ABSTRACT

Use-Modify-Create (UMC) has gained recognition as a viable scaffolding approach for student programming activities, but little is known about how UMC could support CS learning in game-based learning environments. We designed and developed a game to teach middle grade students (ages 11-13) CS through block-based programming challenges. The game integrates a UMC pedagogical framework to promote successful student outcomes for a wide variety of student abilities, including those without prior programming experience. Utilizing a mixed-methods research design, we investigated how the game influenced student learning of CS concepts and the role of UMC on the problem-solving strategies students applied to complete the game. In particular, we were interested in how prior experience would moderate these outcomes. Results from a multilevel model of students’ pre- and post-assessment scores (N = 77) on a CS concepts assessment indicated that all students, regardless of prior programming experience, showed significant learning gains from pre to post after playing the game. Qualitative results revealed that the UMC scaffolding progression provided students, particularly those with little to no prior programming experience, with the foundational knowledge needed to progress through the game levels and challenges. Specifically, we found that the UMC phases of the game reduced novice students’ cognitive load and facilitated the necessary CS conceptual understanding to solve the open-ended programming tasks encountered in the game’s Modify and Create phases. Our findings demonstrate the efficacy of UMC to support the learning of novice programmers in a game-based learning environment while not to the detriment of those more experienced.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ITiCSE 2021, June 26–July 1, 2021, Virtual Event, Germany
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8214-4/21/06...$15.00
https://doi.org/10.1145/3430665.3456349

1 INTRODUCTION

Computational thinking (CT) and computer science (CS) are now recognized as essential skills and practices necessary for productive participation in a global society [15]. Thus, CS education has become critically important for K-12 students, as they must be afforded opportunities to learn and develop these skills throughout their formal schooling [42]. One increasingly popular mechanism for developing K-12 students’ CT and CS skills has been through computer programming activities [4, 19]. Over the past decade, tools such as Scratch and Alice have been integrated into school course designs and curricula to teach CS concepts to K-12 learners [2, 43].

Middle grades (ages 11-13) have been identified as a pivotal point in students’ educational trajectories where they should have ample opportunities to learn and develop favorable dispositions towards STEM domains such as CS [16, 44]. However, middle school students engaging in CS-related activities such as programming often possess wide variability in their prior knowledge and experience [6, 18]. This highlights the importance to developing pedagogical strategies and learning environments that are inclusive and engage novice learners, especially those from populations historically marginalized from participation in CS-related activities [12, 33].
The potential of digital game-based learning to promote increased student engagement and learning outcomes has received considerable research interest across disciplines (e.g.,[23, 37]). This line of research has demonstrated that digital games can facilitate enhanced learning outcomes for students [10, 45], as well as increased motivation for learning when compared with traditional instruction [9, 30]. In particular, immersive game-based learning that integrates socio-constructivist learning principles with problem-based scenarios can offer students rich and engaging learning experiences [26]. However, benefits from digital game-based learning are often dependent upon learning environment design strategies that appropriately manage the level of challenge players experience by including scaffolding and support during gameplay [8, 22]. The integration of appropriate learner support mechanisms within digital games is important to ensure that learners stay motivated and avoid frustration [39, 45].

Our research team recently developed an immersive game-based learning environment (GBLE) designed to develop middle school students’ CT practices, CS concepts, and programming skills. To support a wide range of prior experience in programming we embedded a Use-Modify-Create (UMC) scaffolding progression within the game design [25]. This pedagogical approach was employed to ensure that all students, including those with little to no prior CS or programming experience, could successfully engage with the game and learn targeted CS concepts [25]. While UMC strategies have been popular with general classroom-based programming activities, research into its efficacy is still in its early stages, and even less research has been conducted on its implementation in GBLEs. This paper investigates the efficacy of a UMC progression within a GBLE to teach middle school students CS and programming concepts. The following research questions guided the investigation:

1. Do middle school students learn CS concepts through a CS-focused game-based learning environment that integrates a UMC framework accounting for their prior experience?
2. How does a UMC progression integrated within a GBLE support middle school students’ problem solving and learning of CS concepts?

2 RELATED WORK

2.1 UMC Progression

UMC is an increasingly popular pedagogical framework for supporting CT and CS [24, 28, 38]. Originally proposed by Lee et al., UMC is a three-stage progression designed to scaffold learners as they gradually encounter increasingly complex CS learning activities [25]. The approach often alleviates potential cognitive demands and anxieties that can confront learners new to programming and CS learning. In the Use phase, students are introduced and encouraged to inspect a pre-made program that they run to accomplish a computational task. The Use phase is a highly scaffolded scenario that enables students to explore and become familiar with key CS and programming concepts, gradually building their confidence and competence. The Modify phase is characterized by a computational task that challenges students to utilize their newly acquired knowledge to make modifications to an existing or partially completed computational artifact to accomplish a design task. This phase is considered an interstitial step where scaffolds are faded as students explore how small refinements in code affect programs. Finally, in the Create phase students are presented with a completely open-ended computational task in which they apply their new understandings to develop a new computational artifact. The goal is that students are able to purposefully reflect upon and apply conceptual understanding and programming actions taken in the Use and Modify phases to create their own program.

2.2 Prior Research on the Use of UMC

Within the past couple of years, the CS education community has witnessed an increased focus on the development and research of K-8 CS/CT-related activities that utilize UMC as a pedagogical framework (e.g., [1, 35]). Results from this work have delivered promising evidence that the approach is efficacious for students, particularly those who lack prior programming experience. For example, Lytle et al. conducted a quasi-experimental study with a UMC condition and control group for a four-day CT-infused science lesson where students created a Food-Web simulation using the Snap! programming environment [28]. Students in the UMC group did not experience a detrimental gradation of difficulty in the lesson progression when compared with the control group and reported an increased sense of ownership of their computational artifacts. Franklin et al. analyzed the impact of the UMC approach to support student learning of CS concepts within a Scratch curriculum [11]. They found that UMC promoted student learning of CS concepts with the introduction of specific programming blocks and concepts during Use and Modify phases, which enabled students to extend their new knowledge within the Create phase. One way of generalizing findings to date is that UMC supported an optimal balance of scaffolding and challenge necessary to move students through their Zone of Proximal Development [11, 41].

The use of a UMC scaffolding progression has been extended to support students’ CT experiences beyond a programming-only context. Grizioti and Kynigos incorporated scaffolding based on UMC to support students’ CT learning and the creation of digital games using “game modding” techniques that enabled students to first play and inspect an already made game, then modify a “half-baked” version with “buggy behavior,” and finally create their own game designs [13]. The authors found that when all three stages were embedded within the game design approach, it provided a rich context for facilitating CT skill development. We were inspired by this earlier work to incorporate the UMC framework into the design of our digital GBLE to support students’ understanding of CS and programming concepts. To our knowledge, while popular in the K-8 grades, the UMC scaffolding framework has not been integrated into a GBLE specifically designed for promoting middle school students’ CT and CS learning with block-based programming. Thus, we found this context to be a unique opportunity to investigate the efficacy of such an approach for supporting students’ CS learning.

2.3 Elements of Effective GBLE

Although GBLEs hold enormous potential for increased student learning in a variety of subject areas (e.g., [7, 34]), prior research indicates that game developers must consider the intricate balance between challenge and skill [17]. Hamari et al. conducted a path analysis to investigate the relationship between students’ perceived
challenge, skill, engagement, immersion, and learning [17]. Their findings indicated that challenge, skill, and engagement all had significant positive effects on learning and concluded engagement can be promoted with optimal levels of skill and challenge [17]. Israel-Fishelson and Hershkovitz recently studied the relationship between persistence and difficulty in a CT-focused GBLE for elementary students [20]. They found a positive relationship between the two, however, they caution that the burden is on game developers to ensure an alignment between learners’ abilities and the challenges that they are tasked with solving. The Game Development for Computer Science Education working group met at ITiCSE ‘16 to promote the use of GBLE for CS education [21]. In alignment with the findings noted above, they asserted that students need an optimal balance of challenge and support to stay motivated and engaged within these learning environments. Therefore, they emphasized the importance of design elements such as engagement, differentiated instruction, and deliberate practice. In addition to advocating for use of best practices and educational theory to inform CS-focused GBLEs, they also underscored the need for evaluation of these resources to determine their efficacy.

2.4 The ENGAGE GBLE
The ENGAGE GBLE was designed to promote CT as well as broadening interest in CS for middle grades students. In the game, students assume the role of a protagonist who has been sent to rescue an undersea research facility whose computing infrastructure has been commandeered by a nefarious researcher. Students are tasked with navigating through a series of interconnected rooms within the undersea research station, in which they are presented with computational programming challenges they must solve using a block-based programming interface to control devices within the rooms. The UMC framework informed the design of the game and the progression of the coding challenges, while the CS concepts addressed in each challenge was guided by the CS Focal Skills, Knowledge, and Abilities (FKSA) framework [14]. The three thematic levels of the game (loops, variables, and conditionals) were designed to facilitate CS and CT practices denoted in the FKSA framework via a broad range of programming activities requiring abstraction and algorithmic thinking. In order to provide students with a UMC progression for each CS concept, students begin each mission in the game by operating devices with pre-written programs for them to use and examine. Then, students encounter a similar device with partially correct code for them to complete. Finally, students are presented with a new challenge in which they have to create their own code to solve a new problem. During the “modify” and “create” levels of each mission, students are able to navigate back to the “use” challenge for guidance if needed. These levels were iteratively refined and developed through a series of curriculum design activities with middle school teachers and students. Figure 1 depicts example screenshots of programming tasks for each phase: Use, Modify, and Create.

3 METHODS
3.1 Participants
Students at three different middle schools (one in Texas and two in North Carolina) played the ENGAGE game as part of their formal
within the game. Students in this study took identical randomized randomized trials, such as low experience, and students who chose the remain- remain- ing randomized trials, such as low experience, and students who chose the remain-

Thus, the 18 items we selected measured those concepts addressed addressed by the assessment [14]. The MG-CSCA measures the understanding understanding of four core CS concepts, namely variables, conditionals, loops, and algorithms, three of which compose the thematic levels of the game.

Eighteen items were selected from a validated instru- ment (MG-CSCA) developed by Rachmatullah et al. [32]. Both the ENGAGE programming challenges and the MG-CSCA were developed based on Grover and Basu’s FKSA framework, thus ensuring alignment between student learning experiences in the game and the assessment [14]. The MG-CSCA measures the understanding of four core CS concepts, namely variables, conditionals, loops, and algorithms, three of which compose the thematic levels of the game. Thus, the 18 items we selected measured those concepts addressed within the game. Students in this study took identical randomized items pre and post gameplay. Chronbach’s alpha value was $\alpha = .765$

Prior Experience. Data on students’ prior computer programming experience was collected by using a 5-point Likert scale question asking the frequency of exposure to programming experience prior to participating in this study (1 = never, 5 = every day). We followed Rachmatullah et al.’s methods to categorize students with low and high prior experience in which students who selected 1 and 2 were categorized as low experience, and students who chose the remaining three options (3, 4, and 5) were categorized as high experience students [32].

Semi-strucured interviews. A set of questions intended to explore students’ experiences, problem-solving strategies, affective states, and conceptual understanding during gameplay were developed to use in semi-structured focus group interviews conducted with a subset of students. These questions were extended during the interview processes based upon students’ responses. Some of the questions included: ‘When you didn’t know how to solve a coding problem in the game, describe how you figured it out?’, ‘Was there anything about the game that helped you solve the coding challenges?’, ‘Did you learn anything new from the game?’ Four focus groups were conducted and ranged in size from two to four students. The interview process took between 15 to 30 minutes.

3.2 Data Sources

MG-CSCA. Eighteen items were selected from a validated instrument (MG-CSCA) developed by Rachmatullah et al. [32]. Both the ENGAGE programming challenges and the MG-CSCA were developed based on Grover and Basu’s FKSA framework, thus ensuring alignment between student learning experiences in the game and the assessment [14]. The MG-CSCA measures the understanding of four core CS concepts, namely variables, conditionals, loops, and algorithms, three of which compose the thematic levels of the game. Thus, the 18 items we selected measured those concepts addressed within the game. Students in this study took identical randomized items pre and post gameplay. Chronbach’s alpha value was $\alpha = .765$.

Prior Experience. Data on students’ prior computer programming experience was collected by using a 5-point Likert scale question asking the frequency of exposure to programming experience prior to participating in this study (1 = never, 5 = every day). We followed Rachmatullah et al.’s methods to categorize students with low and high prior experience in which students who selected 1 and 2 were categorized as low experience, and students who chose the remaining three options (3, 4, and 5) were categorized as high experience students [32].

Semi-structured interviews. A set of questions intended to explore students’ experiences, problem-solving strategies, affective states, and conceptual understanding during gameplay were developed to use in semi-structured focus group interviews conducted with a subset of students. These questions were extended during the interview processes based upon students’ responses. Some of the questions included: ‘When you didn’t know how to solve a coding problem in the game, describe how you figured it out?’, ‘Was there anything about the game that helped you solve the coding challenges?’, ‘Did you learn anything new from the game?’ Four focus groups were conducted and ranged in size from two to four students. The interview process took between 15 to 30 minutes.

3.3 Data Analysis

3.3.1 Quantitative. Multilevel modeling (MLM) analysis was used to answer the first research question. MLM is useful to examine the changes or fluctuations in students’ CS concepts understanding from one time-point to another (intra-variability) and from one student to another student (inter-variability) [5]. This analysis is appropriate for unbalanced data, such that when some students took either pre- or post-test only [5]. Thus students with only one test score were not dropped and still included in the analysis, preserving statistical power. We tested the following equations:

Level 1 (Time):

\[
\text{CS Conceptual Understanding}_{it} = \beta_{0i} + \beta_{1i}(\text{Time}) + \epsilon_{it}
\]

Level 2 (Student):

\[
\beta_{0i} = \gamma_{00} + \gamma_{01}(\text{PriorExperience}) + u_{0i} \\
\beta_{1i}(\text{Time}) = \gamma_{10} + \gamma_{11}(\text{PriorExperience}) + u_{1i}
\]

The equation under Level 1 specifies the within-student relationship of CS concepts understanding and test occasion (Pre-post). The intercept $\beta_{0it}$ is the expected CS scores for i student when the test occasion is 0, which is in the pretest (posttest coded as 1), and the prior experience is 0, which is no to low prior computer programming experience. The first slope $\beta_{1it}$ is called CS learning, indicating the changes in CS concepts understanding from pretest to posttest. The $\epsilon_{it}$ is the residual errors representing variation around the mean of CS concepts understanding. The intercept and slopes in Level 1 become outcome variables in the student level (Level 2). We used $\gamma_{0i}$ and $\gamma_{1i}$ to answer the first research question as these intercepts represent CS learning and its prior computer programming experience, respectively. Before testing the above equations, a null model was run to examine whether we had enough within- ($\sigma^2_t$) and between-variability ($\tau_{00}$) in our data to proceed with MLM. This null model consisted of only CS concepts understanding without any predictors (slopes).

3.3.2 Qualitative. The interviews with students focused on eliciting their experiences, perceptions of, and problem-solving strategies during gameplay. All interviews were transcribed for analysis. The interviews were analyzed using constant comparative methods and went through three qualitative coding phases [40]. First, two researchers completed multiple rounds of open coding, mostly line-by-line of the transcripts, to capture each data segment’s essence. In the second phase, through the axial coding process, the codes gathered from the open coding process were grouped into more abstract categories and subcategories, resulting in a codebook. Any disagreements between the two researchers were discussed until consensus was reached, which led to the modification of the
codebook. A third researcher was then trained on and utilized the codebook to estimate the reliability of coding. The initial intercoder reliability was $k = .763$ indicating a satisfactory reliability [29]. The last phase was selective coding, in which a central phenomenon was deduced, and the interrelationship of categories and subcategories were used to generate a constructive story focusing on answering our research questions [36]. Interrelationships were discussed through peer debriefing [31]. Memos of methodological decisions, analytical ideas, and emerging themes were recorded throughout the analysis [40].

4 RESULTS

4.1 Quantitative

The unconditional model was run first to gather evidence for sufficient within- and between-students’ variances in CS conceptual understanding. The results showed a significant within- and between-students’ variances ($p < .001$), indicating that we could proceed with MLM for further analysis. We found that 27% of variability in CS conceptual understanding was within-student ($\sigma^2 = 46.19$, $p < .001$), and 73% was between-students ($\sigma^2 = 127.90$, $p < .001$).

Table 2 shows the MLM results. The results indicated a significant positive change in students’ CS conceptual understanding from pretest to posttest ($t = 2.06$, $p = .045$). This meant that students’ pretest scores ($M = 48.57$) were significantly lower than their posttest scores ($M = 51.78$). In addition, we did not find that students’ conceptual understanding was significantly associated with their prior computer programming experience ($t = -1.71$, $p = .090$), and students’ CS learning did not depend on their prior experience ($t = -0.87$, $p = .386$). Therefore, our results indicated that all students learned CS concepts by playing the ENGAGE game, regardless of their prior experiences. This model accounted for 5% and 4% of, respectively, within- and between-students’ variability in CS concept understanding.

4.2 Qualitative

Interviews with students helped to explain the role of UMC in providing support for student problem solving and learning of CS concepts within the game. We organized our findings into four major themes. First, student quotes revealed the importance of the programming code provided in the Use phase of each game level. Student comments like the ones below indicate that it facilitated a fundamental understanding of the CS and programming concepts needed for successful game completion:

“I would have to say that first initial code that really gave me a feel for how to code and how to put all the parts together.” (Student J)

“When you start the level it shows you how it's made so I went back to the past [challenges] I saw and combine it with what I know now and try to figure it out myself.” (Student C)

“When you start the level it shows you how it's made so I went back there, I looked at it, and I fixed what happened.” (Student E)

Third, student comments indicated that the Modify phase also provided another important opportunity for students to inspect and understand how the code worked:

“And also the bugs they showed me what to do like, A this is wrong, and it made me look at the code before actually pressing Run.” (Student E)

The UMC progression coupled with the narrative elements of the game’s storyline seemed to provide an optimal amount of scaffolding and challenge that encouraged the students’ persistence throughout the game.

“I like what I learned, because I didn’t know nothing about coding and I thought I was going to struggle, but the directions in there made it way easier...I just wanted to continue to see what’s going to happen next, like was [the antagonist] going to pop out of somewhere and try to rush me.” (Student C)

“The next level would be something different, and I’d be excited to figure out like, ‘Oh, what’s that?’” (Student B)

“What kept me going was knowing that I could beat the game if I just kept trying.” (Student F)

“It was interesting, so I just wanted to see what happened at the end of the game.” (Student D)

Ultimately, the qualitative phase of our study helped to corroborate that the game’s narrative design integrated with a UMC scaffolding progression helped to level the playing field and promote CS learning amongst all of the students, regardless of their prior CS and programming knowledge.

5 DISCUSSION

This study demonstrated that the students in our sample who played a CT/CS GBLE, ENGAGE, scaffolded with a UMC progression were able to learn important CS and programming concepts. In addition those students who had little or no previous programming were not disadvantaged in their learning experience. Qualitative interviews with the participants suggest that the Use phase of the UMC framework established an important starting point for novice students to understand and interpret the functionality of the block-based programming code and key CS concepts. These findings align with the work of Lister and colleagues, who articulated a hierarchy of programming skills where novice programmers must first understand basic code before they can move on to problem solving and writing code [27]. Thus, the UMC framework created a scaffolding bridge that equipped students with the confidence and knowledge to approach more complex programming challenges as they progressed through the game. Furthermore, we believe that UMC facilitated the development of schema used to devise problem-solving strategies for accomplishing the programming tasks. This is particularly important for middle grades students who often
need support choosing and executing appropriate problem-solving strategies [3]. Results also indicated that the storyline, enacted as a design element of challenge, motivated students to persist [21], encouraging game completion for both novice and experienced students.

Based on quantitative and qualitative data we found UMC to be a robust scaffolding framework that helped to achieve a desired balance of challenge and support for students who participated in the study. Thus, this study exemplifies how UMC can be extended beyond programming and computational modeling environments (e.g., [11, 25, 28]) to optimize the user experience of a CS GBLE, adding to the current corpus of both UMC and CS GBLE research literature. Furthermore, our work enabled us to employ and evaluate some of the design guidelines proposed by Johnson et al.[21] for novel CT/CS GBLEs. This included designing game challenges incorporating specific CS learning goals aligned to the FKSA framework so that we had a clear pathway for our design and assessment of CS Conceptual Understanding (Table 2: Unstandardized Coefficients (and Standard Errors) of CS Conceptual Understanding, Notes: Reference group/Intercept = Pretest and Low Experience; no asterisk $p > .05$, * $p < .05$, ** $p < .01$, *** $p < .001$).

<table>
<thead>
<tr>
<th>Effects</th>
<th>Parameter</th>
<th>Unconditional</th>
<th>Model 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Conceptual Understanding, $\beta_0$</td>
<td>$\gamma_0$</td>
<td>47.82 ***(1.49)</td>
<td>48.57***(1.74)</td>
</tr>
<tr>
<td>Prior Experience</td>
<td>$\gamma_1$</td>
<td>-6.26 (3.65)</td>
<td></td>
</tr>
<tr>
<td>CS Learning slope, $\beta_1$</td>
<td>$\gamma_2$</td>
<td>3.22* (1.56)</td>
<td></td>
</tr>
<tr>
<td>Time (Post-test)</td>
<td>$\gamma_3$</td>
<td>-3.48 (3.97)</td>
<td></td>
</tr>
<tr>
<td>Prior Experience</td>
<td>$\gamma_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Random Effects
- Between-student ($\tau_0$) 127.90***(27.26) 122.25***(26.24)
- Within-person fluctuation ($\sigma^2$) 46.19***(10.08) 43.66***(9.71)

Thus, this study exemplifies how UMC can be extended beyond programming and computational modeling environments (e.g., [11, 25, 28]) to optimize the user experience of a CS GBLE, adding to the current corpus of both UMC and CS GBLE research literature. Furthermore, our work enabled us to employ and evaluate some of the design guidelines proposed by Johnson et al.[21] for novel CT/CS GBLEs. This included designing game challenges incorporating specific CS learning goals aligned to the FKSA framework so that we had a clear pathway for our design and assessment of CS Conceptual Understanding (Table 2: Unstandardized Coefficients (and Standard Errors) of CS Conceptual Understanding, Notes: Reference group/Intercept = Pretest and Low Experience; no asterisk $p > .05$, * $p < .05$, ** $p < .01$, *** $p < .001$).

Table 2: Unstandardized Coefficients (and Standard Errors) of CS Conceptual Understanding (Notes: Reference group/Intercept = Pretest and Low Experience; no asterisk $p > .05$, * $p < .05$, ** $p < .01$, *** $p < .001$)

<table>
<thead>
<tr>
<th>Effects</th>
<th>Parameter</th>
<th>Unconditional</th>
<th>Model 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Conceptual Understanding, $\beta_0$</td>
<td>$\gamma_0$</td>
<td>47.82 ***(1.49)</td>
<td>48.57***(1.74)</td>
</tr>
<tr>
<td>Prior Experience</td>
<td>$\gamma_1$</td>
<td>-6.26 (3.65)</td>
<td></td>
</tr>
<tr>
<td>CS Learning slope, $\beta_1$</td>
<td>$\gamma_2$</td>
<td>3.22* (1.56)</td>
<td></td>
</tr>
<tr>
<td>Time (Post-test)</td>
<td>$\gamma_3$</td>
<td>-3.48 (3.97)</td>
<td></td>
</tr>
<tr>
<td>Prior Experience</td>
<td>$\gamma_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Random Effects
- Between-student ($\tau_0$) 127.90***(27.26) 122.25***(26.24)
- Within-person fluctuation ($\sigma^2$) 46.19***(10.08) 43.66***(9.71)

6 LIMITATIONS AND FUTURE WORK

There are limitations in this study that should be considered as readers interpret our findings. Additionally, they provide opportunities for future research. First, the results were only based on a single group of students who all experienced the UMC framework within the game design, without a control group for comparison. Thus, this design may threaten the intervention’s internal validity and isolate the variable being investigated, i.e., UMC framework. As such, future studies with a control group where students learn CS concepts within a GBLE that does not include a UMC framework could help ascertain additional evidence for the efficacy of the approach. Secondly, although the current sample size was acceptable to run a multilevel model analysis, the number of students categorized as having high prior programming experience was small, which might have impacted the statistical power. Finally, the qualitative data was collected from a smaller subset of our participants who volunteered to be interviewed by members of the research team. Thus, there is the potential of sample and self-report bias. Hence, future studies should invite more diverse students to the interview process to gain a fuller understanding of their experience learning CS concepts in a GBLE.

7 CONCLUSION

As CS becomes a part of formal K-12 learning experiences, it will become imperative to find engaging resources that effectively support the success of all learners. Additionally, the need for effective virtual learning has become exceedingly crucial, and GBLEs have the potential to be engaging platforms for CS learning. While GBLEs can be engaging for students to learn CT and CS skills and practices, an appropriate amount of scaffolding is needed to support students’ learning and persistence in a self-paced environment [21]. We found the integration of the UMC framework into our GBLE for CS and CT skills to be effective for enabling the students in our study, the majority of whom had little to no prior coding experience, to persist and learn the concepts needed to successfully complete the game. Thus, utilizing the UMC framework as a curricular guide for game-based learning has merit for impacting middle school CS and CT learning.

8 ACKNOWLEDGMENTS

This research was supported by the National Science Foundation through grant DRL-1640141. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES
