The Zones of Proximal Flow Tutorial

Designing Computational Thinking Cliffhangers

Ashok Basawapatna
State University of New York, College
at Old Westbury
Old Westbury, NY
basawapatnaa@oldwestbury.edu

Alexander Repenning
University of Applied Sciences and
Arts Northwestern Switzerland
Windisch, Switzerland
alexander.repenning@fhnw.ch

Mark Savignano
Minnesota State University, Mankato
Mankato, Minnesota
mark.savignano@mnsu.edu

ABSTRACT

The creation of computer science tutorials is becoming critically important as hundreds of millions of students each year get their first CS experience through self-directed online activities. Creating a "cliffhanger" activity, with high engagement during and motivation to continue learning post activity, is a balancing act. If tutorials provide too much detailed information, users may be able to follow instructions but can feel overwhelmed or bored. On the other hand, tutorials that do not sufficiently explain crucial steps risk frustrating users who might drop out of the activity. Zones of Proximal Flow (ZPF) tutorials are simple to create and provide a navigation structure of differentiated instruction allowing users to choose appropriate detail based on their self-assessed state of flow, from bored to anxious. Using Retention of Flow analysis, two Hour of Code game design tutorials were analyzed: a sophisticated online tutorial for the creation of Frogger, and a simple ZPF Pac-Man tutorial for the creation of Pac-Man. One hope was that the simple ZPF Pac-Man tutorial would not do much worse than the sophisticated Frogger tutorial, but surprisingly the ZPF Pac-Man tutorial significantly outperformed the Frogger tutorial in terms of student retention. The Pac-Man tutorial also displayed a high student motivation to continue programming past the end of differentiated instruction.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**;
Computer-assisted instruction;

KEYWORDS

Zones of Proximal Flow, Flow, Zone of Proximal Development, Computational Thinking, Hour of Code, Tutorial, Cyberlearning, Retention of Flow, Retention, Exposure

ACM Reference Format:

Ashok Basawapatna, Alexander Repenning, and Mark Savignano. 2019. The Zones of Proximal Flow Tutorial: Designing Computational Thinking Cliffhangers. In *Proceedings of the 50th ACM Technical Symposium on*

Computer Science Education (SIGCSE '19), February 27–March 2, 2019, Minneapolis, MN, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3287324.3287361>

1 INTRODUCTION

The push to expose students to programming and computational thinking has led to many encouraging developments. One popular example is Computer Science Education Week (CS Ed Week) which includes the Hour of Code that challenges novice learners across the world to spend an hour programming [32]. Students, many with little to no prior computer science experience, choose from a group of scaffolded online activities offering a gentle-slope introduction to coding [19]. Over half a billion students have participated in the Hour of Code from 2013 to 2017 [9].

These massive participation numbers are only positive if Hour of Code activities keep students' interest throughout and motivate them to continue learning after activity completion. If a novice student has an apathetic or negative Hour of Code experience, that student might decide programming is hard and/or boring [12, 15]. If this experience is repeated with half a billion participants, the Hour of Code actually risks losing a generation of computer scientists. Thus, activity designers and evaluators need to give special care to create "cliffhangers" where students are engaged throughout and are motivated to continue programming after reaching the end of activity instruction [26].

To help facilitate a cliffhanger experience, activity designers must optimize how much information to provide participants at each activity instruction to maximize engagement. On one hand, giving participants too much information could overwhelm users, and previous studies into minimal manuals have shown users performing faster and better with less instruction [7, 21]. However, giving participants too little information can lead to incomplete tutorial steps [31] putting users in a state of anxiety or frustration as they request more information [20]. Moreover, the online nature of Hour of Code activities means that anybody with an internet connection is a potential user, including international participation and varying age ranges. Therefore, few assumptions can be made about user skill level. How might a single activity go about creating a cliffhanger experience for the most amount of these diverse participants?

Differentiated instruction provides one possible solution. Differentiation is a form of user needs controlled scaffolding accommodating participants of differing skill level in an effort to engage all learners [16]. Differentiation has been successfully used to teach programming through formative progress monitoring and individualized instruction [14, 22]. How might one extend differentiated

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.

SIGCSE '19, February 27–March 2, 2019, Minneapolis, MN, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-5890-3/19/02...\$15.00

<https://doi.org/10.1145/3287324.3287361>

instruction to an Hour of Code context, optimally balancing on-line tutorial information for a variety of users? For the 2017 Hour of code, a tutorial was introduced that walked students through the creation of the classic arcade game Pac-Man. This tutorial was simple to create but featured differentiated instruction at each tutorial step based on a framework called the Zones of Proximal Flow [5]. The aim of this was to answer two specific research questions related to whether differentiated instruction could help create a cliffhanger activity.

- **RQ1 (Efficacy):** Can a simple differentiated scaffolding tutorial match the efficacy of an optimized professional single path tutorial?
- **RQ2 (Perserverance):** What effects does removing the differentiated scaffolding approach have on participant retention and willingness to continue?

The remainder of this paper will describe the development of this Pac-Man tutorial, including related work, and review the results of the tutorial in relation to the above research questions.

2 RELATED WORK

One strategy for motivating students in programming can be derived from previous theories on learner engagement and targeted instruction including the affective and cognitive experiences of the learner. Flow, as defined by Csikszentmihalyi, is a mental state of focused engagement that a person experiences when the current challenge matches the current skill [10, 11]. If the challenge exceeds skill, the person is put in a state of anxiety, and if the skill exceeds the challenge, the person is put into a state of boredom [10]. The Zone of Proximal Development, defined by Vygotsky, is the domain of challenges a learner can do given appropriate instruction [30]. The Zones of Proximal Flow (ZPF) is a conceptual framework that unifies Csikszentmihalyi's Flow [10] with Vygotsky's Zones of Proximal Development (ZPD) [30]. The Zones of Proximal Flow framework argues that when a student is given a challenge that exceeds their skill, that students can be provided just in time scaffolding before reaching a state of anxiety, targeted to their Zone of Proximal Development, guiding the learner back into a state of Flow [5].

Figure 1 depicts the Zones of Proximal Flow, in the context of the Scalable Game Design project, describing a scaffolding strategy employing a project-first approach to integrate game and simulation development units into middle school classrooms [5]. The y-axis of Figure 1 depicts the game and simulation design challenges present in the Scalable Game Design curricula [29]. The x-axis of Figure 1 depicts the computational thinking skills, entitled Computational Thinking Patterns (a list of which can be found in [1]), necessary to complete a given challenge [18]. At Point A in Figure 1, a student has recently completed creating the game Frogger and is now given another more sophisticated project to complete such as creating the game Pac-Man or creating a Bridge Building simulation (Point B in Figure 1). Since the student has not yet learned the necessary computational thinking skills to create this more advanced project, the challenge far exceeds the student skill level and the student starts to drift out of Flow into a state of anxiety. In the Zones of Proximal Flow model, before the student gets into an anxious state, the student is provided with scaffolding in the form of just-in-time

Figure 1: The Zones of Proximal Flow with Computational Thinking Skills on the x-axis and Challenges on the y-axis [5]

principles aligned to the student's Zone of Proximal Development, guiding the student back into a state of Flow and allowing them to continue programming their project. These just-in-time principles can include embedded tutorial instruction [13], intelligent critiquing systems [2] and instructor intervention [4] among others.

How might one begin to evaluate the efficacy of Hour of Code activities to indicate how many participants are in Flow? Previous Hour of Code research has looked at student retention throughout a given activity to provide just-in-time information [23] and student engagement through qualitative measures [17]. The Retention of Flow is a method of evaluating these cyberlearning activities based on student retention [27, 33]. To better understand this notion, imagine 10,000 people, each one assembling an Ikea table on their own. At each step of the assembly process, some people in the population drop out of the activity, while others are retained and continue assembling their table. Points at which many more people than usual drop out might indicate a challenge step that far exceeds user skill given the current scaffolding (i.e. the Ikea instructions at that step). Similarly, the Retention of Flow uses participant retention, measured by a lines of code (also called program length) metric, which follows a negative exponential curve starting at 100%, at the beginning of the activity (0 lines of code), and declining as people drop out of the activity. The program length can then be matched to corresponding activity milestones in the retention plot to determine at which point of the activity students dropped out. Some assumptions behind the Retention of Flow are that, void of any external factors, the people retained at each step are the ones wherein the current challenge matches their skill, and it is the aim of any activity designer to create activity steps that have a high continuation probability yielding a higher retention.

Retention relates to Research Question 1 in that, assuming that an activity has a high educational value, a higher retention throughout the activity is ideal for the motivational purposes of the Hour of Code. Therefore, one can in-part use retention to compare two similar activities in terms of efficacy. Relating to Research Question 2, participant retention after activity instruction ends indicates a

cliffhanger. To put this another way, devoid of the instructional scaffolding the student continues, implying a level of intrinsic motivation [6]. Instilling this intrinsic motivation in students should be the ultimate goal of any Hour of Code activity, and Retention of Flow begins to provide initial evidence of whether this has occurred.

3 ZONES OF PROXIMAL FLOW TUTORIALS

The Zones of Proximal Flow Tutorial attempts to extend the ZPF Framework into a cyberlearning activity [8] providing differentiated instruction. The idea of the ZPF tutorial is as follows; the participant works through the tutorial steps, and with each instruction encountered, self assesses their current state choosing from three possibilities:

- (1) **OK (Flow)**: The user understands the instruction and does not need any more guidance.
- (2) **How (ZPD)**: The user needs to learn a new concept before implementing the tutorial step.
- (3) **Show (Anxiety)**: The user still cannot, or does not have the time to, follow the instructions. A link to an example project is provided with the instruction implemented allowing the user to mimic or clone the implementation.

The 2017 Hour of Code featured an AgentCubes [28] game-design activity that guided students through the creation of the classic arcade game Pac-Man. The Pac-Man activity used the Zones of Proximal Flow tutorial strategy of differentiated instruction by giving the participant three choices at every activity step. Figure 2 depicts part of the Pac-Man Zones of Proximal Flow Tutorial that asks participants to create the Pac-Man characters. In Figure 2, the user initially encounters a slide describing **What** to do. In this case the user is asked to create the characters, also called agents, necessary for a simple Pac-Man game. These include Pac-Man, ghosts, a wall, and a background floor. If the user is able to follow that instruction, they can click the **"OK"** button and are taken to the next activity instruction. However, if the user does not understand that instruction, the user can click the **"How"** button, which will take the user to another slide with more detailed instructions. In the case depicted in Figure 2, the detailed instruction is in the form of a video that goes through adding an agent and drawing that agent. If this explanation makes sense, the user can implement the instruction and then click **"OK"** to be taken to the next activity instruction ("Program Pacman" in Figure 2). However, if the user still is having trouble understanding the instruction, the user can click **"Show"** to be taken to an example or "prebaked" project, for instance like a prebaked cake on a cooking show, with just the steps up to this point of the activity implemented. Once the project is opened, the user can play, explore, and mimic and/or clone the implementation allowing them to continue working.

The Pac-Man activity is 14 slides long and consists of two parts. Part one consists of a ZPF tutorial from slide one to slide ten, with the first slide describing how to use the tutorial. The first ten slides, corresponding to program lengths of 0 to 17, are dedicated to creating a simple Pac-Man game, including a keyboard controlled Pac-Man and ghosts that randomly move. ZPF instruction abruptly stops at slide ten. Part two, consisting of slides eleven to fourteen and corresponding to program lengths of 18 to 40, encourage participants to do extension activities adding extra functionality to

their Pac-Man game including Pac-Man being consumed by ghosts, Pac-Man reorienting in the direction of movement, intelligent ghost movement based on collaborative diffusion [25], and pellets that Pac-Man can consume to win. However, from slides eleven to fourteen, no differentiated instruction is provided as to how to achieve this; the only support provided to students is a link to a sample prebaked project for each of these steps.

3.1 Pac-Man vs. Frogger

The Zones of Proximal Flow Pac-Man Tutorial is simple to create as compared to the other AgentCubes based Hour of Code tutorial offered since 2015: Frogger [13]. the Frogger tutorial features a video integrated directly into AgentCubes online allowing users to create the game as they watch step by step instructions. Furthermore, the video narration is multilingual, using professional voice over personnel, and is split up into chapters allowing students to navigate to a specific tutorial instruction on demand. The creation of the Frogger tutorial took a team of dedicated programmers, educational designers, and playtesters working together over the course of many months. In this sense, Frogger is an optimized single path tutorial aimed at providing instruction with the right amount of information to retain the widest audience. Previous research shows that the Frogger tutorial had a similar participant retention to the official Hour of Code Angry Birds activity [26].

The Pac-Man activity, on the other hand, was created by two people over the course of a two days in the form of a slideshow presentation. However, despite the low production value, the Pac-Man activity features a ZPF tutorial providing users with the agency to choose instruction appropriate to their skill level. Given this stark difference in production effort, can the simple Pac-Man Hour of Code tutorial, supporting students through the Zones of Proximal Flow via differentiated instruction, be nearly as effective as the professionally created Frogger tutorial in terms of student retention and motivation to continue post instruction?

4 METHODOLOGY

This research analyzed a collected sample of 1,468 Pac-Man games and 767 Frogger games submitted during the Hour of Code week. These games were initially identified as Pac-Man and Frogger through a script and manually checked to ensure they were indeed Pac-Man and Frogger. Answering the research questions necessitates one, determining how engaged students are during the activity and two, evaluating the probability of students to continue programming post activity scaffolding. By analyzing the retention of Pac-Man and Frogger at corresponding program lengths, we can begin to compare the retentions between the two activities during instruction and beyond. To answer Research Question 1, activity efficacy, we look at how the activity retention of Pac-Man compares to Frogger from program length 0 to 17 (Part 1 of the Pac-Man activity featuring the ZPF tutorial). To answer Research Question 2, we look at both the retention plot from program length 18 to 40, where the ZPF instruction ends and students are encouraged to program extension activities, as well as program lengths greater than 40, to determine to what extent, if any, students continue past the extension activities. It should be noted for purposes of comparison to previous research, the program length metric used in this

Figure 2: The ZPF Flow Chart depicting the green OK button leading to the next instruction slide, the orange button leading to a How slide that further explains the current tutorial step, and the red Show button leading to a "prebaked" project.

study is similar, but not identical, to [3]. A manual survey of the sample games was done to ensure the program lengths actually corresponded to activity milestones. Also of note, the collected sample of games does not include cloned projects unless they had significant programming time associated with them as not to bias the data.

5 RESULTS

Figure 3 plots the percent of Participants Retained vs Lines of Code. Pac-Man retention is plotted in red and Frogger in blue. The three shaded regions refer to the parts of the Pac-Man tutorial, with green being the ZPF Tutorial (program lengths 1 to 17), yellow being the extension activities (program lengths 18 to 40) and peach being everything past the end of the Pac-Man activity (program lengths greater than 40). The following results are organized according to the Research Questions.

RQ1: Efficacy. The part of Figure 3 shaded in green corresponds to the ZPF portion of the Pac-Man activity. Both Pac-Man and Frogger retentions start at 100% at the beginning of their respective activities, with the Pac-Man having a higher participant retention than the Frogger activity as program lengths increase. The difference in retention reaches a maximum at Program length 15 wherein the Pac-Man retention is almost 14% higher than the Frogger retention. At program length 17 the ZPF portion of the Pac-Man tutorial ends. A two sample z-test for proportions run at program length 17 found significantly more students in the Pac-Man activity retained through 17 lines of code (75%) than the Frogger activity (62%) with $z=6.72$, $p=.05$.

RQ2: Persistence. The part of Figure 3 shaded in yellow, program length 18 to 40, refers to the Pac-Man extension activities described in Section 3. At this point of the activity the ZPF differentiated instruction part of the tutorial has completed. Figure 3 depicts a drop in retention of almost 10% between program length 17 and program length 18 with a continuation percentage of 87.9%. Around 57% of students who start the extension activities finish all of them (i.e. are retained to program length 40). A two sample z-test for proportions run at program length 40 showed significantly more students in the Pac-Man activity were retained through 40 Lines of code (38%) than the Frogger activity (27%) with $z=5.16$, $p=.05$. To put this another way, 38% of participants who start the Pac-Man Hour of Code activity end up programming past the ZPF instructional section and finish implementing all the extension activities.

An ANOVA analysis comparing the groups of Pac-Man and Frogger projects from Lines of Code 1 to 40 was completed using 105 random samples for each group (sample of size of 210) which it was estimated would achieve a power of 0.95 with a level of significance at 0.05. After running the analysis, the p-value was at .00017 which is lower than the level of significance of 0.05 suggesting that there is indeed a significant difference between the two groups throughout the duration of the Pac-Man activity spanning both the pre and post ZPF activity instruction.

At program length 40 to 41 there is a precipitous drop wherein the final slide of the Pac-Man activity is reached. About half the remaining students drop out at this point and half continue adding to their Pac-Man game. Frogger instruction continues to program length 58, and between program length 41 and 60, Frogger retains

Figure 3: Pac-Man and Frogger Activity Retention. Shaded regions refer to parts of the Pac-Man Activity.

more students than Pacman. Overall 19% of students who start the Pac-Man tutorial continue programming past the end of the Pac-Man extension activities.

6 DISCUSSION

The above results begin to show the effectiveness of the Zones of Proximal Flow Tutorial both during the activity instruction (program length 1 to 17) and after activity instruction (from program length 18 to 40). The ANOVA analysis indicates a significant difference between participants doing the Pac-Man tutorial and the Frogger tutorial. This section will delve into these differences by discussing the results corresponding to activity duration engagement (RQ1) and post-activity perseverance (RQ2). Finally the section will conclude with study shortcomings and implications.

6.1 Evidence of ZPF Tutorial Efficacy

The first research question attempts to answer how effective the ZPF tutorial is for participants. This spans program length 1 to 17 in the Pac-Man activity. Figure 3 depicts that during the activity, the Pac-Man Zones of Proximal Flow Tutorial consistently retained more participants than the Frogger tutorial. By the end of the ZPF tutorial portion of the Pac-Man activity, there is a significantly greater number of participants retained than in the Frogger activity. Furthermore, over three quarters of students who start the activity get through the ZPF portion of the tutorial. This begins to indicate the power of differentiated instruction and the Zones of Proximal Flow approach.

This result is surprising for a few reasons. The first is that major design effort was put into producing the Frogger activity, outlined in Section 3.1, and the Frogger activity itself is embedded in the programming environment. Given these advantages, one might expect that the Frogger activity would allow participants to more easily create their game. Previous research has posited the Retention of Flow conjecture which in part states that a tutorial step that is markedly more challenging than previous steps will exhibit a precipitous decline in retention [26]. The ZPF tutorial strategy attempts to avoid this by allowing users to choose the level of instruction that best support them based on their self-assessed current skill. In this manner, ZPF tutorials aim to keep students in the Flow by always providing them with the necessary on-demand scaffolding to ensure skill level in balance with the challenge. Given the significantly higher retention as compared to Frogger, the ZPF strategy seems to be successful.

6.2 Evidence of Perseverance Post ZPF Activity Instruction

At program length 18 the participant has created a basic Pac-Man game with a keyboard controlled Pac-Man and randomly moving ghosts. At this point, the ZPF tutorial strategy of differentiated instruction is abandoned and students are given extensions that they can add to their Pac-Man game. The only support for these extensions are a few example prebaked projects. There is a 10% drop in retention at this point, which could be due to this marked increase in challenge coupled with the end of creating a basic Pac-Man game.

However, well over one third of the students (38%) who start the Pac-Man activity end up programming past the instructional part of the tutorial and implementing all the Pac-Man extensions. This indicates that for these participants, Pac-Man is a cliffhanger as they program past activity instruction to add functionality to their game. At program length 40, the Frogger activity is still providing participants with step by step instructions, and at that point, the Pac-Man retention is still significantly higher than the Frogger retention. Therefore, student perseverance is such that the Pac-Man activity, devoid of any step by step instructions, still manages to retain a greater number of participants than the Frogger activity.

From Program length 40 to 41 there is another drop in Pac-Man retention (40% to 20% retention). This drop is to be expected as the Pac-Man extension activities end and students reach the end of the Pac-Man activity slides. This is the first point Pac-Man retention falls below Frogger retention, as the Frogger activity instruction does not end until program length 58. Still, half the participants who finished the extension activities continue to independently program past the end of the Pac-Man activity slides. Furthermore, one fifth of the total participants who start the Pac-Man activity end up programming past the end of the Pac-Man activity. At line 61, the Pac-Man activity retention once again overtakes the Frogger activity retention as people drop out once the Frogger activity is complete. At 64 lines of code, 20 lines past the final extension activity and at program lengths over three and a half times as long as the final ZPF instruction, 10% of participants are still retained in the Pac-Man activity.

6.3 Study Shortcomings and Implications

The results of this study begin to suggest that the ZPF tutorial approach with Pac-Man is an effective strategy. To come to this conclusion we compared the results of the ZPF tutorial to another Hour of Code AgentCubes tutorial: Frogger. However, it is hard to say whether the Frogger tutorial itself has a "normal" retention curve with so little Hour of Code retention data published. One interesting data point is that the Frogger tutorial has been improved over three years of the Hour of Code [2], and in past research showed similar retention to the official Hour of Code Angry Birds tutorial [26]. Though it would be nice to have more published Hour of Code retention data from a variety of activities, this provides a bit of outside validity to the comparison.

The ZPF tutorial approach might not be the only thing that engaged students during the activity. For example, one motivator could be that Pac-Man is a much more popular game than Frogger. In fact, more participants attempted the Pac-Man tutorial than Frogger indicating a greater initial interest. This motivation external to the activity could positively affect retention. However, one might not expect any external motivation to continue through the tutorial activity if the activity itself is not engaging.

The data shows that the ZPF tutorial had increased retention over corresponding places in the Frogger tutorial. A major difference between the ZPF tutorial and Frogger tutorial is the idea of differentiated instruction. However, one piece of instrumentation not included in the ZPF tutorial was the amount of users who clicked on each button. For example, in Figure 2, we do not know how many students clicked on the "OK", "How", and "Show" buttons.

Therefore, it is hard to pinpoint which steps students might have found challenging and the tutorial paths they chose when they encountered these challenging steps. Click data, including per-user click data, would help pinpoint how users navigate the tutorial and give insight to which tutorial steps users find challenging and options they find useful. Furthermore, a post-activity questionnaire could allow users to explain what they found helpful or unhelpful about the differentiated instruction approach, and a cognitive walkthrough could show how clearly students perceived each potential choice at each tutorial step [24].

One additional artifact that occurs between program length 1 and 17 is a small drop of around 6% between program length 15 and 16. At program length 15 the participant has just completed programming Pac-Man to move with keyboard input. It is not clear why there would be a drop here, but it is interesting to note that an almost identical drop of 6% occurs in the Frogger retention plot, between program lengths 11 and 12, where again the participant has just completed programming the Frog to move. It could be that these are the first programming interactions users implement, and thus, some participants decide the activity is not for them at this point. Further research must be done to figure out exactly why this is the case.

This research begins to show the power of differentiation in self-directed learning, and adding differentiation might be an easy way to boost activity retention. Further research must be done to explore this, possibly by retrofitting the Frogger Hour of Code tutorial with differentiation to determine the retention effects. Finally, an emerging benefit of this research is that it provides an effective DIY strategy for teachers short on time looking to create Hour of Code activities.

7 CONCLUSION

This paper presents a new approach to Hour of Code style tutorials based off a simple differentiated instruction strategy guided by the Zones of Proximal Flow framework, and evaluated by analyzing activity retention. This evaluation shows that the Zones of Proximal Flow Tutorial has a significantly higher retention than a more professionally made Frogger tutorial, with students programming through the end of ZPF instruction and activity completion. To ensure the Hour of Code is effective in motivating the maximum numbers of students necessitates activities that keeps students in Flow. Zones of Proximal Flow Tutorials provide an avenue to better match instructional challenge to a diverse range of participant skill, hopefully yielding a cliffhanger first exposure computer science experience.

ACKNOWLEDGMENTS

Data used with permission from AgentSheets Inc. Inflatable Icon is patented under US Patent No. 7,630,580. This work is supported by the Hasler Foundation, the Swiss National Science Foundation under Grant CRAGP2_158545, and the National Science Foundation under Grant Numbers 0833612, 1345523, and 0848962. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these organizations.

REFERENCES

- [1] Ashok Basawapatna, Kyu Han Koh, Alexander Repenning, David C. Webb, and Krista Sekeres Marshall. 2011. Recognizing Computational Thinking Patterns. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 245–250. <https://doi.org/10.1145/1953163.1953241>
- [2] Ashok Basawapatna and Alexander Repenning. 2017. Employing Retention of Flow to Improve Online Tutorials. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 63–68. <https://doi.org/10.1145/3017680.3017799>
- [3] Ashok Basawapatna, Alexander Repenning, Mark Savignano, Josiane Manera, Nora Escherle, and Lorenzo Repenning. 2018. Is Drawing Video Game Characters in an Hour of Code Activity a Waste of Time?. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 93–98. <https://doi.org/10.1145/3197091.3197136>
- [4] Ashok Ram Basawapatna, Alexander Repenning, and Kyu Han Koh. 2015. Closing The Cyberlearning Loop: Enabling Teachers To Formatively Assess Student Programming Projects. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 12–17. <https://doi.org/10.1145/2676723.2677269>
- [5] Ashok R. Basawapatna, Alexander Repenning, Kyu Han Koh, and Hilarie Nickerson. 2013. The Zones of Proximal Flow: Guiding Students Through a Space of Computational Thinking Skills and Challenges. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 67–74. <https://doi.org/10.1145/2493394.2493404>
- [6] Roland Benabou and Jean Tirole. 2003. Intrinsic and extrinsic motivation. *The review of economic studies* 70, 3 (2003), 489–520.
- [7] John B. Black, John M. Carroll, and Stuart M. McGuigan. 1986. What Kind of Minimal Instruction Manual is the Most Effective. *SIGCHI Bull.* 18, 4 (May 1986), 159–162. <https://doi.org/10.1145/1165387.275623>
- [8] Christine L Borgman. 2011. *Fostering learning in the networked world: The cyber-learning opportunity and challenge*. DIANE Publishing.
- [9] Code.org. 2018. Leaderboard for the Hour of Code. Retrieved August 28, 2018 from <https://code.org/leaderboards>
- [10] Mihaly Csikszentmihalyi. 1997. *Finding flow: The psychology of engagement with everyday life*. Basic Books.
- [11] Mihaly Csikszentmihalyi and Isabella Csikszentmihalyi. 1975. *Beyond boredom and anxiety*. Vol. 721. Jossey-Bass San Francisco.
- [12] Jie Du, Hayden Wimmer, and Roy Rada. 2016. "Hour of Code": Can It Change Students' Attitudes toward Programming? *Journal of Information Technology Education: Innovations in Practice* 15 (2016), 52–73.
- [13] Nora A Escherle, Silvia I Ramirez-Ramirez, Ashok R Basawapatna, Dorit Assaf, Alexander Repenning, Carmine Maiello, Yasko Ch Endo, and Juan A Nolasco-Flores. 2016. Piloting computer science education week in Mexico. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 431–436.
- [14] Nuno Gil Fonseca, Luis Macedo, and António José Mendes. 2018. Supporting Differentiated Instruction in Programming Courses Through Permanent Progress Monitoring. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 209–214. <https://doi.org/10.1145/3159450.3159578>
- [15] Google CS Ed Research group et al. 2014. *Women who choose computer science—what really matters: The critical role of encouragement and exposure*. Technical Report. Technical report, Google. Retrieved from <http://static.googleusercontent.com/media>.
- [16] Tracey Hall. 2002. Differentiated instruction. *Wakefield, MA: National Center on* (2002).
- [17] CUNO Klopfenstein, Saverio Delpriori, BRENDAN DOMINIC Paolini, and Alessandro Bogliolo. 2018. X Marks the Bot: Online coding-based treasure hunt games for code literacy. *INTED PROCEEDINGS* (2018), 4867–4873.
- [18] Kyu Han Koh, Hilarie Nickerson, Ashok Basawapatna, and Alexander Repenning. 2014. Early Validation of Computational Thinking Pattern Analysis. In *Proceedings of the 2014 Conference on Innovation & #38; Technology in Computer Science Education (ITiCSE '14)*. ACM, New York, NY, USA, 213–218. <https://doi.org/10.1145/2591708.2591724>
- [19] Deepak Kumar. 2014. Digital Playgrounds for Early Computing Education. *ACM Inroads* 5, 1 (March 2014), 20–21. <https://doi.org/10.1145/2568195.2568200>
- [20] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1779–1788. <https://doi.org/10.1145/2470654.2466235>
- [21] Adrianus W. Lazonder and Hans van der Meij. 1993. The minimal manual: is less really more? *International journal of man-machine studies* 39, 5 (1993), 729–752. <https://doi.org/10.1006/imms.1993.1081>
- [22] Vinhthuy Phan and Eric Hicks. 2018. Code4Brownies: An Active Learning Solution for Teaching Programming and Problem Solving in the Classroom. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 153–158. <https://doi.org/10.1145/3197091.3197128>
- [23] Chris Piech, Mehran Sahami, Jonathan Huang, and Leonidas Guibas. 2015. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Proceedings of the Second (2015) ACM Conference on Learning Scale (LS '15)*. ACM, New York, NY, USA, 195–204. <https://doi.org/10.1145/2724660.2724668>
- [24] Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. 1992. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of man-machine studies* 36, 5 (1992), 741–773.
- [25] Alexander Repenning. 2006. Excuse me, I need better AI!: employing collaborative diffusion to make game AI child's play. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*. ACM, 169–178.
- [26] Alexander Repenning and Ashok Basawapatna. 2016. Drops and Kinks: Modeling the Retention of Flow for Hour of Code Style Tutorials. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCe '16)*. ACM, New York, NY, USA, 76–79. <https://doi.org/10.1145/2978249.2978260>
- [27] Alexander Repenning, Ashok Basawapatna, Dorit Assaf, Carmine Maiello, and Nora Escherle. 2016. Retention of Flow: Evaluating a Computer Science Education Week Activity. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 633–638. <https://doi.org/10.1145/2839509.2844597>
- [28] Alexander Repenning, David C Webb, Catharine Brand, Fred Gluck, Ryan Grover, Susan Miller, Hilarie Nickerson, and Muyang Song. 2014. Beyond minecraft: Facilitating computational thinking through modeling and programming in 3d. *IEEE Computer Graphics and Applications* 34, 3 (2014), 68–71.
- [29] Alexander Repenning, David C. Webb, Kyu Han Koh, Hilarie Nickerson, Susan B. Miller, Catharine Brand, Ian Her Many Horses, Ashok Basawapatna, Fred Gluck, Ryan Grover, Kris Gutierrez, and Nadia Repenning. 2015. Scalable Game Design: A Strategy to Bring Systemic Computer Science Education to Schools Through Game Design and Simulation Creation. *Trans. Comput. Educ.* 15, 2, Article 11 (April 2015), 31 pages. <https://doi.org/10.1145/2700517>
- [30] Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes*. Harvard university press.
- [31] Ron Wakkary, Markus Lorenz Schilling, Matthew A. Dalton, Sabrina Hauser, Audrey Desjardins, Xiao Zhang, and Henry W.J. Lin. 2015. Tutorial Authorship and Hybrid Designers: The Joy (and Frustration) of DIY Tutorials. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 609–618. <https://doi.org/10.1145/2702123.2702550>
- [32] Cameron Wilson. 2015. Hour of Code—a Record Year for Computer Science. *ACM Inroads* 6, 1 (Feb. 2015), 22–22. <https://doi.org/10.1145/2723168>
- [33] Yao Xiong, Hongli Li, Mindy L Kornhaber, Hoi K Suen, Barton Pursell, and Deborah D Goins. 2015. Examining the relations among student motivation, engagement, and retention in a MOOC: A structural equation modeling approach. *Global Education Review* 2, 3 (2015).