COLUMBIA UNIVERSITY

Columbia University, Spring 2021

Final Research Report
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

Contents

1 Overview

This document summarizes research from the Spring semester of 2021, following upon the goals and progress described by the research proposal and midterm report (and associated slides). These resources can be found at https://github.com/cu-sage/Documents/tree/master/2021_1_Spring. Additionally, raw and processed data can be found at https://drive.google.com/drive/folders/1dvx4sYoT9Dl6sDhNOEgzZs_ebax5jr62?usp=sharing.

2 Abstract

The aim of this study was to investigate approaches to making learning computational thinking more effective and scalable through SAGE, while exploring the nuances of novel functionalities. A sample of 746 participants took part in a 10 step computational thinking process wherein cognitive load, performance, efficiency, and motivation factors were measured. Significant differences in transfer performance were detected across all groups exposed to the training. Significant differences between groups were found with respect to aspects of cognitive load and instructional efficiency (as measured by both cognitive load and training time). These results help to reveal the mechanisms underlying efficiency and suggest that feedback and palette presence may play a significant role in training efficacy. Further study is necessary to explore this relationship through the SAGE platform.

3 Related Research

3.1 Integrating Parsons Programming Puzzles with Scratch

The fs1 paper (8) 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 fail to support the hypothesis that PPP training yields the highest learning efficiency. These results drive the investigation posed by fs2 and fs3.

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

The original Parson's Programming Puzzles paper (4) 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.

3.3 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) (5). 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.

3.4 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 (3), 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.

3.5 Instructional Efficiency: Revisiting the Original Construct in Educational Research

Instructional and performance efficiency quantification are essential in assessing the validity of efforts to improve computational thinking. The equations employed in such calculations are largely derived from (10) and the subsequent (6). These papers establish the relationship cited in efficiency calculations in the results section for both efficiency

types. Additionally, (6) notes the equivalence of total time and cognitive load as measures, which leaves two methods of calculation for each efficiency type.

4 Study Purpose

Prior research has established the efficacy of Parson's Programming Puzzles in cultivating coding skills, advancing computational thinking, and increasing motivation among novel programmers (4). Building off earlier background research into Parson's Programming Puzzles and Scratch integration, this study aims to study approaches to making learning CT more effective and scalable through SAGE. Main fields of focus include cognitive load, performance, efficiency, and motivation. The study sample, which is significantly larger than that of prior studies on SAGE, was divided into 9 groups, including a control (parsons puzzle with feedback on a different CT concept than assessed in transfer tasks), to enhance specificity and identify which aspects of SAGE drive efficiency, which introducing new functionality.

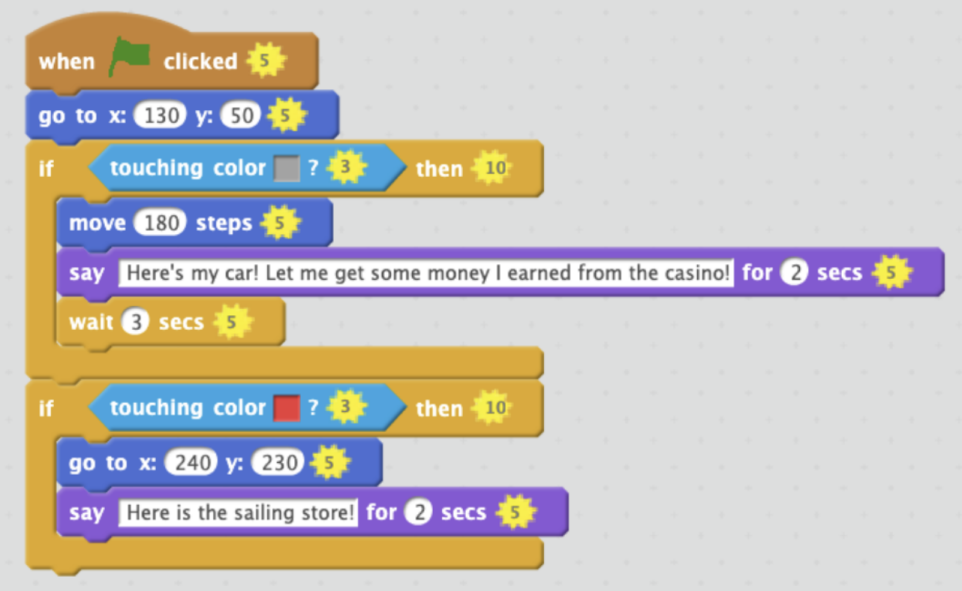

Figure 4.1: Sample Parsons Puzzle Solution in SAGE.

5 Methods

5.1 Study Design

The study pipeline involved a 10-step between-subjects study via Mechanical Turk and Prolific. Steps involved: 1) creation of a username; 2) background survey; 3) review of a 6-minute video tutorial on the UI and CT concept sequences; 4) pretest; 5) pretest feedback; 6) familiarization and training varied by condition; 7) training feedback; 8) posttest; 9) posttest feedback; 10) study feedback. During these steps, participants responded to the validated CS cognitive load component survey (CS CLCS) (2), to a Likert scale based survey assessing attitude (9), and to the Task Evaluation Questionnaire (TEQ) quantifying intrinsic motivation (1). Puzzles individually auto-submitted when correctly completed, or after 500 seconds unless the participant had already submitted a solution attempt.

5.2 Participants

Over 1000 participants took part in individual survey components, though the final number of analysable inputs was reduced due to limited completion of steps. The total number of analysable participants was 746 from both MTurk and Prolific.

Building off the study design of prior work (7), the sample was divided into 9 conditions to help examine the affects of various design parameters:

#	Type	Name	Game Type
1	PPP+f	Let's go sailing! (A)	parsons
2	PPPd+f	Let's go sailing! (B)	parsons
3	ScratchIE+f	Let's go sailing! (C)	parsons (no palette)
4	ScratchIEd+f	Let's go sailing! (D)	parsons (no palette)
5	PPP-f	Let's go sailing! (A2)	parsons (no feedback)
6	PPPd-f	Let's go sailing! (B2)	parsons (no feedback)
7	ScratchIE-f	Let's go sailing! (C2)	parsons (no feedback no palette)
8	ScratchIEd-f	Let's go sailing! (D2)	parsons (no feedback no palette)
9	PPP+f	Your very first recipes! (AC)	parsons

Group 9 was the control condition. More details on conditions are included in (7).

5.3 Materials & Software

Data was collected using the SAGE platform (https://dev.cu-sage.org/#/) and Qualtrics, which output raw data. The majority of analysis was conducted using Python and SPSS. Statistical methodology was dictated by prior studies and related work.

6 Results

6.1 Preprocessing

Prior to analysis, data were consolidated using Python scripts to aggregate participant scores and filter the dataset to subjects who completed the training phase. Condition values were used to subclassify the data and test for the effect of certain variables. For example, groups 1-4 were provided with feedback during training, groups 5-8 were not, and group 9 was the control condition. Composite scores were calculated where applicable (e.g. intrinsic load, extraneous load, germane load, overall cognitive load, motivation, etc.) based on raw survey and training data. Outlier selection was conducted using SPSS plotting. Normality testing was conducted using the Shapiro-Wilk test with normality plots for verification. Based on the data and preprocessing, analysis was conducted using

the Kruskal Wallis H test, Mann Whitney U test, and Wilcoxon Signed Ranks Test.

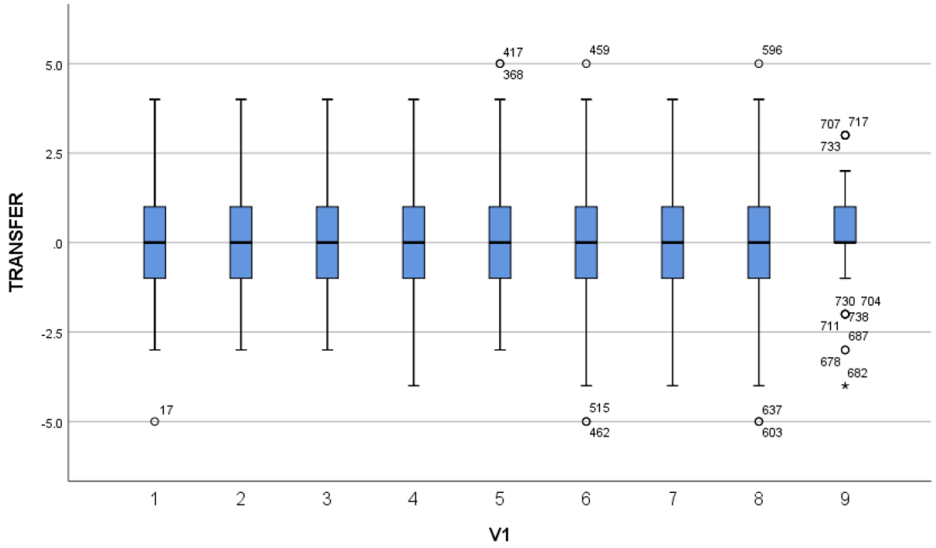

Figure 6.1: Sample descriptive output for transfer performance by condition (V1).

6.2 Cognitive Load

Cognitive load scores were computed at the after pretest (Survey 03), after puzzle (Survey 04), and after post test (Survey 06) stages for intrinsic, extraneous, germane, and overall load. Cognitive load data failed to meet conditions for normality via Shapiro-Wilk testing. No significant difference between groups was detected at the after pretest phase between groups. At the after puzzle stage, significant differences were detected between groups using the Kruskal Wallis H test with regard to intrinsic load (H = 24.55.7, P = 0.001, η^2 = 0.038), extraneous load (H = 20.31, P = 0.005, η^2 = 0.033), and overall load (H = 24.82, P = 0.001, η^2 = 0.034). Mann-Whitney testing yielded significant differences between groups 2 and 6 in intrinsic cognitive load (U = 3601.500, P = 0.028), but failed to find significant differences with regard to feedback as a whole (no significant differences were detected between groups 1 and 5, 3 and 7, and 4 and 8). Inter-group difference between groups 2 and 6 is therefore unlikely to be attributed to feedback as an independent factor. Additional comparisons between various conditions are also underway to elucidate the role of distractors and parson's palettes.

6.3 Performance

Performance analysis was conducted on training and transfer data. No statistically significant difference was found between groups on training performance. Transfer performance (i.e. the difference between pre and post performance data) was found to be greater than zero at a statistically significant rate via the Wilcoxon Signed Ranks Test ($Z = 2.735$, $P = 0.006$, $\eta^2 = 0.011$) for groups 1-8 (Figure 6.2). Notably, group 9 alone displayed no significant transfer performance improvement, which helps to verify the significance of the improvement in groups 1-8 since group 9 was the control condition (Figure 6.2). No statistically significant difference was otherwise found as a result of feedback, nor between groups overall, with regard to transfer performance using the Kruskal Wallis H test. However, further examination between various conditions may illuminate additional insights on performance data.

Test Statistics[a]

	PRE - POST
Z	-2.735[b]
Asymp. Sig. (2-tailed)	.006

Figure 6.2: Sample output for Wilcoxon Signed Ranks Test indicating a significant difference in transfer performance for groups 1-8.

Test Statistics[a]

	PRE - POST
Z	-1.029[b]
Asymp. Sig. (2-tailed)	.303

Figure 6.3: Sample output for Wilcoxon Signed Ranks Test indicating no significant difference in transfer performance for group 9.

6.4 Efficiency

6.4.1 Instructional Efficiency

Instructional efficiency was computed using the methods defined by (10) and (6). The formula for instructional efficiency is given as:

$$E_{instructional} = \frac{Z_{P_{test}} - Z_{E_{learning}}}{\sqrt{2}}$$

$E_{instructional}$ refers to instructional efficiency score, $Z_{P_{test}}$ refers to transfer performance z-score, and $Z_{E_{test}}$ refers to E_{test} z-score. E_{test} in this case may be quantified as a function of cognitive load or total training time, as defined by (6). Instructional findings corroborated this equivalency between cognitive load and training time with respect to instructional efficiency: significant differences were found between groups using both cognitive load (H = 14.81, P = 0.039, $\eta^2 = 0.021$) and training time (H = 40.22, P < 0.001, $\eta^2 = 0.166$) by Kruskal Wallis H test (Figure 6.4).

6.4.2 Performance Efficiency

Performance efficiency was computed using the methods defined in (10) and (6). The formula for performance efficiency is given as:

$$E_{performance} = \frac{Z_{P_{test}} - Z_{E_{test}}}{\sqrt{2}}$$

$E_{performance}$ refers to performance efficiency score, $Z_{P_{test}}$ refers to transfer performance z-score, and $Z_{E_{learning}}$ refers to $E_{learning}$ z-score. $E_{learning}$ in this case may be quantified as a function of cognitive load or total training time, as defined by (6). No significant differences between groups were found with respect to performance efficiency, using either cognitive load or total time as the $E_{learning}$ parameter. These results reflect the findings by (6), in that both measures yielded non-significant coefficients.

6.4.3 Overall Efficiency Results

Sample Kruskal Wallis H test efficiency results are shown below (Figure 6.4). As demonstrated, instructional efficiency was found to vary significantly between groups, while performance efficiency did not. Measure types (cognitive load, total time) were consistent in both cases.

Test Statistics[a,b]

	Performance Efficiency (time based)	Performance Efficiency (CL based)	Instructional Efficiency (time based)	Instructional Efficiency (CL based)
Kruskal-Wallis H	4.721	7.264	40.215	14.808
df	7	7	7	7
Asymp. Sig.	.694	.402	.000	.039

Figure 6.4: Sample Kruskal Wallis H test efficiency output.

6.5 Motivation

Motivation scores were calculated based on the TEQ and within-subject change in programming attitude between the start and end of the study (Background and Intrinsic Motivation). Scores were inverted where applicable and computed for interest/enjoyment, pressure/tension, and perceived competence as dictated by (1). No significant differences were found between groups using the Kruskal Wallis H test. Further motivation analysis will target overall TEQ results, refined subgroup analysis of interest/enjoyment, pressure/tension, and perceived competence, and shifts within attitude as quantified by individual survey questions (e.g. perceiving programming as more fun, more enjoyable, easier to start, less difficult to understand, etc.).

7 Discussion

Results indicate various design factors influence effective cultivation of computational thinking. The overall improvement between the pre-test and post-test stages, as measured by transfer performance, reflects the findings of earlier field studies (8). This finding is essential in assuring that the overall sample subset exposed to the training improved in performance. Differences between individual conditions and sets of conditions in cognitive load, performance, and instructional efficiency indicate that design parameters driving training stage development may play a significant role in participant programming improvement. The presence of feedback and a palette, for example, likely affect the educational efficacy of the training task. However, further investigation is necessary to more deeply explore, quantify, and assess the significance of these results.

8 Paper Submission

The results of the research conducted during the Spring semester of 2021 will be submitted for publication. The current target conference is the SIGCSE Technical Symposium (https://sigcse2021.sigcse.org). The paper template can be found at https://www.overleaf.com/6441998228xhbzjwzmmmmt, and will be updated as the conference approaches. Write-up is underway on several components of the paper (cognitive load, performance, efficiency).

9 Further Work

In addition to the aforementioned fs2 work, fs1 is pending resubmission following editing based on reviewer comments from ITiCSE 2021. Primary goals are reviewing comparable ICCE accepted papers, clarifying protocol content, and considering potential confirmatory factor analysis. Planning is also underway regarding design and analysis of fs3 data, which will further the aims considered by fs2.

References

[1] SDT. Self-determination Theory. https://selfdeterminationtheory.org/intrinsic-motivation-inventory/.

[2] B. B. Morrison, e. a. (2014). Measuring cognitive load in introductory cs: adaptation of an instrument.

[3] 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.*

[4] 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.*

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

[6] Fred G. W. C. Paas, J. J. G. V. M. (1993). The efficiency of instructional conditions: An approach to combine mental effort and performance measures.

[7] Jeff Bender (2021). Spring 2021 field study goals.

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

[9] P. Charters, e. a. (2014). Challenging stereotypes and changing attitudes: the effect of a brief programming encounter on adults' attitudes toward programming.

[10] Tamara Van Gog, F. P. (2008). Instructional efficiency: Revisiting the original construct in educational research.