9-14-2021

An Examination of Middle School Student Learner Characteristics as Related to the Reuse and Remixing of Code in Two Different Computer Science Learning Contexts

Tuba Ketenci
Georgia Institute of Technology

Brendan Calandra
Georgia State University

Jonathan Cohen
Georgia State University

Maggie Renken
Georgia State University

Nurjamal Chonoeva
Concept Schools

Recommended Citation

Ketenci, T., Calandra, B., Cohen, J., Renken, M., & Chonoeva, N. (2022). An examination of middle school student learner characteristics as related to the reuse and remixing of code in two different computer science learning contexts. Journal of Research on Technology in Education, 55(6), 986–1002. https://doi.org/10.1080/15391523.2022.2085215

An Examination of Middle School Student Learner Characteristics as Related to the Reuse and Remixing of Code in Two Different Computer Science Learning Contexts

See below for published version:

Ketenci, T., Calandra, B., & Cohen, J., Renken, M., & Chonoeva, N. (2022). An examination of middle school student learner characteristics as related to the reuse and remixing of code in two different computer science learning contexts. Journal of Research on Technology in Education. 55(6), 986-1002. https://doi.org/10.1080/15391523.2022.2085215

Abstract

The goal of this study was to examine how two groups of middle school students' self-efficacy, interest, goal orientation, and prior experience related to evidence of their building upon existing ideas and code in digital artifacts they created using MIT's App Inventor, a computational practice that Brennan & Resnick (2012) identified as "reusing and remixing." Participants included 110 students in a formal computer science education course and 87 students in an after-school computing club. Data sources included a learner profile survey and participants' digital artifacts. Correlational analysis, followed by logistic regression analysis, uncovered significantrelationships between self-efficacy, goal orientation, and evidence of participants' code-oriented reusing and remixing their digital artifacts.

Introduction

Computational thinking (CT) is commonly viewed as a set of higher-order thinking skills based on how computer scientists define and provide solutions to a problem (Cuny et al., 2010; Voogt et al., 2015; Wing, 2011; Wing, 2006). Wing and others (e.g., Yasar, 2018) have maintained that CT is an important skill set not only for those working in computing but also for those in fields where solving problems is a central concern (Lee et al., 2020). This has been followed in recent years by several curricular initiatives to embed CT in K-12 classrooms across disciplines (Dong et al., 2019; National Research Council, 2012; Tsortanidou, Daradoumis, & Barbera-Gregori, 2022), and several studies on the effectiveness of various CT infused interventions (Buitrago Florez, et al. 2017; Grover & Pea, 2013; Lye & Koh, 2014).

Rationale

Although it has been pointed out that CT can be beneficial in a variety of contexts and professions, one effective way to teach and learn CT is through programming activities (Barr & Stephenson, 2011; Grover & Pea, 2013; Kafai & Burke, 2013; Buitrago Florez, et al. 2017). Lye and Koh (2014) reviewed 27 studies on the effectiveness of teaching and learning CT concepts, practices, and perspectives through programming in K-12 contexts. They recommended that future studies examine CT practices in problem-based learning environments that include constructionist approaches (Papert, 1991), such as the scaffolded creation of digital artifacts. For this study, the authors considered CT practices to be those outlined in Brennan and Resnick, "being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing" (2012, p.7). As pointed out by Brennan and Resnick (2012), recognizing students' CT practices is a way to focus on the process of thinking and learning that moves

beyond *what* students are learning (concepts) to *how* they are learning and applying CT concepts to their own programs and designs. Some researchers have argued that the CT practices of reusing and remixing constitute an important digital literacy for today's youth (Jenkins, 2009; Bruckman, 1998). The authors chose to focus on the practice of reusing and remixing in this study for two main reasons: a) because this process requires students to understand and modify existing projects developed by others with various skills levels, backgrounds, and interests, and thus can help them learn new concepts by building upon the knowledge and experience of others (Dasgupta, Hale, Monroy-Hernández, & Hill, 2016); and b) because prior observations of middle school students during app building activities seemed to indicate that they appeared motivated by the ability to remix existing ideas and code segments into their creative works (Calandra, et al. 2020).

Reusing and remixing

While some scholars have investigated the understanding of computing concepts among novice students (Lye & Koh, 2014), and a few have focused on computing practices (Kim et al., 2018), reusing remixing has attracted significant attention due to the large-scale dataset made available online by the Lifelong Kindergarten Group at the MIT Media Lab, a dataset that included Scratch projects developed by users (Dasgupta & Hill, 2017).

Scholars conducting a large-scale analysis of reusing and remixing have primarily focused on the learning patterns of developers (Monroy-Hernández et al., 2011). For example, Dasgupta et al. (2016) investigated how participants' reusing and remixing behavior changed over time and which blocks were common in their projects (2016). Examining 200,000 projects, Xing (2019) described the relationship between cognitive, environmental, and behavioral factors

and students' CT development. Xing demonstrated how participants' reusing and remixing within the online Scratch community contributed to their CT.

Other recent studies involved fewer students and primarily qualitative methods (Fields et al., 2017). Monroy-Hernandez et al. (2011) analyzed Scratch apps and grouped students' reusing and remixing practices based on originality (i.e., duplicates, incremental, inspirational, and component-based) and generalizability (i.e., versioning, crowd-based, and group-based). Similarly, Dahotre, Zhang, and Scaffidi (2010) analyzed 100 projects to examine the effectiveness of Scratch on student gains in technical skill and their development of reusing and remixing behavior. They categorized reusing and remixing behavior into two groups and found that many projects in the selected repository resulted from multimedia-oriented remixing rather than code-oriented reusing and remixing.

For this study, we further categorized evidence of reusing and remixing in participants' activities using Dahotre et al.'s (2010) taxonomy for reusing and remixing practices, dividing remixing practices into two categories based on remixing behavior: code-oriented and multimedia-oriented. Multimedia reusing and remixing refers to changes in the sound or image of the existing app, while no changes happen in the code blocks in the visual block-based programming setting. Code-oriented reusing and remixing refers to the participants' behavior that shows their understanding of computing concepts based on the changes in the code blocks. Because both learning contexts examined in this study were focused on participants learning CT through block-based programming, we focused on code-oriented reusing and remixing.

Learner characteristics and CT

There is evidence that indicates certain learner characteristics such as prior experience, goal orientation, self-efficacy, and interest are related to motivation and academic outcomes in

programming courses (Author et al., 2019; Beyer, 2014; Biggs et al., 2001; Haungs et al., 2012; Lishinski et al., 2016; Wiedenbeck, 2005).

Self-efficacy is an element of social-cognitive theory that refers to beliefs in one's ability to succeed in a situation or a task (Bandura 1997). Numerous scholars have since accepted self-efficacy as the most significant motivational factor linking prior experience to future performance (e.g., Britner & Pajares, 2006; Pajares & Miller, 1994). In line with Bandura's theory, previous findings have shown a strong positive correlation between self-efficacy and student performance in college-level CS courses (Lishinski, et al. 2016; Wiedenbeck, 2005).

Interest is "a psychological state of having an affective reaction to and focused attention for particular content and/or the relatively enduring predisposition to re-engage particular classes of objects, events, or ideas" (Renninger & Hidi, 2002, p. 174). Scholars have categorized interest into two groups: situational and individual (Krapp, 1999; Krapp & Fink, 1992; Schiefele, 1991). Individual interest refers to a person's interest in specific content over time, whereas situational interest arises at the moment, triggered by whatever might be happening (Renninger & Hidi, 2002). Situational interest is crucial in attracting student attention to an activity, but individual interest is critical to holding that attention (Renninger & Hidi, 2002). Previous findings have shown positive correlations between individual interest and student learning of computing concepts (e.g., Beyer, 2014; Haungs et al., 2012). Goal orientation affects student behavior according to the types of outcomes that students desire (i.e., performance or mastery), in turn affecting their performance. Thus, goal orientation might influence CT skill acquisition. According to Elliott and Dweck (1988), performance orientation characterizes individuals who want to demonstrate their competence to others. In contrast, mastery orientation refers to the development of competence for learning. Elliot and McGregor (2001) further divided each

orientation into approach and avoidance, yielding four types of goal orientation: performance approach, performance avoidance, mastery approach, and mastery avoidance. Previous findings have shown a positive relationship between CS learning and mastery goal orientation (Bergin et al., 2015; Zingaro, 2015).

Students come to CS classrooms with a broad range of prior knowledge, skills, and experience (Svinicki, 2004). The content in programming courses typically focuses on a programming language or tool. Prior experience with creative technologies is one way to demonstrate CT skills (ISTE, 2016). Interaction with technology allows students to reflect on their CT skills. Grover, Pea, and Cooper (2016) found that prior experience predicted higher grades in an online programming course for K-12 students. In addition, Beyer (2014) found that students who had positive prior experience were more likely to remain in CS.

Recently, the authors and colleagues examined the relationship between learner characteristics and student outcomes in a middle school computing course using structural equation modeling. The results indicated that 52% of the variance of the 142 middle school participants' success was related to their self-efficacy, interest, and prior computing experience. Student success was measured by a computational thinking quiz and rubric-based evaluations of participants' computing artifacts that covered both CT concepts and practices (Authors, 2019). In a second study, the authors and colleagues "analyzed data from a sample of thirty-two middle school students from an urban school district in the southeastern United States who used MIT's App Inventor to design, create, and remix mobile apps during an afterschool program for one school year. Findings indicated a linear relationship between the number of apps a participant created during the given afterschool program and their level of accuracy on the

assessment" (Authors, 2020. p.17). In addition, participants who created more apps did so in part by reusing and remixing existing code and multimedia in unique app designs.

Purpose

As a follow up to the previous studies and to shed further light on how selected learner characteristics might be related to the CT practice of reusing and remixing, the authors used data collected in two different middle school contexts in the southeastern United States, one during a regular school day and one in an after-school learning environment. This data was collected and analyzed to uncover any relationships between evidence of participants' code-oriented reusing and remixing and their prior experience, goal orientation, self-efficacy, and interest in computing. Participants included middle school girls and boys in grades 6-9, so gender and grade level were considered control variables. Both groups of students participated in similar app-building activities designed to facilitate CT acquisition through programming. Data sources for the current study included a learner profile survey and an examination of reusing and remixing practices based on an analysis of the participants' digital artifacts. The following research question guided the current investigation: "What is the nature of the relationships (if any) between a set of learner characteristics and their code-oriented reusing and remixing practices?"

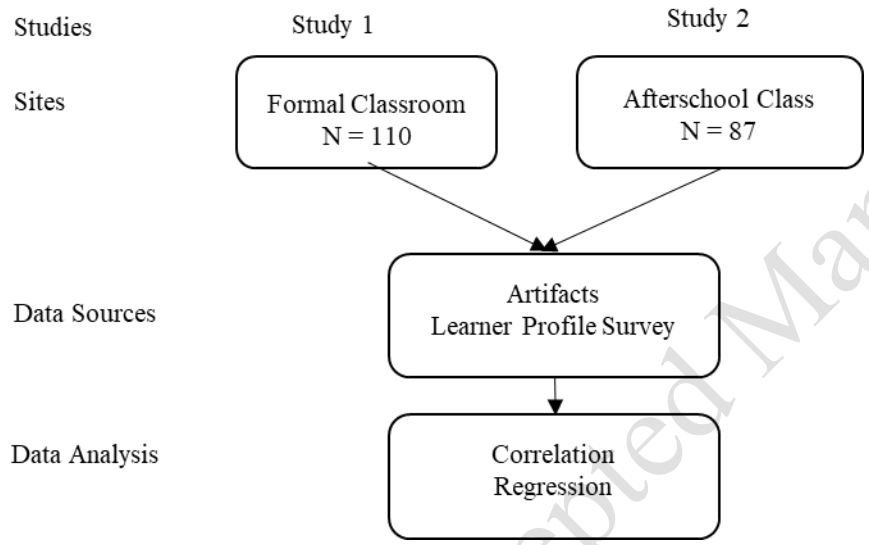
Methods

Results from this study are derived from two related rounds of data collection: Study 1 and Study 2 (see Figure 1). We kept data from the two study sites separate due to differences in the two learning environments described in the next section. Both studies involved a quasi-experimental design due to the naturalistic and dynamic contexts within which data collection occurred (White & Sabarwal, 2014). The site for the first study was a formal classroom that did

not allow for random assignment. The second study took place in an after-school setting in which students self-selected into a computing club with no required attendance.

Figure 1

Data Collection and Analysis for Study 1 and Study 2



Site and Participants

Study 1

We conducted Study 1 in a private, STEM-focused middle school located in a large southeastern United States city, and 166 students participated in the study. Female students accounted for 56.8% of the participants and male students for 29.7%, while 13.5% did not provide gender information. The size of the seven classes participating in this study ranged from 18 to 21 students. Each student in the school was required to attend a computer class within their grade level. Two teachers facilitated our intervention, and both had experience with App

Inventor. Because students followed worked example scripts, the teachers' role was to help students who felt stuck during a project.

Study 2

The participants in Study 2 were predominantly African American middle school students who were participating in a free after-school program operating at multiple middle school sites in a metropolitan area in the southeastern United States. We integrated the intervention into an existing after-school program to foster CT skills through visual, block-based programming for middle school students. In this after-school computing club program, 87 students were able to select from a list of available programs in which to participate. Class size ranged from four to ten students at nine schools under the supervision of nine teachers and mentors and five graduate research assistants, whose role was the same as the teachers in Study 1. Since the study's context was in an informal setting where attendance was not mandatory, there was a fluctuation in participants' attendance. The duration of student participation in Study 2 was explained in detail in the data analysis section.

Intervention and Data Collection

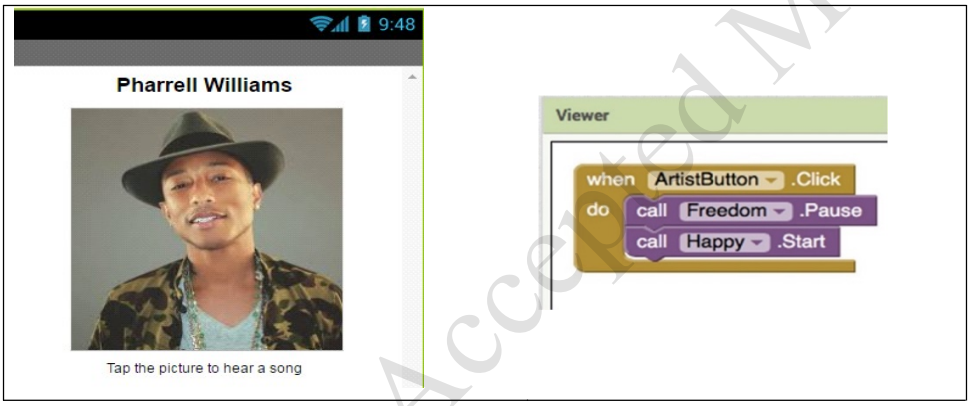
We designed the curriculum to feature gradually more challenging tasks and progressively decreasing amounts of scaffolding (Kirschner et al., 2006). The first set of activities included worked examples, which students used to develop pre-designed apps using App Inventor following step-by-step instructions (Jou et al., 2016). These activities taught computational concepts (i.e., loops, variables, sequence, events, parallelism, operation, conditionals) (Brennan and Resnick, 2012). Upon completing five worked examples, students then worked on an open-ended activity that encouraged applying skills they had learned in the

worked examples to create a more personalized app. Students had the option to work in groups of twos or threes.

Whenever possible, each activity involved a connection between app building and a culturally relevant theme. For example, we created the favorite artist app to appeal to an interest in youth pop culture. In this activity, students learned some of the basics of app programming by creating a soundboard app that plays audio (e.g., sound effects, music) when users press buttons. See Figure 2.

Figure 2

Favorite Artist App



Study 1

Study 1 was an 8-week-long intervention. Weekly sessions were fifty minutes long. The first week was orientation, during which time students took the learner profile survey (Barron et al., 2010). Students studied worked examples between the second and fifth weeks that helped them become familiar with App Inventor. Students then worked on a well-defined, performance-based, open-ended project based on problem-based learning: a mobile app for a given condition. They were also given the option to build an app based on personal interests.

Study 2

Under the supervision of nine teachers and mentors and five graduate research assistants, the after-school program classes met for up to 90 minutes twice a week and with between 4 and 10 participants per class at nine school sites. In this study, students worked on a slightly revised version of the instructional materials used in Study 1, but the underlying CT concepts and practices were the same. Data collection occurred during the first week of the after-school program when 87 students completed a learner profile survey. After the intervention, the online system stored the students' digital artifacts. The app completion rate by Study 2 participants was normally distributed with a mean of 4.6 apps.

Data Sources

Learner Profile Survey

Our survey targeted four constructs from Barron et al. (2010): interest, prior experience with creative computing technologies, self-efficacy, and goal orientation. We also added items for the grade level and gender.

Interest. In Barron et al. (2010), Cronbach's alpha for the items related to student interest in learning about computing technology was 0.83, indicating sufficient reliability (Cronbach, 1951). In the current study, we added three interest-specific, five-point Likert-scale questions to the survey and one additional question regarding interest in App Inventor. The choices for measuring interest in the program were "I cannot wait to get started!" "I am not very interested," and "I do not want to do it." The Cronbach's alpha was .85 in Study 1 and .86 in Study 2, indicating good internal consistency (Hair, 2006). We calculated the average of the four questions to build a continuous interest variable.

Prior experience with creative computing technologies. The Cronbach's alpha for items related to prior experience with creative computing technologies was .86 in Study 1 and .95 in Study 2. Items for prior experience (e.g., "How often have you coded a website using HTML?") featured a five-point Likert scale (e.g., "I have never done this," "I've done this once in my life," "I've done this a few times," "I've done this a lot," and "I don't know what this is."). We calculated students' breadth and depth of experience, described below in more detail (Barron et al., 2010).

Depth of prior experience with creative computing technologies. Students indicated how many times they had participated in each of the thirteen creative activities, and we measured the depth of prior experience based on their responses. For each student, we coded activities that students reported having participated in five or more times as "1" and the rest "0," similar to Barron et al. (2010). Then we calculated a total score for each student by summing the codes, yielding a score between 0 and 13. This overall score indicated the depth of prior experience with creative computing technologies. We employed a median split method to distinguish the students with high depth (coded as "1") from those with low depth (coded as "0").

Breadth of prior experience with creative computing technologies. We computed this variable by summing the number of activities each student had participated in at least once. We again used the median split method to assign students to the "high breadth" group (coded as "1") and the "low breadth" group (coded as "0").

Self-efficacy. We measured self-efficacy using the following survey items: "I feel confident in my ability to learn how to build phone apps." The response options were on a five-point Likert scale (1 = strongly disagree; 5 = strongly agree).

Goal orientation. We asked the following question to assess goal orientation in the intervention: "Select each of the following that describes your goal in learning the App Inventor program." The options included "I want to be better than everyone else in the group" ("GoalBetter" in the current study), "I do not want to fail" ("GoalFail"), "I want to understand how to do this stuff" ("GoalUnderstand") and "I want to have fun" ("GoalFun"). All questions came from the achievement goal theory proposed by Elliot and McGregor (2001). All four variables were coded as "1" if a participant chose the option, "0" otherwise. In Study 1, we presented this construct in a multiple-choice item so that students could pick only one. Each sub-goal variable was dummy coded. However, in Study 2, we presented this measure in a multiple-selection format, allowing students to select more than one. We then coded the binary variables as "1" if that goal orientation was included in the selection and "0" otherwise.

Grade level. Grade level was a categorical variable and coded as "0" for sixth, "1" for seventh, and "3+" for eighth grade.

Gender. We collected student gender information through one survey question. We coded female students "1" and male students as "0" in the dataset.

Reusing and remixing practices found through artifact analysis

To evaluate evidence of participants' code-oriented reusing and remixing, the mobile apps that the participants created at the end of the study were analyzed. The first author of this paper and a research assistant with a bachelor's degree in computer science conducted open thematic coding of the artifacts. The research assistant went through all the curriculum material and completed all the assignments. After the self-training session, they individually analyzed the artifacts in terms of code blocks and multimedia (image and sound) usage. Then, they conducted open thematic coding by comparing each artifact with the projects taught in the curriculum. Both

of them noted the differences and similarities between the projects provided in the curriculum as a written, step-by-step tutorial and the participant's artifact built at the end of the semester. The first round of coding aimed to identify a common trend in the artifacts. The researchers discussed all of the misalignments, eventually establishing 100% agreement.

In the second round of analysis, they individually categorized the apps according to the taxonomy for remixing practices from Dahotre et al. (2010), dividing reusing and remixing practices into two categories: code-oriented or multimedia-oriented. Multimedia oriented (e.g., adding a new image, no changes to code) included changes to the multimedia aspects of the projects with no code manipulation. In contrast, code-oriented (e.g., script changes, bugs fixed) included changes to the code in existing projects.

In this study, 95 apps were analyzed from both studies, with a binary variable called "remixing practices." Participants who demonstrated code-oriented remixing practices in their app were coded as "1" and all others as "0." participants who did not submit an artifact were also coded as "0."

One code-oriented practice found in the participants' apps was modifying or remixing code from previous projects shown in class. Furthermore, in some of the projects, students demonstrated that they applied computational concepts and practices and used an interdisciplinary approach by incorporating scientific or mathematical concepts into their apps. Another code-oriented practice that appeared in the participant artifacts was reusing existing code more efficiently. Figures 3 and 4 illustrate how participants shortened existing code blocks and demonstrated remixing not by copying code but by altering it more efficiently. For example, one student combined code blocks "A" and "B" (see Figure 3) to create code block "C" (see

Figure 4). The last code-oriented practice found in the artifacts was building apps through a more advanced code block not covered in the curriculum (e.g., multiple screens and clock timer).

Figure 3

Code Sample from Calculator Worked Example

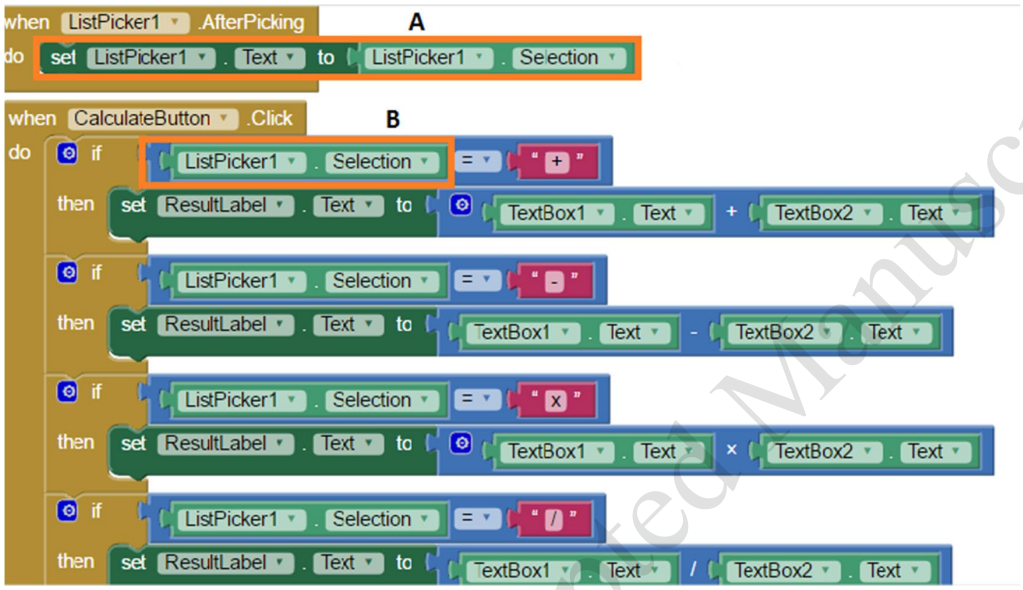
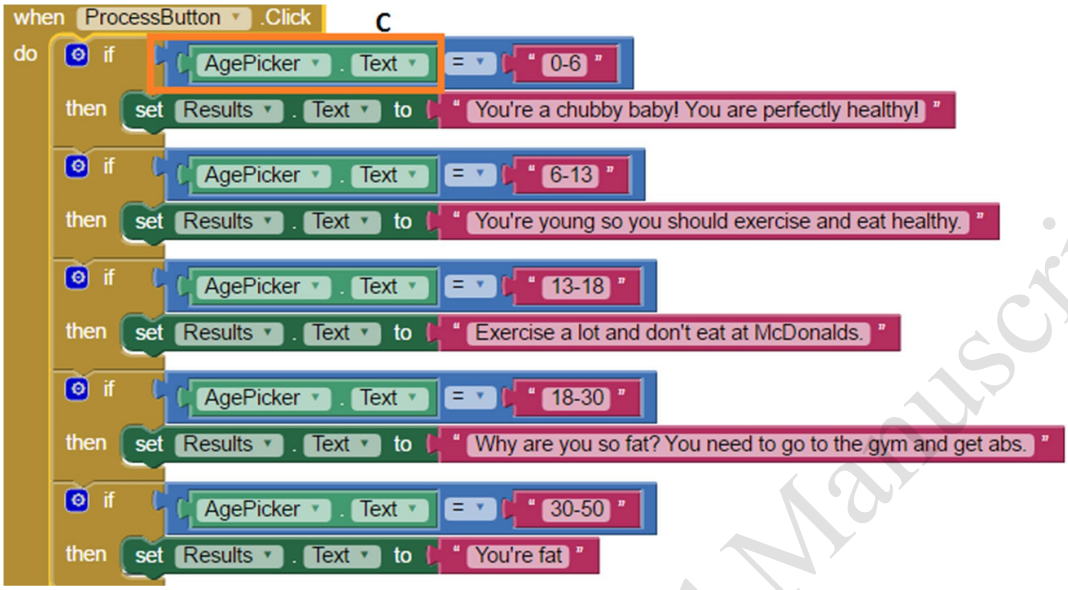


Figure 4

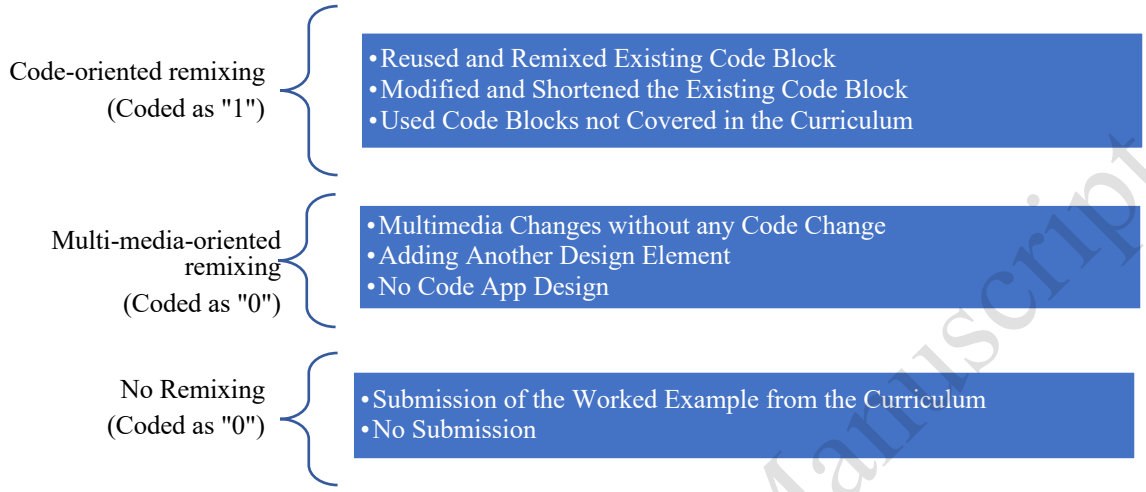
Student Sample *Code*



On the other hand, general trends among students who only demonstrated multimedia or no remixing at all, coded as "0", built apps by changing the appearance of the worked example but not the code, adding the same code block used in the worked example for different design elements, building the app through design elements without any coding, or submitting one of the worked examples without changing anything. These projects did not reflect any changes to the code blocks in the originally worked examples (see Figure 5). Agreement among raters in this study ranged between 95% and 100%, and thus the interrater reliability was deemed within an acceptable range (Israel et al., 2015).

Figure 5

Coding Schema for Students' Digital Artifacts



Data Analysis

We compiled and compared quantitative data to generate insights according to variable- and person-centered models (Roeser et al., 2002). Using the entire sample, a variable-centered model refers to traditional variable-centered correlation techniques for discovering the relationships between independent and outcome variables. Person-centered analysis requires researchers first to identify subgroups within the sample. Researchers can then compare the groups in terms of the outcome variable.

We used SPSS 22 for all the analyses done for this study, including missing data analysis for Study 1 and Study 2 separately (see section 2.4.1; Statistics, 2013). We conducted correlation analyses between the independent variables and the dependent variable to determine which variables were individually related to the dependent variable. Specifically, we conducted two types of correlation analysis: phi coefficient and point-biserial. Phi coefficient correlation

analysis is for examining the relationship between two binary variables. However, we used Pearson's bivariate correlation because previous findings indicate that a Pearson correlation coefficient predicted for two dichotomous variables yields the phi coefficient (Guilford, 1954). Point-biserial correlation analysis is for examining the relationship between categorical and binary variables. Because the point-biserial correlation is mathematically equivalent to the Pearson correlation, we used Pearson in the current study (Linacre and Rasch, 2008).

After determining which variables had significant relationships with the outcome variable individually, we investigated the relationship between the outcome variable and all the variables included in this study using logistic regression analysis. Additionally, we checked the impact of attendance on remixing practices in Study 2 due to high fluctuation in attendance. Students with an attendance rate of 25% or lower demonstrated significantly fewer remixing practices than students with an attendance rate higher than 25%, F (3 127) = 9.036, $p < .001$. We conducted the same analysis with the other three groups (i.e., students with an attendance rate of 25%–50%, 50%–75%, and 75%–100%, respectively). The results revealed no significant difference between the three groups, F (2, 80) = 2.872, $p = .062$, or their remixing practices in Study 2. Therefore, the analysis of the relationship between learner characteristics and remixing practices in Study 2 included only those students with an attendance rate higher than 25%.

Missing Data

Study 1. Study 1 included 166 participating students. Among them, 110 took the learner profile survey. In order to understand the causes of the missing data, we used SPSS to conduct the test developed by Little (1998). The results show that the missing data was completely random (MCAR) for Study 1, $\chi 2$ (1) = 1.431, $p = .232$. No significant relationship between the

missing and existing data points emerged, and the missing data was a random subset of the data. Thus, no further analysis was necessary.

Study 2. A total of 146 students participated in the learner profile survey, attended artifact building sessions, or did both. Among them, 87 participated in the learner profile survey. We used the same statistical procedure described above to examine the missing data for Study 2. The results show that the missing data was completely random for Study 2, $\chi 2$ (1) = .352, p = .553. Thus, no further analysis was necessary.

Results

Descriptive statistics

A total of 110 students in Study 1 and 87 students in Study 2 completed the learner profile survey. Please see the detailed descriptions and the charts represented below for each study.

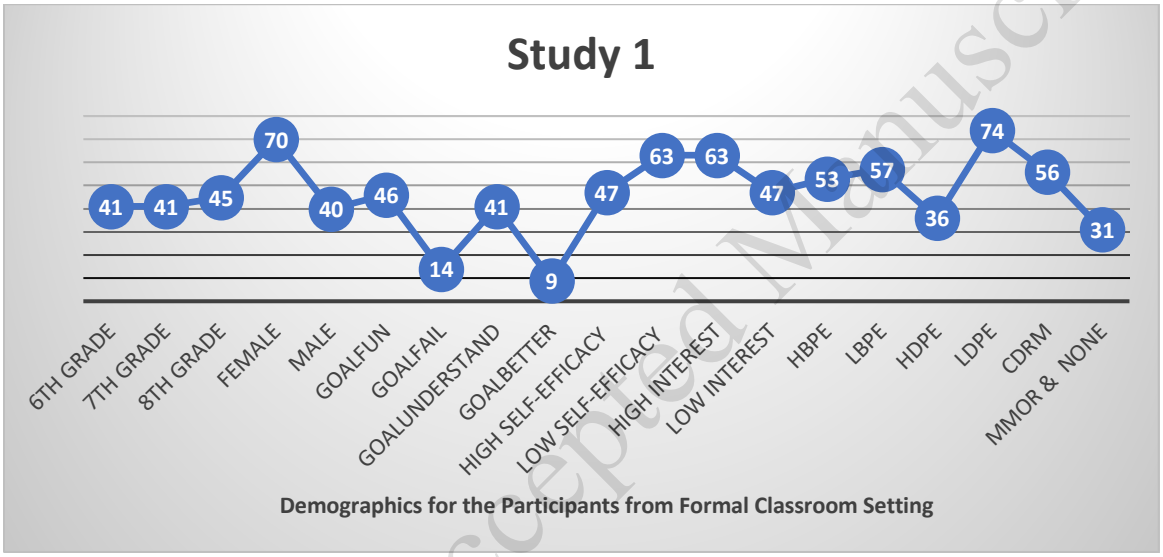
Study 1

In Study 1, 31% of the participants were sixth-graders, 32% seventh graders, and 36% eighth graders. For Study 1 participants who took the learner profile survey, 33% rated their self-efficacy in learning how to build phone apps as “strongly disagree,” 3% “disagree,” 20% “neutral,” 30% “agree,” and 14% “strongly agree .”Furthermore, 36% of them rated their interest in learning about computing technology as "strongly agree", 31% "agree", 17% "neutral", 16% "disagree", and 1% "strongly disagree". Students' prior experience with creative computing technologies was also categorized: 40% of the participants expressed their prior experience as "beginner", 28% "specialist", 10% "explorer", and 23% "generalist". Finally, students’ initial purpose was also obtained through the survey: 8% wanted to be better than their peers, 13% did not want to fail, 37% wanted to understand how the system works, and 42% wanted to have fun.

The other variable was the students' demonstration of remixing practices in their digital artifacts; 61% of the participants did not develop the final artifact or demonstrate remixing practices in their apps (coded as "No"). On the other hand, 39% of the students demonstrated remixing practices in their artifacts (coded as "Yes"). Please see Figure 6 for more details.

Figure 6

Study 1 Demographics



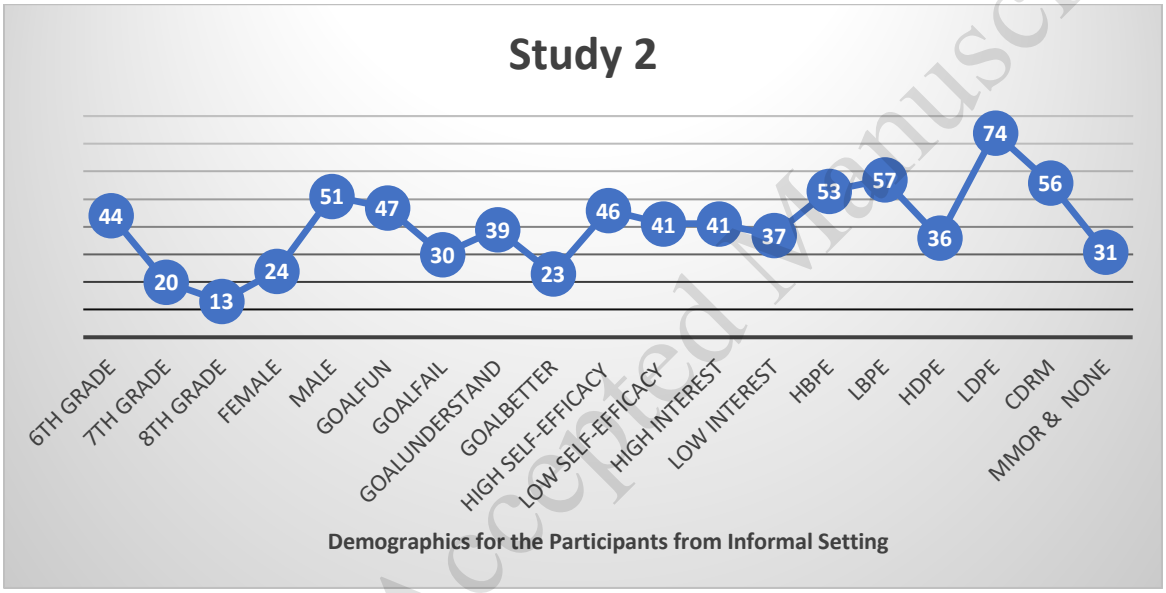
Study 2

Around 36% of the students who took the learner profile survey in Study 2, 9% were sixth-graders, 16% seventh graders, and 6% eighth graders. Two percent of the participants who participated in the learner profile survey rated their self-efficacy in learning how to build phone apps as "disagree," 7% as "neutral," 9% "agree," and 21% as "strongly agree ."Furthermore, 17% of them rated their interest in learning about computing technology as "strongly agree", 20% "agree", 4% "neutral", and 2% "disagree". Students' prior experience with creative computing technologies was also categorized; 4% of the participants expressed their prior experience as "beginner", 2% "specialist", 11% "explorer", and 20% "generalist". Finally,

students' initial purpose was also obtained through the survey; 16% wanted to be better than their peers, 18% did not want to fail, 4% wanted to understand how the system works, and 1% wanted to have fun. 30% of the students used remixing practices in their artifacts, whereas the rest did not. Please see Figure 7 for more details.

Figure 7

Study 2 Demographics



Relationships between learner characteristics and code-oriented reusing and remixing

In order to examine the relationship between learner characteristics and evidence of code-oriented remixing in participants' artifacts, we computed a Pearson product-moment correlation coefficient to examine the correlation between (a) remixing practices (code-oriented vs. multimedia-oriented or none) and (b) learner characteristics (Pearson 1920). The data for Study 1 show that self-efficacy had a significant positive relationship with evidence of code-oriented remixing ($r = .564$, $n = 110$, $p < .001$). On the other hand, gender (male = 0, female = 1) had a significant negative relationship with the dependent variable ($r = -.225$, $n = 110$, $p = .018$),

meaning that male students demonstrated more code-oriented remixing practices than female students.

The results in Study 2 were similar; self-efficacy had a significant positive relationship with code-oriented remixing practices ($r = .224$, $n = 86$, $p = .038$). Furthermore, students who had the goal of understanding how to build an app had a similarly significant relationship with their code-oriented remixing practices ($r = .238$, $n = 77$, $p = .037$). These results indicate that students who wanted to understand code employed more code-oriented remixing practices than their classmates. (see Table 2).

Table 2

Correlations between Learner Characteristics and Code-Oriented Remixing in Both Studies

	Remixing Practices	
	Study 1	Study 2
Age	.197	.172
Prior Depth	-.053	-.061
Prior Breadth	-.113	-.108
Gender	-.225**	-.047
Self-Efficacy	.564**	.224*
Interest	-.025	.091
GoalFail	-.134	-.001
GoalFun	.040	.119
GoalUnderstand	.089	.238*
GoalBetter	-.064	-.025

Variance in evidence of code-oriented reusing and remixing explained by IVs

Study 1

We conducted a logistic regression analysis to predict evidence of participants' code-oriented remixing using all the independent predictors included in this study. The independent variables in binary format included in this analysis were the depth of prior experience, breadth of

prior experience, gender, goal fail, goal fun, goal understanding, and goal better. Other independent variables were age (categorical), self-efficacy (categorical), and interest (continuous). The full model test against a constant-only model was statistically significant, indicating that the variables reliably distinguished participants who employed code-oriented remixing practices in their digital artifacts and those who did not ($\chi 2 = 59.226$, $p < .001$ with df = 9).

Nagelkerke's R^2 of .588 indicates a moderately strong relationship between prediction and grouping. Prediction success overall was 78% (77% for students with multimedia-oriented remixing practices and 79% for students with code-oriented remixing practices). The Wald criterion revealed that self-efficacy ($p < .001$), interest ($p = .005$), prior experience breadth ($p = .027$), goal_understand ($p = .033$) and gender ($p = .004$) made a significant contribution to prediction at the alpha level .05. Moreover, the Cox and Snell R Square shows that the model explained 44% of the variance in the outcome variable.

The Exp(B) value associated with self-efficacy was 3.443, with students' interest level was .332, with prior experience breadth was 5.179, with the goal of understanding how to build mobile applications was 9.169, and with gender was 6.293. One individual unit increase in these variables increases the likelihood of using code-oriented remixing behavior as times as their Exp (B) value. For example, when self-efficacy increased by one unit while there was no change in the other variables, the odds ratio was 3.443 times as large, indicating that students with high self-efficacy levels were 3.443 times more likely to demonstrate code-oriented remixing practices in their digital artifacts than their fellow students in the same learning environment. The same interpretation of the self-efficacy variable is valid for the gender variable.

Although the independent variables interest, prior experience breadth, and the goal of understanding how to build an app did not significantly correlate with the dependent variable, their Exp (B) were statistically significant. This result showed that these variables were suppressors in the model; they raised the R-square due to its accounting for the residuals, not having a strong relationship with the outcome variable. See Table 3. The interpretation of these metrics will be discussed in the discussion section below. All assumptions of logistic regression (independence of errors, linearity in the logit for continuous variables, absence of multicollinearity, and lack of strongly influential outliers.) were checked, and no issues were found.

Table 3

Summary of Logistic Regression Analysis for Learner Characteristics Predicting Remixing Practices in Study 1

Predictor	*B*	SE *B*	Wald	Sig.	*e*B
Self-efficacy	1.236	.286	18.742	.000*	3.443
Interest	-1.102	.395	7.797	.005*	.332
Prior Experience					
Depth	.100	.681	.022	.883	.905
Breadth	-1.645	.742	4.910	.027*	.193
Goal					
Fun	1.385	1.023	1.833	.176	3.995
Better	2.298	1.424	2.603	.107	9.950
Understand	2.216	1.037	4.566	.033*	9.169
Age	.564	.337	2.800	.094	1.757
Gender	-1.839	.634	8.420	.004*	6.293

Note. *$p < .05$.

Study 2.

We repeated the same analyses for Study 2. The test of the full model against a constant-only model was statistically significant, indicating that the model reliably distinguished students who used code-oriented remixing practices in their digital artifacts from those who did not ($\chi 2$ = 18.561 p = .046 with df = 10). See Table 4.

Table 4

Summary of Logistic Regression Analysis for Learner Characteristics Predicting Remixing Practices

Predictor	*B*	SE *B*	Wald	Sig.	*e*B
Self-efficacy	1.117	.667	2.801	.094	3.055
Interest	-.554	.563	.968	.325	.575
Prior Experience					
Depth	-.131	.809	.026	.871	.877
Breadth	-1.202	1.335	.811	.368	.301
Goal					
Fun	2.538	1.326	3.661	.056	12.655
Better	1.135	.915	1.541	.214	3.112
Understand	1.213	.919	1.740	.187	3.362
Fail	-1.616	.936	2.980	.084	.199
Age	-.177	.409	.188	.664	.837
Gender	.727	.802	.821	.365	2.068

Note. *$p < .05$.

Nagelkerke's R^2 of .393 indicates a moderate relationship between prediction and grouping. Prediction success overall was 98% (81.3% for students with multimedia-oriented remixing practices in their apps and 31% for students with code-oriented remixing practices). The Cox and Snell R Square shows that the specified model explained 25% of the variance in the outcome variable. Although the logistic regression model was significant, the Wald criterion revealed that none of the variables significantly contributed to the prediction. Power analysis

showed that the sample size was not large enough to reach a power of .80 at an alpha level of .05.

Limitations

Although some patterns emerged in our findings of middle school students' remixing practices in the computing contexts, a few limitations restrict the broader conclusions and implications of the current study, including research design and data collection. Because the focus of this study was a correlation (i.e., regression analysis), we cannot conclude causation. For this reason, scholars and practitioners should interpret the results cautiously. Scholars can use these results to know which constructs warrant further attention, perhaps leading to experimental studies that enable causal conclusions. Scholars should also assess students' self-efficacy in CS learning contexts more rigorously with that goal in mind. In addition, qualitative data collection did not occur in the formal context in Study 1, and field notes collected in the after-school environment in Study 2 did not focus on remixing practices or individual learner characteristics. Qualitative data may have provided a richer understanding of individual differences and would be recommended for future studies.

While some of the learner characteristics, such as interest, the goal of understanding how to build an app, and breadth of prior experience with creative technologies, explained some of the variances in our model, there was no significant association between the learner characteristic and the code-oriented remixing, possibly due to small sample size. Future studies should continue to examine these relationships with larger sample sizes. If clustered in levels, the large sample sizes may also give a broader picture than regression analysis. Although our data was structured in levels, the sample size was insufficient to run advanced statistical analyses such as hierarchical linear modeling (HLM). Although there is no consensus on how large the sample

should be for HLM, as a rule of thumb, HLM is suggested if the data is collected from at least 30 schools at Level 2 for an accurate estimation of standard errors and is strongly not recommended with less than 10 Level 2 units (Hoyde & Gottfredson, 2015; Maas and Hox 2005, Snijders and Bosker 2011).

Discussion and Implications

In sum, this study aimed to understand whether a set of learner characteristics relate to a group of middle school students' use of code-oriented remixing in a visual block-based computing context. Part of the rationale for this endeavor was to provide implications for both CT instruction and CT assessment. The research question addressed the nature of the relationships between a set of middle school students' learner characteristics and a particular CT practice: code-oriented remixing. We measured these relationships by analyzing digital artifacts produced by participants in both a formal learning environment and an after-school learning environment. See Table 5.

Table 5

Summary of Overall Findings

	Setting			
	Formal Classroom Context (Study 1)		After-School Context (Study 2)	
Variable of Interest	Variables Correlated with Outcome	Significant Variables in Regression	Variables Correlated with Outcome	Significant Variables in Regression
Remixing Practices	Self-efficacy Gender	Self-efficacy Gender	Self-efficacy Goal Approach	None

Correlation analysis revealed that students' self-efficacy significantly correlated with evidence of code-oriented remixing practices in their digital artifacts, regardless of the learning

context. This finding aligns with previous results showing the positive effect of self-efficacy on programming performance (Cigdem and Oncu, 2015; Lishinski et al., 2016; Wiedenbeck, 2005).

One of the takeaways of this study is that among all of the variables, self-efficacy is most strongly related to the use of code-oriented remixing practices as measured by artifact analysis in both formal classroom and after-school settings. Scholars should consider conducting an experimental study to identify potential causal links between the two variables despite this strong relationship. Our finding that self-efficacy related to code-oriented remixing practices in *both* learning environments also suggests designers and teachers should pay special attention to self-efficacy during CT instruction. One strategy for increasing self-efficacy is incorporating explicit instruction in problem-solving. The instructional design for the current study involved direct instruction followed by more open-ended activities, allowing students to build upon the code they learned about through worked examples. It would be interesting for future studies to examine this process more closely to develop a more fine-grained understanding of how self-efficacy changes for specific activities or over time during an intervention, as suggested by Lishinski et al. (2016).

In addition to self-efficacy, goal orientation and code-oriented remixing practices were related in the after-school setting but not in the formal classroom setting. Those students whose goal orientations included understanding how to build an app demonstrated code-oriented remixing practices in their apps more than those without that goal orientation. This finding aligns with previous results showing differences in student performance based on goal approach (Zingaro, 2015). One possible explanation for the positive correlation between code-oriented remixing practices and the mastery goal approach is that students with performance goals tend to put more effort into handling the tasks to avoid looking incompetent rather than mastering the

content to acquire new skills (Zarankin, 2008). This finding suggests that instructional designers should reinforce mastery goal orientation at the beginning of an intervention. One strategy would be to give students agency through collective goal setting. As Bandura (1997) argued, participation in collective goal setting helps with students' goal setting and increases self-efficacy and performance (1997).

Both self-efficacy and goal orientation stood out among the other variables (e.g., interest and prior experience with creative computing), suggesting that self-regulated learning theory might be a helpful lens for understanding the causal relationships between these two variables and CT practices (Lishinski et al., 2016). Instructional design strategies in self-regulated learning literature might be applicable because self-regulated learning theory posits that self-efficacy and goal setting are essential predictors of learning outcomes (Zimmerman, 2013).

Results also showed that male students in the formal classroom setting tended to demonstrate more code-oriented remixing behavior than female students; however, we did not see the same trend in the after-school environment, where students self-selected into the computing club. In this case, no significant correlation between gender and evidence of code-oriented remixing emerged. This difference might be due to greater student interest overall among those in the afterschool computing club, although students in the after-school program tended to produce less code-oriented remixing overall in their artifacts than students in the formal classroom. This outcome was likely due to the nature of each setting. Because comparison of the settings was outside the scope of the current study, scholars should continue to investigate the relationship between gender and contextual differences in CT learning contexts.

Results from this study demonstrated a significant relationship between self-efficacy, goal orientation, and evidence of code-oriented remixing practices among a group of middle

school girls and boys in two different learning contexts. It is hoped that this finding can be useful for both researchers and practitioners interested in instructional design for K12 Computer Science.

References

Atmatzidou, S., and Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661-670.

Balanskat, A., and Engelhardt, K. (2014). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet.

Bandura, A. (1997). Self-efficacy: The exercise of control: Macmillan.

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

Barron, B., Walter, S. E., Martin, C. K., and Schatz, C. (2010). Predictors of creative computing participation and profiles of experience in two Silicon Valley middle schools. *Computers and Education*, *54*(1) 178-189.

Bergin, S., Mooney, A., Ghent, J., and Quille, K. (2015). Using machine learning techniques to predict introductory programming performance. *International Journal of Computer Science and Software Engineering (IJCSSE)*, *4*(12), 323-328.

Beyer, S. (2014). Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education* 2*4*(2-3) 153-192.

Biggs, J., Kember, D., and Leung, D. Y. (2001). The revised two-factor study process questionnaire: R-SPQ-2F. *British journal of educational psychology*, *71*(1) 133-149.

Blanchard-Fields, F., Jahnke, H. C., and Camp, C. (1995). Age differences in problem-solving style: The role of emotional salience. *Psychology and aging* 1*0*(2) 173.

Brennan, K., and 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,

Britner, S. L., and Pajares, F. (2006). Sources of science self-efficacy beliefs of middle school students. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, *43*(5), 485-499.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834-860.

Calandra, B., Renken, M., Cohen, J., Hicks, T., and Ketenci, T. (2020). An examination of a group of middle school students' engagement during a series of afterschool computing activities in an urban school district**.** *Tech Trends. 65,* 17-25.

Cigdem, H., and Oncu, S. (2015). E-Assessment Adaptation at a Military Vocational College: Student Perceptions. *Eurasia Journal of Mathematics, Science and Technology Education* 1*1*(5).

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *psychometrika* 1*6*(3) 297-334.

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

Bruckman, A. (1998). Community support for constructionist learning. *Computer Supported Cooperative Work (CSCW)*, *7*(1-2), 47-86.

Dasgupta, S., Hale, W., Monroy-Hernández, A., and Hill, B. M. (2016, February). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work and Social Computing* (pp. 1438-1449).

Dasgupta, S., & Hill, B. M. (2017, April). Learning to code in localized programming languages. In *Proceedings of the fourth (2017) ACM conference on learning@ scale* (pp. 33-39).

Dong, Y., Catete, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., ... & Andrews, A. (2019, February). PRADA: A practical model for integrating computational thinking in K-12 education. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 906-912).

Dahotre, A., Zhang, Y., and Scaffidi, C. (2010, September). A qualitative study of animation programming in the wild. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-10).

Durak, H. Y., and Saritepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers and Education* 1*16* 191-202.

Elliot, A. J., and McGregor, H. A. (2001). A 2× 2 achievement goal framework. *Journal of personality and social psychology*, *80*(3), 501.

Elliott, E. S., and Dweck, C. S. (1988). Goals: An approach to motivation and achievement. *Journal of personality and social psychology*, *54*(1), 5.

Equation, C. t. (2016). *New data: Bridging the computer science access gap*. http://changetheequation.org/blog/new-data-bridging-computer-science-access-gap-0

Foundation, N. S. *Women, minorities, and persons with disabilities in science and engineering: 2017*. www.nsf.gov/statistics/wmpd/

Govender, I., Govender, D. W., Havemga, M., Mentz, E., Breed, B., Dignum, F., and Dignum, V. (2014). Increasing self-efficacy in learning to program: exploring the benefits of explicit instruction for problem solving. *TD: The Journal for Transdisciplinary Research in Southern Africa* 1*0*(1) 187-200.

Grover, S., and Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational researcher*, *42*(1), 38-43.

Grover, S., Pea, R., and Cooper, S. (2016). Factors influencing computer science learning in middle school. Proceedings of the 47th ACM technical symposium on computing science education,

Guilford, J. P. (1954). Psychometric methods.

Hair, J. F. (2006). *Multivariate data analysis*. Pearson Education India.

Haungs, M., Clark, C., Clements, J., and Janzen, D. (2012). Improving first-year success and retention through interest-based CS0 courses. Proceedings of the 43rd ACM technical symposium on Computer Science Education,

Henderson, P. B., Cortina, T. J., and Wing, J. M. (2007). Computational thinking. *ACM SIGCSE Bulletin*, *39*(1) 195-196.

Hoyle, R. H., & Gottfredson, N. C. (2015). Sample size considerations in prevention research applications of multilevel modeling and structural equation modeling. *Prevention Science*, *16*(7), 987-996.

Israel, M., Wherfel, Q. M., Shehab, S., Melvin, O., and Lash, T. (2017). Describing elementary students interactions in K-5 puzzle-based computer science environments using the

collaborative computing observation instrument (c-coi). Proceedings of the 2017 ACM Conference on International Computing Education Research,

ISTE. (2016). *ISTE standards for students*. https://www.iste.org/resources/product?id=3879&childProduct=3848

Jou, M., Tennyson, R. D., Wang, J., and Huang, S.-Y. (2016). A study on the usability of E-books and APP in engineering courses: A case study on mechanical drawing. *Computers and Education*, *92* 181-193.

Kirschner, P. A., Sweller, J., and Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, *41*(2), 75-86.

Krapp, A. (1999). Interest, motivation and learning: An educational-psychological perspective. *European journal of psychology of education* 14(1) 23-40.

Krapp, A., and Fink, B. (1992). The Development and function of interests during the critical transition from home to preschool. *The role of interest in learning and Development*, 397-429.

Kreft, I. G. (1996). Are multilevel techniques necessary? An overview, including simulation studies. *Unpublished manuscript, California State University, Los Angeles*.

Lee, I., Grover, S., Martin, F., Pillai, S., & Malyn-Smith, J. (2020). Computational Thinking from a Disciplinary Perspective: Integrating Computational Thinking in K-12 Science, Technology, Engineering, and Mathematics Education. *Journal of Science Education and Technology*, 29(1), 1–8.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads* 2(1), 32-37.

Linacre, J., and Rasch, G. (2008). The expected value of a point-biserial (or similar) correlation. *Rasch Meas Trans* *22*(1) 1154-1157.

Lishinski, A., Yadav, A., Good, J., and Enbody, R. (2016). Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. Proceedings of the 2016 ACM Conference on International Computing Education Research,

Little, J. R. (1998). Missing data. In *Biostatistics*: John Wiley and Sons.

Margolis, J., Estrella, R., Goode, J., Holme, J. J., and Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing*. MIT Press.

Mark, J., and Hanson, K. (1992). Beyond Equal Access: Gender Equity in Learning with Computers. *Women's Educational Equity Act Publishing Center Digest*.

Maas, C. J., & Hox, J. J. (2005). Sufficient sample sizes for multilevel modeling. *Methodology*, *1*(3), 86-92.

Mills, C. J., Ablard, K. E., and Stumpf, H. (1993). Gender differences in academically talented young students' mathematical reasoning: Patterns across age and subskills. *Journal of Educational Psychology*, *85*(2), 340.

National Research Council. (2012). A framework for K-12 science edu- cation: Practices, crosscutting concepts, and core ideas. Washington, DC: The National Academies Press. https://doi.org/ 10.17226/13165.

Pajares, F., and Miller, M. D. (1994). Role of self-efficacy and self-concept beliefs in mathematical problem solving: A path analysis. *Journal of Educational Psychology*, *86*(2) 193.

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex.

Pearson, K. (1920). Notes on the history of correlation. *Biometrika* 1*3*(1) 25-45.

Renninger, K. A., and Hidi, S. (2002). Student interest and achievement: Developmental issues raised by a case study. In *Development of achievement motivation* (pp. 173-195). Elsevier.

Robertson, J. (2013). The influence of a game-making project on male and female learners' attitudes to computing. *Computer Science Education* 2*3*(1), 58-83.

Roeser, R. W., Strobel, K. R., and Quihuis, G. (2002). Studying early adolescents' academic motivation, social-emotional functioning, and engagement in learning: Variable-and person-centered approaches. *Anxiety, Stress and Coping* 1*5*(4), 345-368.

Román-González, M., Pérez-González, J.-C., Moreno-León, J., and Robles, G. (2016). Does computational thinking correlate with personality?: the non-cognitive side of computational thinking. *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality.*

Schiefele, U. (1991). Interest, learning, and motivation. *Educational psychologist* 2*6*(3-4) 299-323.

Snijders, T. A., & Bosker, R. J. (2011). *Multilevel analysis: An introduction to basic and advanced multilevel modeling*. sage.

Statistics, I. S. (2013). IBM Corp. *Released IBM SPSS Statistics for Windows, Version* 2*2*.

Svinicki, M. D. (2004). *Learning and motivation in the postsecondary classroom*. Anker Publishing Company.

Voogt, J., Fisser, P., Good, J., Mishra, P., and Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies* 20(4), 715-728.

Watson, C., and Li, F. W. (2014). Failure rates in introductory programming revisited. Proceedings of the 2014 conference on Innovation and technology in computer science education,

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. Proceedings of the first international workshop on Computing education research,

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine* 20-23.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., and Korb, J. T. (2011). Introducing computational thinking in education courses. *Proceedings of the 42nd ACM technical symposium on Computer science education.*

Yasar, O. (2018). Computational Thinking, Redefined. Society for Information Technology and Teacher Education International Conference,

Zarankin, T. G. (2008). A new look at conflict styles: goal orientation and outcome preferences. *International Journal of Conflict Management* 19(2) 167-184.

Zimmerman, B. J. (2013). Theories of self-regulated learning and academic achievement: An overview and analysis. In *Self-regulated learning and academic achievement* (pp. 10-45). Routledge.

Zingaro, D. (2015). Examining interest and grades in Computer Science 1: a study of pedagogy and achievement goals. *ACM Transactions on Computing Education (TOCE)* 15(3) 14.

Jenkins, H. (2009). *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press.

Kim, C., Yuan, J., Vasconcelos, L., Shin, M., and Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, *46*(5), 767-787.

Monroy-Hernández, A., Hill, B. M., Gonzalez-Rivero, J., and Boyd, D. (2011, May). Computers can't give credit: How automatic attribution falls short in an online remixing community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 3421-3430).

Dahotre, A., Zhang, Y., and Scaffidi, C. (2010, September). A qualitative study of animation programming in the wild. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-10).

Fields, D. A., Kafai, Y. B., Nakajima, T., and Goode, J. (2017, October). Teaching practices for making e-textiles in high school computing classrooms. In *Proceedings of the 7th Annual Conference on Creativity and Fabrication in Education* (pp. 1-8).

Margulieux, L., Ketenci, T. A., and Decker, A. (2019). Review of measurements used in computing education research and suggestions for increasing standardization. *Computer Science Education* *29*(1), 49-78.

White, H., and Sabarwal, S. (2014). *Quasi-Experimental Design and Methods: Methodological Briefs-Impact Evaluation No. 8* (No. innpub753).

Xing, W. (2019). Large-scale path modeling of remixing to computational thinking. *Interactive Learning Environments* 1-14.

Tsortanidou, X., Daradoumis, T., & Barberá-Gregori, E. (2022). Unplugged computational thinking at K-6 education: evidence from a multiple-case study in Spain. *Education 3-13*, 1-18.

Appendix A

Learner Profile Survey Sample Items

1. Which one of these sentences best describes your feelings about the app-building activities you have signed up to work on?
    a. I'm not very interested in the task. I don't want to do it.
    b. My interest is "so-so." I can take it or leave it.
    c. I can't wait to get started!
2. Select each of the following that describes your goals?
    a. I want to better at this than everyone else in the group.
    b. I don't want to fail.
    c. I want to understand how to do this stuff.
    d. I want to have fun.

Profile of Confidence and Interest

Rate the following (1–5; 1 = strongly disagree; 5 = strongly agree):

1. I am good with computers.
2. I would like to learn more about computers.
3. Learning about what computers can do is fun.

4. It is important to me that I am knowledgeable about computers.

Profile of Creative Production Experience

How often have you participated in the following (never, once or twice, three to six times, and more than six times)

5. Creation of multimedia
6. Programming
7. Creation of art
8. Publication generation using desktop publishing program
9. Starting a newsgroup
10. Building robots or other technological inventions
11. Coding web sites using HTML
12. Generating web sites using an application
13. Publishing a web site
14. Using a computer to stimulate or model phenomena
15. Designing with a CAD program
16. Developing a database
17. Creating a digital movie
18. Creating an animation
19. Creating a computer game
20. Creating a piece of music