating with others' and 'learning from experience' which might suggest both an apetite for applied learning (especially if they are not exposed to it at home) and valuing traditionally 'feminized' qualities of work such as collaboration. Most kids listed 'making learning fun' and 'learning specific content' but very few ticked 'learning about logic', which might indicate a gap in terms of what students understand computer programming instruction to be outside or inside a school setting.



	I feel comfortable using the computer at home	I am confident I can fix the computer if it stops working	I enjoy learning new progams on the computer	I can learn to understand computing programming concepts	I have a lot of self-confidence when it comes to computer programming	I am comfortable with learning computer programming
girls	4.27	2.09	4.18	3.66	3.16	4.06
post-test girls	3.77	2.32	3.68	3.35	2.87	3.52
boys	4.53	3.06	4.38	4.07	4.07	4.27
post-test boys	4.32	3.03	4.29	4.06	3.65	4.44

Figure 4. Attitudes to computer use and computer programming by gender: pre and post test

The post-survey on attitudes to CS (Computer Science) highlights several important trends (see Figure 4). It seems that the experience of being immersed in computer programming instruction for the duration of a week and a half served to temper most kids' self-reported confidence regarding both computer use and, more specifically, programming. However, once again we observe troubling gender differences: while boys pre- and post-attitudes hardly change, girls' confidence is observably lower (though not universally significantly so), except in the case where girls felt more confident with their ability to troubleshoot computer problems. This very well may be one of the strongest indicators and prerequisites for performance in computer programming – the confidence and knowledge to solve one's own problems. So while boys on the average went into the experience with that confidence, and then gained additional instrumental skills, for girls the exercise very much served to develop this type of confidence with computer programs. Since most participants added post-survey comments about their experience, we can also trace their reported impressions in relation to their attitudes and confidence levels. Interestingly, both girls and boys on the whole described the experience as 'fun' and 'exciting', highlighting the opportunity to do something they've 'never done before' and learn new skills. At the same time, both boys and girls acknowledged how challenging and difficult coding is – something that was a bit of a surprise to them:

> "When I did my own game I felt happy because I never created a game before so I felt excited. I also find it hard like for some of the parts but it was exciting." (Girl, aged 11)
>
> "My experience making my own game was amazing because I learned a lot of cool things. I also got a chance to see that computers aren't just for playing games but for making them too." (Boy, aged 11)
>
> *"It took so long to make something seemingly simple. It's really hard to use when you know nothing about it (the program). So much coding goes into one action." (Girl, aged 11)*
>
> *"It is hard and not simple because there are many codes needed and memorized." (Boy, aged 11)*

This newfound awareness of the complexity of computer programming might explain an interesting result with regard to attitudes to gender and specifically girls' perceived ability to program code. There was a significant difference between male and female students in their attitudes towards girls' ability to perform well at computer programming, $t(61) = -2.50$, $p < .05$. Girls' opinions changed to more strongly endorse the belief that girls could not do well at computer programming; conversely, boys' opinions changed to more strongly disagree with the belief that girls could not do well at computer programming. In other words, as the quantitative data shows, while girls lost some confidence in their own abilities, boys gained respect and appreciation for girls' abilities, having worked alongside them and also having been mentored by skilled female game programmers.

5.3 Study Design and Facilitation: Lessons Learned

While we have indicated in another section above that the curriculum we developed included direct instruction for the parts that involved procedural coding using the coding "window" that is available in Game Maker (see Figure 1, left side), we did have to spend quite a bit of time "tweaking" our curriculum while we were developing it. In this section we will briefly detail three primary lessons we learned in this pilot study. First, we found that it was necessary to begin each session with a piece of direct instruction that highlighted the programming concepts we wanted students to practice in their own games. Once that direct instruction was accomplished, we turned the rest of the time over for them to work on their own games, supported by the researchers and a team of facilitators that were trained in Game Maker. Interestingly, we also had to manage the facilitators' expectations for providing help, as they sometimes provided help by directly fixing code and/or by providing code that the students could not yet know in order to make a game work. Second, we found that we had to do quite a lot of managing of student expectations for the games they wanted to create; all too often they wanted to make games that exceeded their abilities and were not able to re-design their games with their limited abilities in mind. While this is not necessarily a surprising outcome, it was surprising to us how many participants were unfamiliar with just how much is involved in game design, and how demanding their designs were from a programming standpoint. This points back to the lack of any formal curriculum in Ontario with regards to computer programming, and also to the necessity for that at much earlier grade levels. Third and finally, as much as we wanted to create an 'open design' experience for participants, in hindsight the fact that we did not assign a game theme or genre, nor insisted that they replicate the game we used to demonstrate core concepts, meant that (for some) the task was overwhelmingly vague. For those who were overwhelmed, we often had them recreate the game that we used

as demonstration, which allowed all of them to proceed with the task, and some to change/hack the game in an interesting way.

6 In conclusion

This paper presents just some of the core findings from a pilot study that made use of a free, commercially available game design program (Game Maker), to introduce children to key computational thinking constructs such as variables, operations, functions, and conditionals, and allow them to practice applying this new knowledge. Overall, participants were enthusastic users of the tool, and did not struggle in the time we spent with them (for the most part) to stay on task or stay interested in their own game development. While we have not reported here on the affective engagement of our participants, it is in fact a highly relevant outcome of the study, and one that we will elaborate on in future papers, as we also had an opportunity to hear from parents of participating students who reported that their children were keen to continue working on their games outside of classroom time. Based on our preliminary discussion of the data above, there are three primary conclusions that are worth emphasizing. First, as others have pointed out, claims that today's students are defacto 'digitally native' is not the case for all students, nor does it indicate that students have familiarity or even facility with basic computer programming skills and competencies. Second, there are still gender differences in attitude and confidence with computers that in an instructional study such as this can and did affect performance on programming related tasks, not only on the post-test, but also in our many observations of girls during the time we spent with them. In general, girls were less willing to participate in public displays of knowledge (like answering questions to the whole group) and were more likely than their male counterparts to 'disavow' their skills with speech acts, such as "I always break the computer", and "I am not good at computers". We show that such differences in attidues can and do affect performance. Finally, our model of a structured curriculum that combines applied work with direct follow-along instruction is encouraging, and we hope eventually replicable in a school district-wide instructional programme. In conclusion, this preliminary analysis has shown that using commericallly available game design software, that permits a variety of scalable programming actions in the process of coding and testing a game, is not only a viable way of introducing a middle-school demographic to computational literacy but is an effective means for fostering and supporting STEM related competencies, vocabularies and skills.

Acknowledgements

We would like to acknowledge the Social Science and Humanities Research Council of Canada for funding this research and thank the many student participants and their teachers. We also acknowledge the invaluable support that our facilitators provided.

References

Barab, S, & Dede, C 2007, 'Games and Immersive Participatory Simulations for Science Education: An Emerging Type of Curricula', *Journal of Science Education and Technology*, 16(1), 1–3. doi:10.1007/s10956-007-9043-9

Carbonaro, M, Szafron, D, Cutumisu, M, & Schaeffer, J 2010, 'Computer-game construction: A gender-neutral attractor to Computing Science', *Computers & Education*, 55(3), 1098–1111. doi:10.1016/j.compedu.2010.05.007

De Castell, S, Boschman, L, & Jenson, J 2008 'In and out of control: Learning games differently'. *Loading...*, 2(3). Retrieved from http://journals.sfu.ca/loading/index.php/loading/article/viewArticle/66

Denner, J & Werner, L 2007 'Computer Programming in Middle School: How Pairs Respond to Challenges', *Journal of Educational Computing Research*, 37(2), 131–150. doi:10.2190/12T6-41L2-6765-G3T2

Denner, J, Werner, L & Ortiz, E 2012 'Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?' *Computers & Education*, 58(1), 240–249. doi:10.1016/j.compedu.2011.08.006

diSessa, A 2000 *Changing Minds: Computers, Learning, and Literacy*, MIT Press.

Duncan, C, Bell, T & Tanimoto, S 2014 'Should Your 8-year-old Learn Coding?' *WiPSCE 2014*, November05-072014, Berlin, Germany.

Francis, D 2012, 'It's time to fix our broken education system', *Financial Post* April 27, Retrieved October 8, 2013, from http://opinion.financialpost.com/2012/04/27/its-time-to-fix-our-broken-education-system/

Gee, JP 2003, *What video games have to teach us about learning and literacy*. Palgrave Macmillan.

Gee, J P 2005, 'Semiotic Social Spaces and Affinity Spaces: From The Age of Mythology to Today's Schools, in D Barton & K Tusting (eds.), *Beyond Communities of Practice: Language, Power, and Social Context* , N.Y: Cambridge University Press, Cambridge, pp. 214–232.

Heersink, D & Moskal, B 2010, 'Measuring High School Students' Attitudes Toward Computing', *SIGCSE 2010*, March 10–13, 2010, Milwaukee, Wisconsin, USA.

Hoegh, A & B Moskal 2009, 'Examining Science and Engineering Students' Attitudes Toward Computer Science', *SEE/IEEE Frontiers in Education Conference*, October 18-21, 2009, San Antonio, TX.

Jenson, J & de Castell, S 2005, 'Her own Boss: Gender and the pursuit of incompetent play', paper presented to Changing Views: Worlds in Play, DIGRA, DIGRA.

Kafai, YB 2006 'Playing and making games for learning: Instructionist and constructionist perspectives for game studies', *Games and Culture*, vol. 1, no. 1, pp. 34–40.

Klahr, D & Carver, S 1988, 'Cognitive Objectives in a LOGO Debugging Curriculum: Instruction, Learning and Transfer', *Cognitive Psychology*, 20, pp. 362-404.

Livingstone, S 2008, 'Internet Literacy: Young People's Negotiation of New Online Opportunities', in T McPherson (ed.), *Digital Youth, Innovation, and the Unexpected,* Cambridge, MA: MIT Press, pp. 101–122.

Lye, SY & Koh, JHL 2014, Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41, 51–61.

Lynch, K 2013, 'Toward Canadian public education 2.0', *The Globe and Mail* March 11, retrieved October 8, 2013 from http://www.theglobeandmail.com/commentary/toward-canadian-public-education-20/article9532122/

Papert, S 1980, *Mindstorms: Children, computers, and powerful ideas*, Basic Books New York.

Papert, S 1993, *The Children's Machine: Rethinking School in the Age of the Computer*, New York: BasicBooks.

Peppler, KA & Kafai, YB 2007, 'From SuperGoo to Scratch: exploring creative digital media production in informal learning', *Learning, Media and Technology*, 32(2), 149–166. doi:10.1080/17439880701343337.

Prensky, M 2005, 'Computer Games and Learning: Digital Game-Based Learning', in J Raessens & J Goldstein (eds.), *Handbook of Computer Game Studies* (pp. 97–122). Cambridge, MA: MIT Press.

Rieber, LP 1996, 'Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games', *Education Technology Research & Development*, 44(2), 43–58.

Rupp, AA, Gushta, M, Mislevy, RJ, & Shaffer, DW 2010, 'Evidence-centered Design of Epistemic Games: Measurement Principles for Complex Learning Environments', *The Journal of Technology, Learning, and Assessment*, 8(4), retrieved from http://napoleon.bc.edu/ojs/index.php/jtla/article/view/1623

Salen, K 2007, 'Gaming literacies: A game design study in action', *Journal of Educational Multimedia and Hypermedia*, 16(3), pp. 301–322.

Squire, K 2011, *Video Games and Learning: Teaching and Participatory Culture in the Digital Age*, Technology, Education–Connections (the TEC Series), Teachers College Press, 1234 Amsterdam Avenue, New York, NY 10027.

Steinkuehler, C 2008, 'Cognition and literacy in massively multiplayer online games', in J Coiro, M Knobel, C Lankshear & DJ Leu (eds.), *Handbook of research on new literacies*, New York: Lawrence Erlbaum Associates/Taylor & Francis Group, pp. 611–634.

Werner, L, Denner, J, Campe, S & Kawamoto, DC 2012, 'The Fairy Performance Assessment: Measuring Computational Thinking in Middle School', ACM Press, doi:10.1145/2157136.2157200, p. 215.

Wing, JM 2006, 'Computational Thinking', *Communications of the ACM*, 49(3), 33–35.

Yadav et al. 2014 'Computational Thinking in Elementary and Secondary Teacher Education', *ACM Transactions on Computing Education*, vol. 14, no. 1, article 5.