COLUMBIA UNIVERSITY

Columbia University, Spring 2021

Parson's Programming Puzzles: Optimizing Efficiency and Investigating the Effects of Feedback

Further research on Social Addictive Gameful Engineering (SAGE) design and computational thinking (CT)

Alexander Liebeskind | al3853

Advisors: Jeff Bender, Gail E. Kaiser

RESEARCH PROPOSAL

1 Abstract/Overview

The purpose of this research is to extend upon prior studies into Social Addictive Gameful Engineering (SAGE) (Bender (2018)) and computational thinking (CT) by performing additional field studies and analysis. Field study 1 (fs1) was recently completed by Jeffrey Bender, Bingpu Zhao, Lalitha Madduri, and Gail Kaiser, and submitted for publication at the 26th annual conference on Innovation and Technology in Computer Science Education (ITiCSE) under the title *Integrating Parsons Programming Puzzles with Scratch*. Field study 2 (fs2) is underway, and data collection has already been completed. Preprocessing, data analysis, and write-up have not yet begun. The proximate goal is therefore to complete a second paper on Gameful Direct Instruction (GDI) targeted at the 14th Annual International Computing Education Research Conference or the SIGCSE Technical Symposium. These two conferences have paper submission deadlines in March and August respectively. It should be noted that fs2 differs significantly from fs1 in both the sample size and condition groups. Fs1 involved 75 participants spread across 3 condition groups, while the sample in fs2 is comprised of 662 participants (estimated based on trends in MTurk and Prolific) across 9 total groups (including a control group). The results of fs2 will not only corroborate or contradict those of fs1, but ideally present independently meaningful findings.

Stretch goals and secondary objectives for the semester include work on field study 3 (fs3), such as protocol content, game creation, and study instructions, as well as potential development of other Parson's environments. These topics will be pursued if fs2 is completed and there is remaining time for additional research.

2 Related Work

2.1 Integrating Parsons Programming Puzzles with Scratch

The fs1 paper (Jeff Bender (2020)) is perhaps the most natural place to begin, given the similarities between fs1 and fs2. In fs1, participants were randomly assigned to

either Parsons Programming Puzzles (PPP) training, PPP with distractors training, or training by solving puzzles with access to all blocks and without move correctness or score feedback. The study found that overall cognitive load did not differ between groups, though self-reported extraneous cognitive load in the PPP group without distractors was lower than that of the PPP group with distractors, which was in turn lower than that of the limited-constraint-feedback condition. In certain training test settings, both PPP groups outperformed the limited-constraint-feedback condition feedback group, though these results were not replicated in the transfer task. Those trained in the PPP and PPP-distractor conditions also required less time to complete the puzzles, and therefore performed more efficiently. Finally, PPP group participants demonstrated higher motivational scores regarding attitude surveys than their limited-constraint-feedback counterparts. Overall, these findings support the hypothesis that PPP and PPP with distractors training increases motivation and reduces extraneous cognitive load (as compared to limited constraint and feedback), and refute the hypothesis that PPP training yields the highest learning efficiency.

2.2 Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses

The original Parson's Programming Puzzles paper (Dale Parsons (2006)) was designed around several key aims: maximize the engagement, constrain the logic, permit common errors, model good code, and provide immediate feedback. The emerging puzzle structure used activity diagrams and distractors, each aimed to influence the learning environment.

These components conjunctively taught programming principles such as order of statements, variables, and nested for loops. The initial proposal stated that Parson's Programming Puzzles were best implemented in a web-based format, in order to facilitate dispersion in a variety of settings. Many of these principles continue to guide modern Parson's Programming Puzzles. Additionally, the paper espoused future updates to improve the user interface, data collection, and other facets of the environment. An example Parson's activity diagram can be found in the Appendix.

2.3 Lessons Learned from Available Parsons Puzzles Software

Since the inception of the original Parson's Programming Puzzles, a number of applications have been built to best develop upon the principle. These platforms include the Hot Potatoes Implementation, JS-Parsons, Epplets, and EvoParsons (proposed by the study in question, (Dmytro Vitel (2019)). These implementations vary in user experience, required software, deployment and access, and interface, among other factors. It is essential to consider the advantages and disadvantages of prior methodologies (uncovered by past research) when attempting to build a Parson's Programming Puzzle system.

2.4 Measuring Cognitive Load in Introductory CS: Adaptation of an Instrument

Cognitive Load Theory (CLT) holds that learning is impaired when the total amount of processing requirements exceed the limited capacity of working memory (Briana B. Morrison (2014)), which here dictates that the Parson's Programming Puzzle must not exceed the capacity of working memory. More specifically, in the context of Social Addictive Gameful Engineering (SAGE), cognitive load theory and the metrics defined for its objective measurement prefigure environment design, since it is imperative that cognitive load be balanced with rigor in Parson's Programming Puzzle construction. This study specifically applied a set of instructions to determine Intrinsic Load, Extraneous Load, and Germane Load, to broadly define an instrument for the objective measurement of cognitive load. The study also investigated the role of each facet in learning efficiency.

3 Scheduled Objectives

3.1 Background Research

3.1.1 fs1 GDI Paper

The first step to background research is fundamentally understanding the methods and results of fs1. Though I have read through fs1 and viewed the results directly, I will begin by reading the papers referenced by the fs1 paper submission in full to inform methodology on fs2 analysis and write-up.

3.1.2 Relevant Statistics and Graphical Methods

In order to understand the statistics and figure types that will be used for fs2, it is important to review results included in similar studies. Related works have already been consolidated for review.

3.1.3 Data Extraction

Analytics processes through Qualtrics and SAGE will account for statistic and figure generation. The Qualtrics results have been exported, stored, and validated for MTurk/Prolific approvals, and are uploaded to the 2020/Results/fs2/Surveys folder. The remaining work is to categorize participants by condition (with feedback: PPP (A), PPPd (B), ScratchIE (C), ScratchIEd (D), without feedback: PPP (A2), PPPd (B2), ScratchIE (C2), ScratchIEd (D2), and control condition: PPP on sequences (AC)). For the SAGE database (Atlas), fs1 and fs2 are accessed through MongoDB, using a Dev DB Connection. The remainder of the results will come from this database source, as described in the analysis section.

3.2 Data Timing and Correctness Preprocessing

Pre and post test data must be found across conditions to determine if there are any significant differences between groups. This will help guide posttest results to the experimental condition, as compared to incoming knowledge.

3.3 Analysis

Analysis mechanisms can be coded to extract pertinent data from SAGE such as time per puzzle per student, correctness of final submission, number of moves, explanation upon submission, and move visualizations. These data will be analyzed for significant inter-condition differences. SAGE data can also be used to compute usage data, cognitive load, efficiency, performance, and motivation. Analysis on Qualtrics and SAGE data will deliver quantitative findings, as well as tables and figures, for the final paper.

3.4 Writing, Editing, and Submission

Once data preprocessing and analysis are complete, the work can be written as a paper for submission using the guidelines on length and sections of the selected conference (which currently remains to be determined). This step will also include iteratively editing to improve the piece.

3.5 Stretch Goals

There is a large amount of work remaining for fs3. This includes creating protocol content for looping CT concept, games across conditions, game instructions, and a Qualtrics study guide, as well as further deliverables. Additional objectives can also be found on the Spring 2021 Field Study Goals document.

4 Closing Notes

With regard to logistics, Jeff Bender and I will be meeting once a week (currently scheduled on Thursdays, pending approval by other group members), to discuss research progress and collaborate on common objectives. We will also consult with Dr. Kaiser based on availability whenever possible.

The proposal steps are intended to guide research this semester, but are not exhaustive. More resources can be found on the 2020 SAGE GDI Study page (https://drive.google.com/drive/folders/1ZbGE3CRhp_QxJmLabZv5mZg8TCjjt8Zg)

(Jeff Bender et al. (2020)), or at the shared Github page (Github CU-SAGE (https://github.com/cu-sage/Documents/).

The Spring 2021 Field Study Goals was used as a general guide for semester objectives, and can be found on Google Drive (Jeff Bender (2021)).

References

Bender, J. (2018). Social addictive gameful engineering (sage): An intelligent game-based learning and assessment system that infuses computational thinking in grade 6-8 curricula.

Briana B. Morrison, Brian Dorn, M. G. (2014). Measuring cognitive load in introductory cs: Adaptation of an instrument. *ICER '14: Proceedings of the tenth annual conference on International computing education research.*

Dale Parsons, P. H. (2006). Parson's programming puzzles: A fun and effective learning tool for first programming courses. *ACE '06: Proceedings of the 8th Australasian Conference on Computing Education - Volume 52.*

Dmytro Vitel, Bari A.T.M. Golam, A. G. (2019). Lessons learned from available parsons puzzles software.

Jeff Bender (2021). Spring 2021 field study goals.

Jeff Bender, Gail Kaiser, C. L. J. T. A. G. et al. (2020). 2020 sage gdi study.

Jeff Bender, Bingpu Zhao, L. M. G. K. (2020). Integrating parsons programming puzzles with scratch.

Appendix

A1 Figures

Figure A1.1: Parson's Programming Puzzle activity diagram