Computers & Education 141 (2019) 103633

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Can playing a video game foster computational thinking skills?

Weinan Zhao[a,*], Valerie J. Shute[b]

[a] Olivet University, 181 Hutchinson Ave, Wingdale, NY, 12594, USA
[b] Florida State University, College of Education, 1114 West Call Street, Tallahassee, FL, 32306-4453, USA

ARTICLE INFO

Keywords:
Computational thinking
Video games
Abstraction
Block-based programming

ABSTRACT

In this study, we evaluated the cognitive and attitudinal influences of playing a video game, *Penguin Go*, designed to target the development of middle school students' computational thinking (CT) skills. In addition to the overall effectiveness of the game, we investigated the impact of a specific game feature—constraints on the number of blocks in a solution. Results showed that after playing *Penguin Go* for less than two hours, students' CT skills improved significantly, but the additional constraints did not generate a significant impact on learning. In addition, the game overall did not influence students' attitudes toward computer science, but the constraints condition of the game negatively influenced students' attitudes toward computer science. Implications of the findings and possible directions for future research regarding using these types of games to foster computational thinking skills are discussed.

1. Introduction

1.1. Context of the problem

There continues to be increasing interest from researchers, educators, and policymakers in developing computational thinking (CT) at different educational levels, especially for K-12. In her influential article "Computational thinking" published in the *Communications of the ACM* in 2006, Jeannette Wing used the term computational thinking to represent how computer scientists think when they are approaching a problem. She argued that CT is a set of attitude and skills that is universally applicable and important for everyone to learn (Wing, 2006); and called for adding CT to every child's basic skill set, along with reading, writing, and arithmetic. In January 2016, President Obama announced *Computer Science for All*, the most recent government initiative to promote computer science (CS) education and CT for students in K-12 education (The White House, 2016).

Although the definition of CT varies among authors, the most frequently-cited definition by Cuny, Snyder, and Wing (2010) describes CT 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." It focuses on the conceptualization and underlying thought processes of solving a problem computationally rather than syntactically. Moreover, it is becoming generally accepted that the main components of CT include problem decomposition, abstraction, algorithmic thinking, conditional logic, recursive thinking, and debugging (Grover & Pea, 2013). Research has shown that CT may bring about a variety of cognitive and non-cognitive benefits, especially for learning in STEM subjects (e.g., Barr & Stephenson, 2011; Repenning et al., 2015; Sneider, Stephenson, Schafer, & Flick, 2014).

Increasing attention is also being paid to introducing CT to middle school children because the middle school years are a critical

* Corresponding author. 106 Building 11, 181 Hutchinson Ave, Wingdale, NY, 12594, USA.
E-mail addresses: mwzhao@olivetuniversity.edu (W. Zhao), vshute@fsu.edu (V.J. Shute).

https://doi.org/10.1016/j.compedu.2019.103633
Received 2 January 2019; Received in revised form 7 July 2019; Accepted 11 July 2019
Available online 12 July 2019
0360-1315/ © 2019 Elsevier Ltd. All rights reserved.

stage when children develop their relationships with specific academic fields and establish initial career orientations (Grover, Pea, & Cooper, 2014, pp. 343–348; Tai, Liu, Maltese, & Fan, 2006). However, middle school students often hold negative attitudes toward CS (Yardi & Bruckman, 2007), which can hinder them from developing CT skills. In this context, researchers have been actively exploring opportunities to introduce CT to middle school students with an aim to improve their attitude toward CS.

Programming, being a fundamental skill of CS, is also employed as a vehicle for CT development (Grover & Pea, 2013). Furthermore, to preclude learners from being distracted from the focus of CT on conceptualization and the underlying thought processes of solving a problem, block-based programming environments, represented by Scratch (Resnick et al., 2009), Snap! (Harvey & Mönig, 2010), and Blockly (Fraser, 2013), have been widely used to reduce the complexities related to the syntax of programming languages.

In terms of informal settings, Scratch and other block-based environments follow the tradition of constructionism (Harel, 1991; Papert, 1980) and encourage children to come up with their own ideas for a project (usually games, animations, or media projects) and to create personally meaningful artifacts through programming. This approach (often called design-based), has shown some success in motivating children into programming activities (Monroy-Hernández & Resnick, 2008), but has fallen short in relation to fostering deeper learning (e.g., Brennan & Resnick, 2012; Meerbaum-Salant, Armoni, & Ben-Ari, 2011).

Another less documented type of environment that can be used to introduce children to CT in informal settings is video games. In general, video games are very popular among secondary school students, and well-designed educational games can be good vehicles to foster a variety of valuable skills in education (Gee, 2007; Shute & Ventura, 2013). Video games that can foster CT usually provide players with a motivating context where they need to construct programs to solve a problem (e.g., Horn et al., 2016; Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012; Weintrop & Wilensky, 2012). Compared to design-based environments, such games have the potential to promote more intentional learning and provide richer learning support through different game mechanics (e.g., Land, 2000; Mayer, Mautone, & Prothero, 2002). However, research on using video games to foster CT skills is very limited. A number of video games have been developed to teach programming, such as *Wu's Castle* (Eagle & Barnes, 2009), *CodeCombat* (https://codecombat.com/), *CodeSpell* (Esper, Foster, Griswold, Herrera, & Snyder, 2014), and MiniColon (Ayman, Sharaf, Ahmed, & Abdennadher, 2018). But these games use a text-based programming language that requires the player to pay considerable attention to the details of syntax, which is not well aligned with CT (Kazimoglu et al., 2012). From the limited number of studies conducted with children (Esper et al., 2014), these types of games did have a positive impact on children's attitude toward programming and CS, as well as on some components of CT, such as conditional logic and debugging. However, their alignment with CT skills was incomplete.

A few games, though not explicitly targeting CT, based their design on block-based programming languages, such as *LightBot* (Gouws, Bradshaw, & Wentworth, 2013), *RoboBuilder* (Weintrop & Wilensky, 2012), games at Code.org, *Blockly Games* (https://blockly-games.appspot.com/), *Run Marco* (https://runmarco.allcancode.com), and the latest version of *CodeSpell*. However, their impact on CT development has not been evaluated or only evaluated to a small extent. The results are very preliminary considering the small sample sizes and the qualitative nature of the evaluations (Giannakoulas & Xinogalos, 2018; Kazimoglu et al., 2012; Weintrop & Wilensky, 2012).

In summary, research has shown some promising results regarding the cognitive and attitudinal impacts of video games on CT development among children, such as promoting learning of some CT components, and improving attitudes toward CS. However, the potential of video games in this regard seems not fully tapped. Video games can be designed to target CT skills, including using a block-based language like Blockly, to enable the player to further shift his or her cognitive focus from the syntax of programming to particular CT skills.

1.2. Purpose of the study

This study aimed to investigate the cognitive and attitudinal impacts of playing a video game that targeted the development of CT skills among middle school students. Toward this end, we designed and developed a web-based video game called "*Penguin Go*." It used Blockly as the code editor, and was carefully designed to align with main components of CT. Also, we integrated different types of learning support into the game to facilitate CT learning during gameplay. A detailed description of the game can be found in the "Interventions" section.

In addition to the overall impact of the game, one particular variable we explored in this study involves constraints imposed on the number of blocks used in a program, and how that affected the game's impact. We expected that such constraints would drive the player to look deeper into the structure of the problem, as well as the program he or she originally created, to identify common logic that can be used multiple times in the program to tackle different parts of the problem, and thus reduce its size. During this process, we expected the player to identify essential properties common to multiple objects or procedures, and ignore irrelevant differences among them. This is what Wing (2010) defined as the process of *abstraction*, the keystone of CT (Wing, 2008). Features similar to this have been included in games like *Lightbot*, *Blockly Games*, and *Program your bot*, to encourage players to create reusable code blocks, such as loops, functions, or modules (Kazimoglu et al., 2012), which is considered good programming practice. However, the impact of such constraints on CT development had not been evaluated prior to this study.

The attitudinal impact of this feature is related to multiple factors. First, the feature exerts additional constraints on gameplay, which is usually not preferred by video game players (Klimmt, Hartmann, & Frey, 2007; Ryan, Rigby, & Przybylski, 2006), and thus may result in reduced enjoyment. Furthermore, the additional challenges associated with the constraints may affect a player's enjoyment of gameplay. According to Csikszentmihalyi (1997), challenging yet achievable levels of difficulty is a necessary condition for a player to experience the state of flow in which the player is immersed completely into the activity with deep enjoyment. Since the perceived difficulty depends on the skill levels of each individual student, the impact of the constraints may vary among them. In

short, we had no definite hypothesis regarding how this constraint feature may influence students' attitudes. Given the potential benefits of such constraints and the absence of clear empirical evidence regarding their effects, we believe that investigations like would be valuable.

1.3. Research questions

The research questions of the study were:

1. Does playing *Penguin Go* have a positive impact on middle school students' CT skills?
2. Does playing *Penguin Go* have a positive impact on middle school students' attitudes toward future learning in CS?
3. Does playing *Penguin Go* with constraints imposed on the number of blocks in a solution have a greater impact on middle school students' CT skills than playing *Penguin Go* without such constraints?
4. Does playing *Penguin Go* with constraints imposed on the number of blocks in a solution have a greater impact on middle school students' attitudes toward CS than playing *Penguin Go* without such constraints?

1.4. Hypotheses

Based on the literature review regarding the impact of video games on CT skills and attitudes toward CS, as well as the aforementioned features of the game, we made the following hypotheses:

1. Playing Penguin Go would have a positive impact on middle school students' CT skills.
2. Playing Penguin Go would have a positive impact on middle school students' attitudes toward CS.
3. Adding constraints on the number of blocks in a solution to *Penguin Go* would further improve students' CT skills.
4. We did not have a definite hypothesis regarding the effect of constraints on the number of blocks on middle school students' attitudes toward CS. The investigation in this regard was exploratory in nature.

2. Material and methods

2.1. Research design

Since it was impossible to implement a control group by limiting a group of students from learning coding for several weeks, this study used a one-group pretest-posttest design to address research questions 1 and 2. In addition, a randomized experimental design was used to address research questions 3 and 4. We implemented the two designs at the same time within the study.

2.2. Participants

The participants of this study were recruited from a rural middle school in south Georgia. A total of 69 eighth grade students enrolled in this study on a voluntary basis, and we provided $15 gift cards as an incentive for participation. In addition, to encourage the students to try their best on both the CT pretest and posttest, an extra $10 gift card was given to each of the students who ranked in the top 25% of their class based on their performance on both tests.

2.3. Interventions

The game that was designed, developed, and used in the study is called "*Penguin Go*." Its backstory is based on the breeding behaviors of emperor penguins (Emperor Penguin, 2016). We chose this backstory not only because it provides a compelling narrative, but also because it provides a rich context where problems eliciting CT skills can be built upon. For example, a typical puzzle in the game will ask the player to create a program to guide the penguin through an area. Different parts of the terrain are covered with ice, snow, or rocks. To solve this puzzle, the player must employ different components of CT skills: 1) *Algorithmic thinking*: the player needs to develop a sequence of steps and explicitly put them into a program so that the penguin can carry it out. 2) *Conditional logic*: to make the penguin go forward, the player has the option to issue a *waddle* or a *toboggan* command. However, depending on the condition of the ground, one method of travel works while the other does not. Therefore, the player needs to choose the action to take depending on environmental conditions, which is conditional logic. 3) *Debugging*: when a program is not working as desired, the player must review the program, identify the errors in it, revise and test it, and then repeat the process until the program works as desired. This is exactly the process of debugging in programming. Depending on the complexity of the puzzle, other CT components may also be involved.

Fig. 1 shows a sample level of *Penguin Go*. The left part of the screen is the game world and the right part is the code editor based on Blockly. The task of the player is to create a program to guide the penguin to the goal—the yellow footprints. In this level, there are two possible routes from the penguin to the goal. Both routes are composed of areas covered by snow (the white tiles) and ice (the light blue tiles). To move forward, the penguin has to *waddle* on the snow areas and *toboggan* on the ice areas.

The blocks available for this level are shown in Fig. 2. Using these blocks, the player may develop different solutions that involve different levels of abstraction. Suppose the player selected the route marked by the shaded arrows. The three possible solutions to

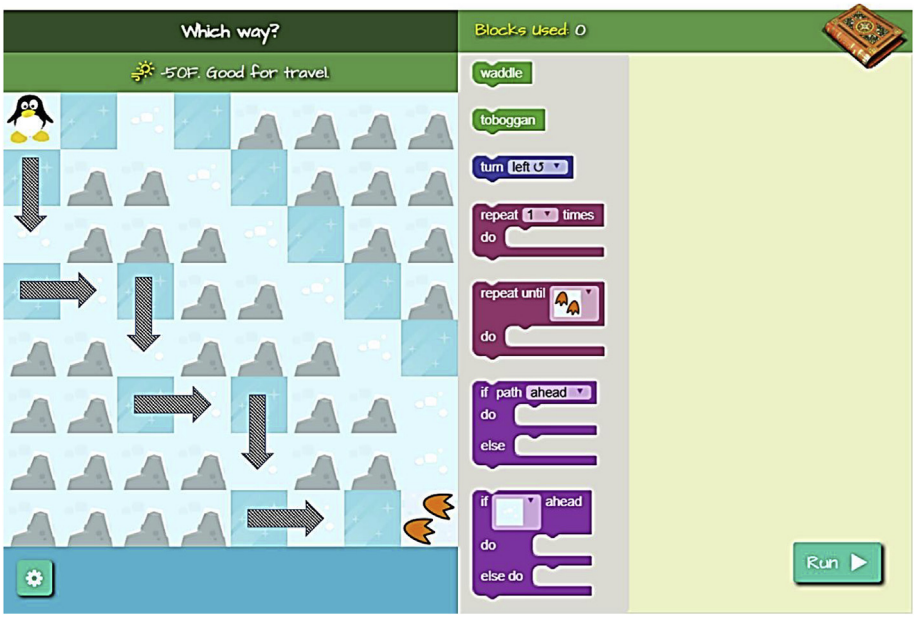


Fig. 1. Screen capture of a sample level in *Penguin Go*. The shaded arrows were added in this document for illustration purpose. They are not part of the game.

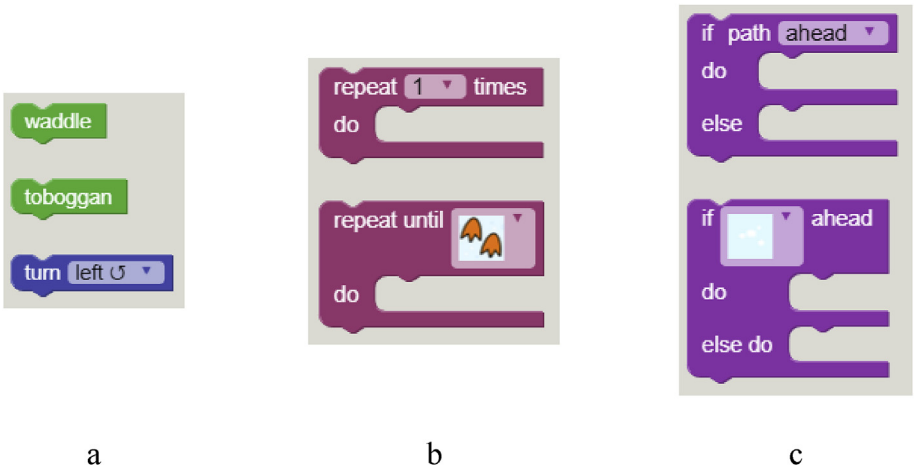
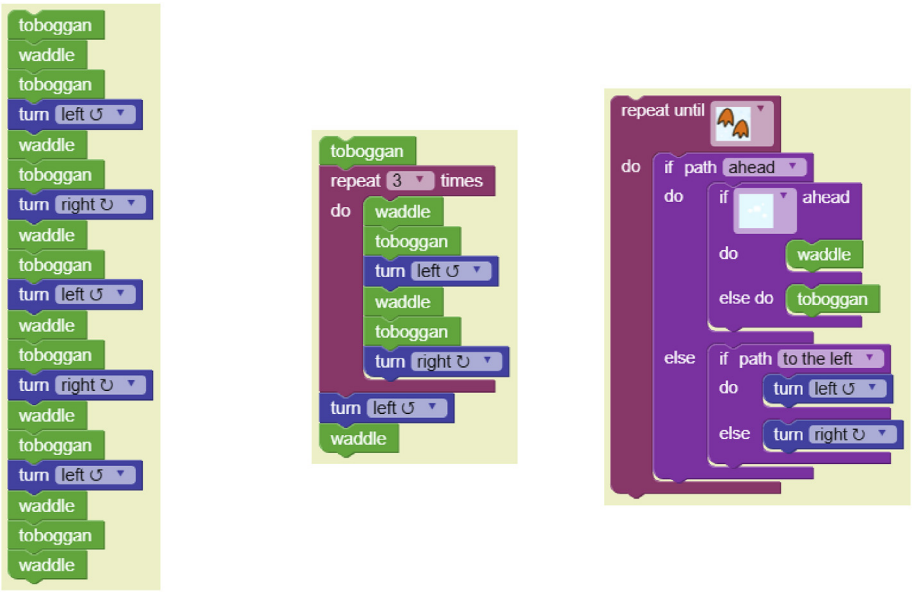


Fig. 2. Blocks available in the sample level of *Penguin Go*: a. action commands, b. loop commands, c. conditional commands.

navigate the penguin through the route are shown in Fig. 3. Solution *"a"* represents a basic understanding of algorithmic thinking. The player only needs to put the action commands into a proper sequence without thinking further. Solution *"b"* involves a certain level of abstraction. The player needs to identify a code snippet that can be repeatedly used in the solution. This involves breaking the path down to small parts to which the same code snippet can be applied. Solution *"c"* represents an even higher level of abstraction that requires the player to group the action commands and develop a generalized code snippet for each group. For example, *waddle* and *toboggan* are generalized to an if-do-else statement that moves the penguin forward regardless of the type of area ahead. With this piece of code, the player can move the penguin forward without worrying about the detailed logic. This piece of code, in turn, can be used as a unit to construct larger generalized structures.

As the level of abstraction increases, the number of blocks used in the solution decreases from 19 to 10 to 8. In this study, one version of the game has constraints on the number of blocks in a solution (to promote players' abstraction skills) while the other version had no constraints.

The learning support in the game takes three forms. First, guidance is provided at the beginning of the game, to orient the player to the game mechanics; and when a new type of programming block is introduced, to inform the player of the usage of the blocks. Second, the sequence of the puzzles was designed to be in line with the three stages of the "Use-Modify-Create" progression, a learning progression frequently used to engage young learners in CT (Grover & Pea, 2013; Lee et al., 2011). Third, new types of blocks



a b c

Fig. 3. Solutions of the sample level: a. number of blocks = 19, b. number of blocks = 0, c. number of blocks = 8.

are introduced gradually to reduce cognitive load.

The game consists of 27 puzzles and takes about three hours for a child of middle school age to finish.

2.4. Independent variable

The independent variable in this study was the version of the game, with two possible levels: *constraints* and *no-constraints*. In the *constraints* version, after certain types of blocks (such as loops and functions) were introduced, the number of blocks a player can include in his or her program to solve the puzzle was limited to a specific number. In the *no-constraints* version, there were no such constraints.

2.5. Dependent variables

The dependent variables of this study included middle school students' CT skills and their attitudes toward future learning in CS.

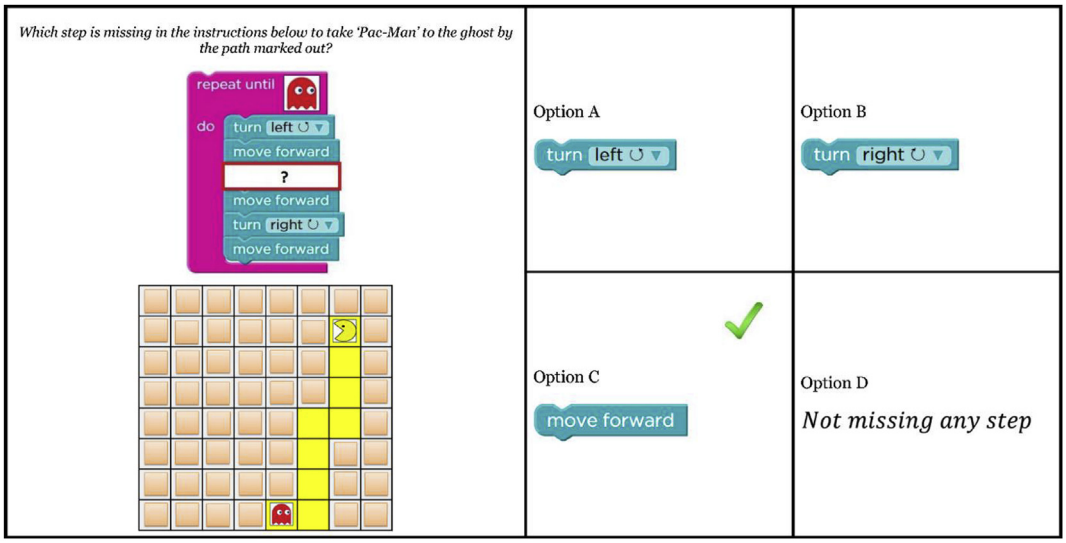
To measure middle school students' CT skills, a pretest and a posttest were administered online before and after gameplay. Both tests were adapted from the Computational Thinking Test (CTt) developed by Román-González, Pérez-González, and Jiménez-Fernández (2017). The adaptations included dropping irrelevant items, fixing some language and representation issues, and creating an alternate form. Each new form consisted of 21 items, and took about 20–30 min to finish. For convenience, this modified version of CTt is referred to as MCTt in the following sections. Two example items of MCTt are shown in Fig. 4, representing respectively the two different programming interfaces used in MCTt (*block* vs. *arrow*).

To measure middle school students' attitudes toward CS, a survey adapted from Ericson and McKlin (2012) was used both before and after gameplay. The survey targets CS perceptions and consists of ten 5-point Likert scale questions, such as “computers are fun” and “computing jobs are boring.” The survey was also administered online. For convenience, this survey is referred to as the CSA Survey in the following sections.

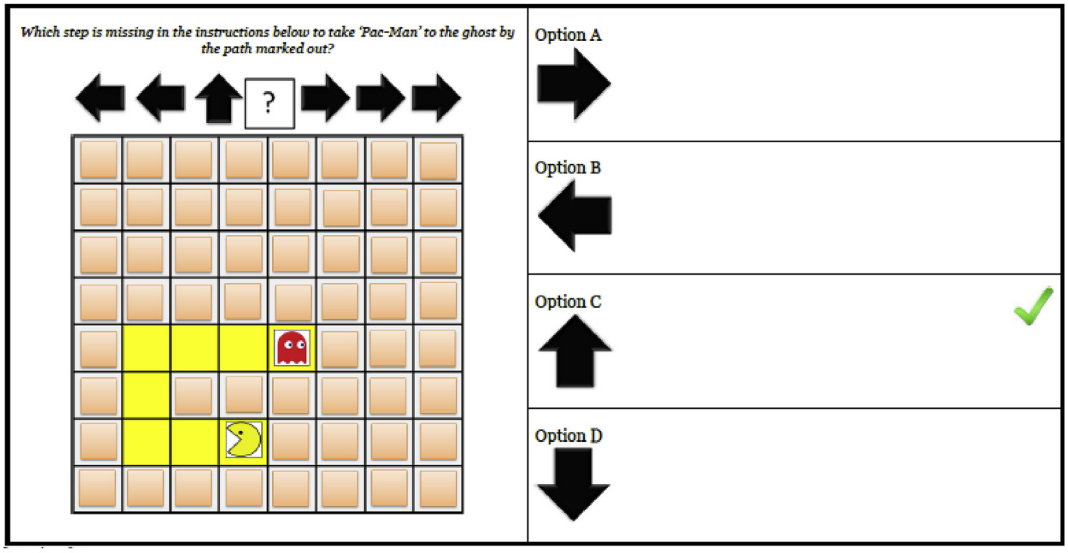
2.6. Other measures

Demographic questionnaire. The demographic questionnaire collected basic demographic information and information about students' experience with coding, to provide a general profile of the participants.

Reporting items. At the end of the study, students were asked to report whether they performed other coding activities during the period of study, and whether they tried their best on the posttest. Such information was used to identify and filter out data affected by confounding factors.



a. An example item of MCTt using blocks



b. An example item of MCTt using arrows

Fig. 4. Example items of MCTt, a. An example item of MCTt using blocks. b. An example item of MCTt using arrows.

2.7. Procedure

Students who signed up for the study were randomly assigned to one of the two conditions. The study included three 60-min sessions during three weeks in 2017. During the first session, all students completed the demographic questionnaire, the CSA Survey, and the pretest of CT for 30 min, and then played *Penguin Go* for 30 min. During the second session, students played *Penguin Go* through the 60-min session. During the final session, students played *Penguin Go* for the first 20 min, and completed the CSA Survey; as well as the CT posttest for the rest of the session. Depending on their assigned condition, students played the *constraints* or the *no-constraints* version of the game. Fig. 5 illustrates the procedure of the study.

All of the activities took place in three computer labs at the target school. Students finished the activities individually, and two proctors provided minimal support if a student asked for help regarding how to solve a puzzle in the game.

There were some technical issues that occurred during the administration of the study. For some students (about 20%), the game

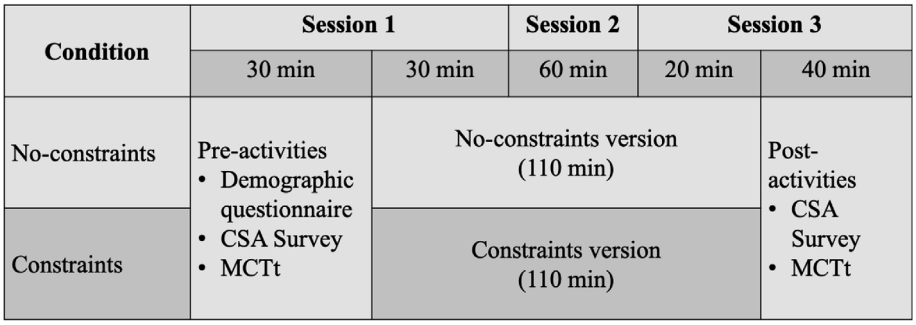


Fig. 5. Procedure of the study.

froze randomly at some point when students passed a level, and they had to refresh the webpage in order to continue playing. Occasionally, after refreshing the page, some students lost their latest progress in the game. Such issues, however, happened randomly and not specific to either of the two game conditions.

3. Results

3.1. Demographic information

Among the 69 students who signed up for the study, three students dropped out due to changes in their schedules or other unknown reasons. To reduce the influence of confounding factors, we filtered the data using students' responses to the reporting items. First, we excluded 16 students who reported that they performed other coding activities during the period of study. Second, we excluded another five students who reported that they rushed through the posttest. In addition, we excluded two students who had extensive coding experience prior to the study (*prior coding time* $>$ 40 h), according to their responses to one of the questions on the demographic questionnaire. As a result, a total of 43 students were included in the final analysis. There was a small imbalance in the numbers of participants between conditions, with $n = 20$ for the *constraints* group and $n = 23$ for the *no-constraints* group.

The average age of the 43 students was 13.3 ($SD = 0.55$). Among them, 22 (51%) students were male and 21 (49%) students were female. In terms of ethnicity, 23 (53%) students were White; 14 (33%) students were Black; and 6 (14%) students were Hispanic, Latino, or other. In general, most students (95%) had experience with video games in the past, but their experience with puzzle-style games was rare—72% of students reported that they rarely or had never played puzzle-style games.

The majority of students (93%) had prior experience with coding, usually through Code.org (54%), Scratch (2%), or both (37%). However, most of them (70%) only spent less than 5 h on coding in the past. Table 1 shows the frequencies of students' prior "coding time" by condition. A two-way chi-square test showed that students' prior "coding time" did not differ by condition, χ^2 (4, $n = 43$) = 0.17, $p = .19$.

3.2. Modified Computational Thinking Test (MCTt)

CT skill was measured by the MCTt. Students were randomly assigned one of the forms as pretest and the other as posttest. Each item was scored as 1 (correct) or 0 (incorrect), and the maximum score for each test was 21. The Cronbach's alpha coefficients for pretest and posttest were .74 and .72, respectively. Both indicated acceptable reliability.

Table 2 shows the descriptive statistics of pretest and posttest scores by condition. For both groups, the mean posttest score was higher than the mean pretest score.

3.3. Computer science attitude survey (CSA survey)

Students' attitudes toward computer science (CS) were measured by the CSA Survey that consisted of ten 5-point Likert scale items, with 1 representing strongly disagree and 5 representing strongly agree. The score for each student was calculated as the mean

Table 1
Frequencies of Participant Prior Coding Time by Condition.

	Prior Coding Time					Total
	0	< 5 h	5–10 h	10–20 h	20–40 h	
no-constraints group	3	14	3	3	0	23
constraints group	0	13	3	2	2	20
All	3	27	6	5	2	43

Table 2
Descriptive Statistics of Pre- and Post-MCTt Scores by Condition.

	Pretest *M* (*SD*)	Posttest *M* (*SD*)
no-constraints group	12.91 (3.79)	14.09 (3.82)
constraints group	12.90 (3.55)	14.00 (3.04)
All	12.91 (3.64)	14.05 (3.44)

score across all the 10 items. The Cronbach's alpha coefficients for the pre- and post-survey were 0.85 and 0.83, respectively. Both indicated good reliability. Table 3 shows the descriptive statistics of the pre- and post-survey scores by condition. For the *no-constraints* group, the mean of the post-survey scores was slightly higher than the pre-survey scores, while for the *constraints* group, the mean of the post-survey scores was lower than the pre-survey scores.

An independent *t*-test showed that the mean of the pre-survey scores of the *no-constraints* group ($M = 3.98$, $SD = 0.39$) was significantly higher than that of the *constraints* group ($M = 3.54$, $SD = 0.70$); $t\ (41) = -2.52$, $p = .016$. It indicated that prior to playing *Penguin Go*, the two groups were not equivalent in terms of students' attitudes toward computer science. Given participants were randomly assigned to the two conditions, and the two groups were mixed in the same computer lab instead of being separated, it was hard to find a good explanation for such non-equivalency, except for an unfortunate random assignment. Such non-equivalency was taken into consideration in the following analysis.

3.4. Research question 1

The first research question was, "Does playing *Penguin Go* have a positive impact on middle school students' CT skills?" Our hypothesis was that playing *Penguin Go* would have a positive impact on middle school students' CT skills. Given no serious violation of the assumptions of normality and outliers, we conducted a paired *t*-test to compare pretest and posttest scores. Results showed an overall significant difference between pretest and posttest: $t\ (42) = -3.43$, $p = .001$. Students' CT skills after playing *Penguin Go* were significantly higher ($M = 14.05$, $SD = 3.64$) than their CT skills prior to playing *Penguin Go* ($M = 12.91$, $SD = 3.44$), with a moderate effect size (*Cohen's d* = 0.53). This result supported our hypothesis that playing *Penguin Go* would have a positive impact on middle school students' CT skills. However, since this was a one-group pretest-posttest design, the result should be interpreted with caution, which will be examined further in the Discussion section.

3.5. Research question 2

The second research question was, "Does playing *Penguin Go* have a positive impact on middle school students' attitudes toward future learning in CS?" Our hypothesis was that playing *Penguin Go* would have a positive impact on middle school students' attitudes toward CS. A review of the data revealed that there was a large outlier that was not due to errors in data collection or scoring, and thus an assumption of the paired *t*-test was violated.[1] Therefore, we conducted a paired-samples sign test instead to compare the pre-CSA and post-CSA Survey scores, and found no significant difference: $Z = -1.11$, $p = .27$. Students' attitudes toward CS after playing *Penguin Go* ($M = 3.63$, $SD = 0.65$) were not significantly higher than their attitudes toward CS prior to playing *Penguin Go* ($M = 3.74$, $SD = 0.61$). The effect size of the difference was small (*Cohen's d* = 0.18). This result did not support our hypothesis that playing *Penguin Go* would have a positive impact on middle school students' attitudes toward CS.

We further examined the attitudinal changes by condition. Table 4 shows the standardized 95% confidence intervals of the change (*pre-survey - post-survey*) in CSA Survey scores for each condition. For the *no-constraints* group, the confidence interval indicated that although the mean of the post-CSA Survey scores was higher than that of the pre-CSA Survey scores, the difference between them was not significant. For the *constraints* group, the confidence interval suggested that the post-CSA Survey scores were significantly lower than the pre-CSA survey scores.

3.6. Research question 3

The third research question was, "Does playing *Penguin Go* with constraints imposed on the number of blocks in a solution have a greater impact on middle school students' CT skills than playing *Penguin Go* without such constraints?" Our hypothesis was that adding such constraints to *Penguin Go* would further improve students' CT skills.

Given no serious violation of the assumptions of normality or homogeneity variance, we conducted a mixed-design ANOVA with MCTt scores as the dependent variable, time (pretest vs. posttest) as the within-subjects factor, condition as the between-subjects factor, and pre-CSA survey scores as the covariate (to control for the non-equivalency between conditions in terms of students' attitudes toward computer science). Results from the analysis showed that the interaction between time and condition was not significant, $F\ (1, 40) = 0.005$, $p = .941$; that is, the changes in MCTt scores from pretest to posttest were not significantly different between the two conditions. Fig. 6 compares the two conditions regarding their change from pretest to posttest, using the estimated marginal means.

[1] Excluding the outlier from the dataset did not change the results of the analysis on student attitudes toward CS after playing *Penguin Go*.

Table 3
Descriptive Statistics of Pre- and Post-CSA Survey Scores by Condition.

	Pre Survey *M* (*SD*)	Post Survey *M* (*SD*)
no-constraints group	3.54 (0.70)	3.60 (0.78)
constraints group	3.98 (0.39)	3.66 (0.48)
All	3.74 (0.61)	3.63 (0.65)

Table 4
Standardized 95% Confidence Intervals of the CSA Survey Gain Scores by Condition.

	Lower	Upper
no-constraints group	−0.535	0.329
constraints group	0.284	1.221

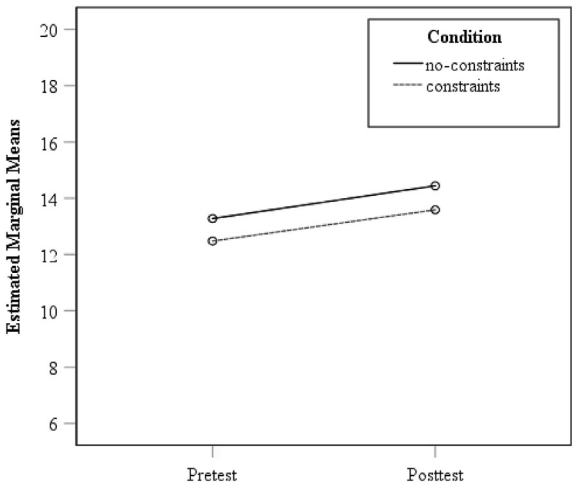


Fig. 6. MCTt estimated marginal means by condition and time (pretest vs. posttest).

3.7. Research question 4

The fourth research question was, "Does playing *Penguin Go* with constraints imposed on the number of blocks in a solution have a greater impact on middle school students' attitudes toward CS than playing *Penguin Go* without such constraints?" We did not have a definite hypothesis for this question and the investigation in this regard was exploratory in nature.

Given the low statistical power (0.49) with the small sample size (n = 43), and the correlation (r = 0.55) between the pre- and post-CSA Survey scores, we did not conduct any hypothesis testing; instead, we calculated the 95% confidence interval for the difference between gain scores of the two conditions, which was between 0.03 and 0.75. Therefore, it is reasonable to believe that there was a real difference between the gain scores of the two conditions. Fig. 7 shows the two conditions in their changes from pre-to post-survey, using the original means.

4. Discussion

In this study, we evaluated the effectiveness *Penguin Go*—a video game designed to target main components of CT—on developing middle school students' CT skills, as well as on their attitudes toward CS. In addition to the overall effectiveness of the game, we investigated the impact of a specific game feature—constraints on the number of blocks in a solution on the same learning outcomes. Results showed that after playing *Penguin Go* for less than two hours, students' CT test scores improved significantly with a medium effect size; however, there was no significant difference between the learning gains of the two conditions. In terms of changes in attitudes toward CS, the two conditions differed significantly. The no-constraints group showed a positive but nonsignificant change in attitude, while the constraints group showed a significant negative change.

4.1. Research question 1

Our hypothesis regarding the first research question was that playing *Penguin Go* would have a positive impact on middle school students' CT skills. This hypothesis seemed to be supported by the results. That is, after playing *Penguin Go* for less than two hours, the

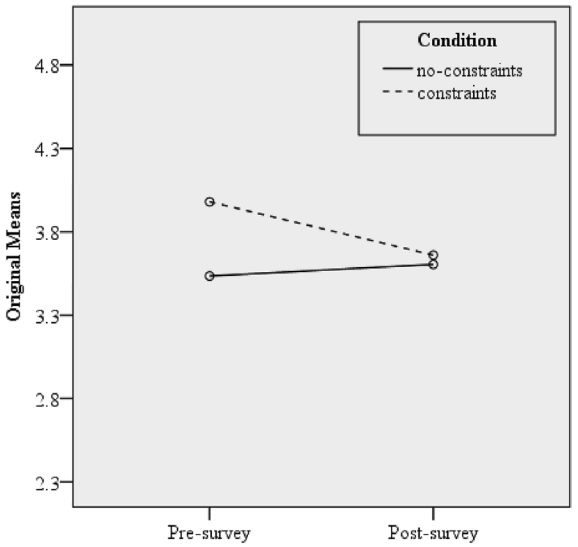


Fig. 7. CSA Survey mean scores by condition and time (pre-survey vs. post-survey).

average CT skills of 43 middle school students improved significantly with a medium effect size. Although it was not possible to compare this treatment effect quantitatively with previous research due to the differences in many aspects of the studies, the effect was generally consistent with the findings from prior research using games to teach CT related skills to different target populations (Eagle & Barnes, 2009; Horn et al., 2016), as well as the effects of educational games in general (e.g., Gee, 2007; Shute, Rieber, & Van Eck, 2011; Wouters, van Nimwegen, van Oostendorp, & van der Spek, 2013).

However, because the results were obtained from a one-group pretest-posttest design, it was possible that the difference between the pretest and posttest was due to uncontrolled confounding factors. Though most of these factors were either irrelevant (e.g., instrument decay, regression toward the mean) or very minor (e.g., maturation) to this study, or were partially controlled (e.g., history), the effect of testing cannot be safely ignored. In order to investigate if the gain in students' MCTt scores was due to students' increased familiarity with the block-based interface, we compared the gain of students' MCTt scores on the *block* items (Fig. 4a) and the *arrow* items (Fig. 4b). Results showed that students' posttest scores on the *block* items were significantly higher than their pretest scores on the *block* items; $t\,(42) = -3.25$, $p = .003$; while the difference between students' pretest and posttest scores on the arrow items was not significant; $Z = -0.49$, $p = .624$. As the *block* items, compared to the *arrow* items, have a programming interface more similar to the interface used in the game, such results indicated that a non-trivial portion of students' improvements on their MCTt scores may be attributed to increased student familiarity with the block-based interface.

4.2. Research question 2

Our hypothesis regarding the second research question was that playing *Penguin Go* would have a positive impact on middle school students' attitudes toward CS. This hypothesis was not supported by the results. Overall students' attitudes toward CS after playing *Penguin Go* were not significantly higher than their attitudes before gameplay. When viewing the data by specific condition, there was a nonsignificant increase in the mean attitude score of the *no-constraints* group, and a significant decrease in the mean attitude score of the *constraints* group, indicating a stronger main effect of condition over the treatment effect of the game. The differences between conditions, as well as the significant decrease in the *constraints* group, are described in detail in the Discussion regarding research question 4. Here we provide some possible explanations for the result of the nonsignificant increase in the mean attitude score of the *no-constraints* group.

First, the effect itself is not very large. Previous research has shown some positive effect of video games on certain aspects of students' attitudes toward CS, including confidence in programming and self-identification as a programmer (Esper et al., 2014), as well as enjoyment of the activities (Horn et al., 2016). However, the results were either qualitative or based on only specific items in the survey, and it is thus unclear whether the effects were significant or not. In addition, according to a meta-analysis conducted by Wouters et al. (2013), the impact of video games on students' attitudes in general was not significantly different from the impact of other instructional methods (e.g., passive instruction, drill and practice). Moreover, students' attitudes toward CS often decrease after introductory experiences in CS (Anderson, McKenzie, Wellman, Brown, & Vrbsky, 2011). Therefore, it is reasonable to believe the effect size of playing *Penguin Go* on students' attitudes was small, if not none. Second, the treatment time was less than two hours, which may not be long enough to generate detectable effects. Finally, with a sample size of 43 and sufficient statistical power of .80, the minimum detectable effect size was 0.44, which was unlikely given the reasons discussed above.

4.3. Research question 3

Our hypothesis regarding the third research question was that adding constraints imposed on the number of blocks in a solution to *Penguin Go* would further improve students' CT skills. This hypothesis was not supported by the results. The changes in MCTt scores from pretest to posttest of the *constraints* group were not significantly different from that of the *no-constraints* group. One possible explanation for these results was that students in the *constraints* group used a trial and error method to figure out the solution to a level, which involved not much abstraction thinking. Also, although abstraction is believed by many to be the keystone of CT (Grover & Pea, 2013; Wing, 2008), empirical evidence regarding the correlations between abstraction thinking and learning CS is mixed (Bennedssen & Caspersen, 2008). The actual correlation between abstraction and other target components of CT in this study may not be sufficiently high to warrant detectable differences with the small sample size.

4.4. Research question 4

We did not have a hypothesis regarding the fourth research question "Does playing *Penguin Go* with constraints imposed on the number of blocks in a solution have a greater impact on middle school students' attitudes toward CS than playing *Penguin Go* without such constraints?" The investigation was exploratory in nature. Results showed a significant difference between the attitude changes of the two conditions. There was a slight increase in the mean scores of the *no-constraints* group, and a significant decrease in the mean scores of the *constraints* group. The negative attitude change in the *constraints* group may be the result of additional constraints imposed on the game. First, the constraints increased the difficulty for students to do what they wanted to do in the game, and thus may have reduced students' sense of autonomy. Second, these constraints made the levels more challenging to solve, and thus may have lowered students' perceptions of competency. Since perceptions of autonomy and competency are both important predictors of enjoyment and future gameplay (Klimmt et al., 2007; Ryan et al., 2006), reduced autonomy and competency may account for the negative attitude change in the *constraints* group.

4.5. Implications

This study aimed to provide empirical evidence regarding the cognitive and attitudinal impacts of playing video games that target the development of CT skills among middle school students. Toward this end, we designed and developed a video game called *Penguin Go* and evaluated its effectiveness with 43 middle school students. Results indicated that video games like *Penguin Go* may be an effective tool to develop CT skills among middle school students. Such games should include features like: 1) a code editor based on block-based programming languages; 2) tasks that need a player to employ core components of CT to complete, including problem decomposition, abstraction, algorithmic thinking, conditional logic, iterative thinking, and debugging; and 3) carefully designed in-game learning support to appropriately facilitate CT learning.

In addition, this study investigated the effect of a specific game feature—constraints on the number of blocks in a solution. Results showed that this feature could result in negative changes in students' attitudes toward CS, with no significant cognitive benefits. As discussed above, the reason for the negative attitude changes may be that the game version with the additional constraints was too challenging for students who had only a few hours of coding experience. Therefore, it is not recommended to integrate this feature into the game until students' CT skill has reached a certain level. The appropriate time to introduce this feature to the game is a topic deserving further research.

Finally, changes in attitude in the no-constraints group may be either small or none, and thus games similar to Penguin Go cannot be expected to generate a major impact on students' attitudes toward CS in educational practice.

4.6. Limitations

As mentioned earlier, due to the availability of the participants, this study did not implement a control group when investigating the overall impact of the game, which posed some threats to the internal validity of the study. On the other hand, the game was played in a classroom setting where students' perceptions of the game and their exhibited behaviors may be different from students who play a game related to their own particular interests in informal settings. Therefore, the results of this study should be generalized with caution.

Furthermore, the actual sample size obtained was small, which may account for not detecting significant effects regarding some of the research questions. Also, the time interval between study sessions was one week, with the posttest administered immediately after gameplay during the last session. Such an arrangement was necessary given the class schedule at the middle school, instead of an optimal design based on relevant research to maximize the effect of the intervention.

Finally, the game used in this study had some limitations. First, in order to navigate the penguin though a puzzle, the player needs to take the perspective of the penguin. Depending on the degree of difference between the perspectives of the player and the penguin, such a cognitive task may result in different levels of cognitive load that is not relevant to their CT skills. In the current version of the game, the penguin always starts at the top and moves down toward the bottom of the screen. Consequently, for most of the gameplay time the perspective of the penguin is different from that of the player. Such a difference may have resulted in extraneous cognitive load in the students that is not negligible, and thus affected students' learning of CT skills. Second, some aspects of the game design were not consistent with well-established principles of game design. For instance, at the time of the study, despite some words of excitement and praise from the penguin after winning a level, the game did not have an additional reward system, which is believed

to have a motivational effect (i.e., increasing players' motivation in playing the game) if well designed (Wang & Sun, 2011). Also, as mentioned earlier, there were some technical issues that occurred during the administration of the study. Although those issues only affected a small portion of students, and their occurrence was approximately equal across the two condition groups, they may have caused reduced enjoyment with the activity among those students and in turn affected their attitudes toward CS, and thus introduced some confounding factors to the study.

5. Conclusions

The findings described in this paper are generally consistent with prior research on using games to teach CT-related skills to different target populations. However, this study distinguishes itself in three aspects. First, *Penguin Go* was specifically designed to target CT; therefore, unlike the games used in prior research, it covers most core components of CT. Second, instead of a text-based programming language or mini-language, *Penguin Go* used Blockly as the code editor to reduce the need for students to master the details of syntax while at the same time keeping the affordances to support advanced skills. Finally, unlike prior research, this study was conducted with typical middle school students enrolled in a public school. Given the increasing importance of CT and the need to introduce CT to middle school students, findings from this study may provide valuable information. In fact, *Penguin Go* shares many features with the games on Code.org. Code.org has been widely used among middle school students; however, to the best of our knowledge, there is no quantitative evaluation regarding its effectiveness on developing CT skills. The findings in this study could may be a good reference when trying to estimate the cognitive and attitudinal impact of such games.

There are several possible directions that deserve further investigation. First, compared to other types of CT environments, such as *Scratch* and *Alice*, games like *Penguin Go* have a relatively low level of "Openness," which refers to the types of things a player can do and the number of choices the player has in these games. We have argued earlier that the added control to the environments could provide better learning support to the player; however, it may also result in a lower sense of autonomy and thus affect the player's motivation to learn. It would be interesting to compare different versions of such games with different degrees of "openness" to determine the optimal configuration. Second, as mentioned earlier, it is meaningful to examine the proper time when the constraints on the number of blocks can be added to the game to generate any cognitive or attitudinal benefits. Finally, given the increasing needs in CS education to draw more females and minorities to the area, it is important to learn how these populations respond to different types of games, and how we can improve these games to accommodate their unique needs.

Acknowledgments

We would like to thank Robert Reiser, Fengfeng Ke, Russell Almond, and Larry Dennis, for providing valuable feedback on this work. In addition, we are grateful to Chen Sun, Xi Lu, Lucas Liu, and Seyed Ahmad Rahimi, who helped us proctor different study sessions; as well as Marcos Román-González, who generously shared with us the Computational Thinking Test (CTt).

References

Anderson, M., McKenzie, A., Wellman, B., Brown, M., & Vrbsky, S. (2011). Affecting attitudes in first-year computer science using syntax free robotics programming. *ACM Inroads, 2*(3), 51–57.

Ayman, R., Sharaf, N., Ahmed, G., & Abdennadher, S. (2018). MiniColon; teaching kids computational thinking using an interactive serious game. In S. Göbel,, A. Garcia-Agundez,, T. Tregel,, M. Ma,, J. Baalsrud Hauge,, & M. Oliveira, (Vol. Eds.), *Serious games: Vol. 11243*, (pp. 79–90). . https://doi.org/10.1007/978-3-030-02762-9_9.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads, 2*(1), 48–54.

Bennedssen, J., & Caspersen, M. E. (2008). Abstraction ability as an indicator of success for learning computing science? *Proceedings of the fourth international workshop on computing education research* (pp. 15–26). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1404523.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (pp. 1–25). . Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf.

Csikszentmihalyi, M. (1997). *Finding flow: The psychology of engagement with everyday life.* New York, NY: Basic Books.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.

Eagle, M., & Barnes, T. (2009). Experimental evaluation of an educational game for improved learning in introductory computing. *ACM SIGCSE bulletin: Vol. 41*, (pp. 321–325). . ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1508980.

*Emperor Penguin*Wikipedia. Retrieved February 2, 2017, from https://en.wikipedia.org/wiki/Emperor_penguin.

Ericson, B., & McKlin, T. (2012). Effective and sustainable computing summer camps. *Proceedings of the 43rd ACM technical symposium on computer science education* (pp. 289–294). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2157223.

Esper, S., Foster, S. R., Griswold, W. G., Herrera, C., & Snyder, W. (2014). CodeSpells: Bridging educational language features with industry-standard languages. *Proceedings of the 14th Koli calling international conference on computing education research* (pp. 05–14). . Retrieved from http://dl.acm.org/citation.cfm?id=2674684.

Fraser, N. (2013). *Blockly.* https://developers.google.com/blockly/:Google.

Gee, J. P. (2007). *What video games have to teach us about learning and literacy.* New York: Palgrave Macmillan.

Giannakoulas, A., & Xinogalos, S. (2018). A pilot study on the effectiveness and acceptance of an educational game for teaching programming concepts to primary school students. *Education and Information Technologies, 23*(5), 2029–2052. https://doi.org/10.1007/s10639-018-9702-x.

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game light-bot. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 10–15). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2466518.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051.

Grover, S., Pea, R., & Cooper, S. (2014). *Remedying misperceptions of computer science among middle school students.* ACM Presshttps://doi.org/10.1145/2538862.2538934.

Harel, I. (1991). *Children designers: Interdisciplinary constructions for learning and knowing mathematics in a computer-rich school.* Ablex Publishing.
Harvey, B., & Mönig, J. (2010). Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists. In: *Proc. Constructionism*.
Horn, B., Clark, C., Strom, O., Chao, H., Stahl, A. J., Harteveld, C., et al. (2016). *Design Insights into the Creation and Evaluation of a Computer Science Educational Game.* ACM Press576–581. https://doi.org/10.1145/2839509.2844656.
Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia - Social and Behavioral Sciences, 47*, 1991–1999. https://doi.org/10.1016/j.sbspro.2012.06.938.
Klimmt, C., Hartmann, T., & Frey, A. (2007). Effectance and control as determinants of video game enjoyment. *CyberPsychology and Behavior, 10*(6), 845–848. https://doi.org/10.1089/cpb.2007.9942.
Land, S. M. (2000). Cognitive requirements for learning with open-ended learning environments. *Educational Technology Research & Development, 48*(3), 61–78.
Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. *Acm Inroads, 2*(1), 32–37.
Mayer, R. E., Mautone, P., & Prothero, W. (2002). Pictorial aids for learning by doing in a multimedia geology simulation game. *Journal of Educational Psychology, 94*(1), 171–185.
Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in scratch. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 168–172). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1999796.
Monroy-Hernández, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. *Interactions, 15*(2), 50–53.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.
Repenning, A., Grover, R., Gutierrez, K., Repenning, N., Webb, D. C., Koh, K. H., et al. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education, 15*(2), 1–31. https://doi.org/10.1145/2700517.
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67. https://doi.org/10.1145/1592761.1592779.
Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior, 72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047.
Ryan, R., Rigby, C. S., & Przybylski, A. (2006). The motivational pull of video games: A self- determination theory approach. *Motivation and Emotion, 30*(4), 344–360. https://doi.org/10.1007/s11031-006-9051-8.
Shute, V. J., Rieber, L., & Van Eck, R. (2011). Games…and…learning. In R. Reiser, & J. Dempsey (Eds.). *Trends and issues in instructional design and technology* (pp. 321–332). Upper Saddle River, NJ: Pearson Education Inc.
Shute, V. J., & Ventura, M. (2013). *Measuring and supporting learning in games: Stealth assessment.* Cambridge, MA: The MIT Press.
Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms. *The Science Teacher, 81*(5), 53.
Tai, R. H., Liu, C. Q., Maltese, A. V., & Fan, X. (2006). Planning early for careers in science. *Science, 312*(5777), 1143–1144.
The White House (2016). Computer science for all. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.
Wang, H., & Sun, C. T. (2011, September). Game reward systems: Gaming experiences and social meanings. *DiGRA conference*.
Weintrop, D., & Wilensky, U. (2012). RoboBuilder: A program-to-play constructionist video game. *Proceedings of the constructionism 2012 conference. Athens, Greece*. Retrieved from http://ccl.sesp.northwestern.edu/papers/2012/DCM_Weintrop.pdf.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215.
Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences, 366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118.
Wing, J. M. (2010). Computational thinking: What and why? Retrieved November 12, 2016, from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.
Wouters, P., van Nimwegen, C., van Oostendorp, H., & van der Spek, E. D. (2013). A meta-analysis of the cognitive and motivational effects of serious games. *Journal of Educational Psychology, 105*(2), 249–265. https://doi.org/10.1037/a0031311.
Yardi, S., & Bruckman, A. (2007). What is computing?: Bridging the gap between teenagers' perceptions and graduate students' experiences. *Presented at the Proceedings of the third international workshop on Computing education research* (pp. 39–50). ACM.