SAGE Final Report: Spring 2021

Lalitha Madduri, B.S. Candidate in Computer Science, 2021
Columbia University
Supervised by Jeffery Bender, Gail Kaiser

***Abstract*—The main purpose of this paper is to provide a review of work completed on SAGE (Social Addictive Gameful Engineering) in the Spring 2021 semester, namely in the area of Field Study 3 (fs3). The third field study for SAGE is conducted on the computational thinking (CT) concept of looping The three main areas of progress were 1) Puzzle Authoring Across Conditions, 2) Survey Development, and 3) Preliminary User Testing. Progress on the SAGE platform itself included Parsons Puzzle authoring across conditions, configuring of puzzles within missions, instruction authoring and clarification across conditions, and setup of class rosters for user testing. Furthermore, surveys for a study guide and pre-test and post-test on the CT concept of looping were constructed with isometric questions.**

I. Introduction

THE objective of SAGE (Social Addictive Gameful Engineering) is to provide teachers a platform to design puzzles emphasizing computational thinking concepts for students to complete in a self-motivated and rewarding manner. Computational thinking (CT) can be framed as a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use. Thus, SAGE places the programmer, rather than the computer, at the forefront in its learning environment. Previous field studies conducted on SAGE dealt with the CT concepts of sequences and conditionals respectively, while the field study work for this semester, fs3, deals with the CT concept of loops. creating games within the platform across conditions with a gameful, thematic edge. Another key feature of SAGE improved upon this semester are the Parsons Programming Puzzles and associated instructions. We want the students to need to read the minimal amount of instructions in order to solve the puzzles, and shift the focus from instructions to clear directives. Thus, it was important to determine if any variation of instruction needed across conditions.

II. Related Work

i. "Enabling Independent Learning of Programming Concepts through Programming Completion Puzzles"

Parsons puzzles are a relatively new type of practice problem aimed at helping novice programmers [1]. Each puzzle is made of a plain English description of a program's requirements, accompanied by its implementation. This "reference solution" is then scrambled, usually by isolating each line of code into a "fragment" and shuffling them. This paper examines the effectiveness of Parsons Programming Puzzles in teaching CT concepts through independent learning. The researchers compared the learning effectiveness of programming puzzles and tutorial. Participants who learned using programming puzzles spent 23 percent less time in the learning phase and performed 26 percent better on transfer tasks than participants who learned using tutorials. Additionally, they present their work in developing programming completion puzzles that facilitate learning for middle school children. Much like SAGE, they iteratively developed and tested a puzzle interface and curriculum within a blocks-based programming environment, so the work in this paper is valuable in figuring out how to gauge the effectiveness of Parsons Programming Puzzles (PPP) and construct effective puzzles themselves. Key guidelines include only showing the user's work in the program output, limiting the editable dimensions of the puzzle, limiting distractions, and providing the user with ambiguous and incremental feedback towards the puzzle's solution. The puzzle tasks can also be broken down into familiarization, training, and transfer tasks for quicker evaluation and survey optimization.

ii. ”Learning Programming from Tutorials and Code Puzzles: Children’s Perceptions of Value”

This paper ties into the previous one discussed in that it emphasizes the value of Parsons Programming Puzzles, a point that is emphasized within ”Enabling Independent Learning of Programming Concepts through Programming Completion Puzzles.” Though the former paper emphasizes the actual authoring of the PPP and surrounding curricula, this paper focuses more on the perceived value of PPP based on field studies with students. Though this is not as effective considering that the target audience for SAGE for this semester are adult learners, and the participants in this study were 30 participants between the ages of 10 to 15 years, there is still value in the way this paper collects and analyzes the correct data in order to check the emotional effectiveness of PPP. For example, the researchers conducted semi-structured interviews similar to the ones in fs1, developing questions for four semi-structured interviews: a pre-study interview, a post-task interview, an early termination interview, and a post-study interview. Each interview contained questions that sought extended responses and one-word responses and can be used in the classroom with adult learners and instructors alike.

iii. ”Computational Thinking in Constructionist Video Games” This paper describes a genre of learning environments called constructionist video games that are especially well suited for developing learners’ computational thinking skills. Constructionism can be a powerful framework for teaching complex content to novices. At the core of constructionism is the suggestion that by enabling learners to build creative artifacts that require complex content to function, those learners will have opportunities to learn this content in contextualized, personally meaningful ways, something integral to SAGE as a gameful and social addictive platform.

III. Completed Work

i. Puzzle Condition Set

For Field Study 3, the condition set and any participant demographic constraints were first finalized. These conditions will cover the looping CT concept that will be assessed in FS3. There were 9 conditions to be tested in total. In terms of notation, ”PPP” refers to Parsons Programming Puzzles, ”o” marks objectives, ”ScratchIE” marks Parsons Puzzles without a palette, and ”CPP” includes the Constructionist Video Games. Below are the 9 conditions:

1) PPP: Parsons Programming Puzzles 2) ScratchIE : Parsons Programming Puzzles Without Palette 3) PPPo : Parsons Programming Puzzles With Objectives 4) ScratchIEo : Parsons Programming Puzzles Without Palette With Objectives 5) CPP : Constructionist Programming Puzzles 6) CPPie : Constructionist Programming Puzzles Without Palette 7) PPP, PPPo, CPP 8) ScratchIE, ScratchIEo, CPPie 9) PPP, PPPo, ScratchIE, ScratchIEo, CPP, CPPie

Additionally, there is a control condition that spans the computational thinking concept of ”sequences.” 6 puzzles were constructed for each condition, including one familiarization puzzle.

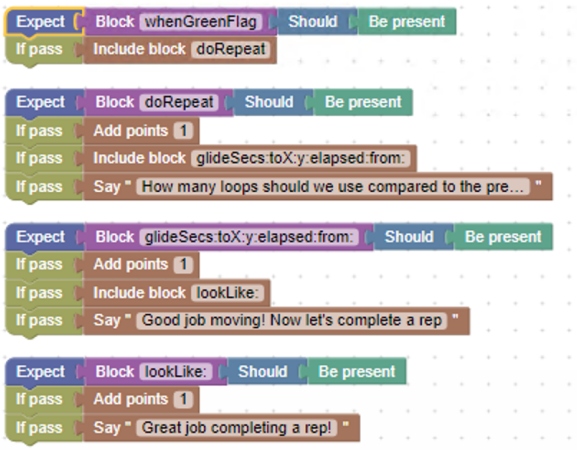
ii. Parsons Puzzle Authoring Across Conditions
Parsons puzzles are a relatively new type of practice problem aimed at helping novice programmers [1]. Each puzzle is made of a plain English description of a program’s requirements, accompanied by its implementation. This “reference solution” is then scrambled, usually by isolating each line of code into a “fragment” and shuffling them. This paper examines the effectiveness of Parsons Programming Puzzles in teaching CT concepts through independent learning.
Originally in the semester, there were issues with the environment as Flash Player is no longer supported on browsers, and there needed to be an additional hotfix that enabled Flash to run retroactively on the browser. Furthermore, there were obstacles preventing the successful loading of Parsons Puzzles from previous field studies due to an issue on the SAGE platform that was also eventually resolved, allowing for puzzles to move from being authored on documents outside of the platform to being authored on the platform themselves.

The goals of puzzle authoring include:
1. Author puzzle programs with motivating scenarios
2. Author puzzle programs with memorable segments
3. Provide a challenge without being tricky
4. Leave the users with a positive impression

As such, for the looping content, the theme of ”Training for a 5K” came into creation. Prior to the 5K training concept, a ”Bank account” theme was developed that involved using loops to see how interest, compound interest, saving, and spending came into play. This was mainly because of SAGE’s focus on adult learners, but this concept was abandoned due to multiple computational thinking concepts coming into play (variables and looping), leading to potential confusion within users. Different instructions were provided for different conditions, namely the instructions for puzzles including objectives (B1), objectives without a palette(B2), and constructionist video games(C). Looping instructions were constructed with block explanations and introductory guidelines and are available here: Instructions

The 5K Training concept begins with a warm-up that asks the user to make the sprite complete 3 jumping jacks, providing instructions on how to complete a jumping jack. A new concept, costume changes, is introduced in this looping concept within the ”Looks” pane, visible within the user’s PPP when he or she starts. This concept intends to make the play more gameful for users, as there is more variation with animation on the screen than prior puzzles which simply utilized the ”Say” block in the ”Looks” pane. The following puzzles iterate on this concept with differing varieties of loops and complexities. For example, another puzzles utilizes mainly ”Motion” pane blocks and a repeat until that involves the sprite reaching a finish line. Further puzzles include making a sprite run around a track, using the ”forever” and ”repeat until” loops to change the type of ”exercise” performed by the sprite, and then running around a track while doing a jumping jack/squat sequence at each corner. The puzzles in this concept were replicated and modified across conditions, with the mappings from missions to quests to conditions is detailed here: Mappings

Functionality for game objectives has been largely implemented since the beginning of the semester, which is at the forefront of authoring as well. Game objectives and Constructionist Video Games are new for FS3, and in previous field studies, the only conditions tested were feedback inclusion and palette inclusion. Objectives, which create two new conditions for users to test across, provide the students with additional guidance as they go through solving the puzzle. Objectives focusing on actual/matcher: block/presence, block-type/presence, sprite/say were implemented within the 5K training puzzles. Example configurations of objective blocks are, for example, when block type should be present: if pass, say [next step guidance] if fail, [current step guidance] if pass, include [next blocks]. Another possible configuration is sprite should say: if pass [text if any] if pass [next blocks]. Below is an example of the objective editor and the block functionalities within.



Within each objective, there were 2-3 points expected to be added when core portions of the puzzle were solved. If the user completes all of them, they are presented with an ”Objective Completed!” message in the side pane. Additionally, within the ScratchIEo condition, when the participant completes a puzzle with a palette provided, the puzzles constructed originally only start with the When Green Flag is Clicked block being unlocked. As the correct blocks are dragged onto the stage, more and more blocks become unlocked with objectives, allowing programmers to approach solutions more iteratively in their approaches. This shift in expectation in block unlocking is reflected within the instructions and

clearly delineated. When specifying Block Type and Block in Actual constructs, it is important to match the Block Type to the category name (e.g. “Looks”) in the categories array and Block to the opcode (e.g. “forward:”) in the commands array in Specs.as. Additional documentation was created to facilitate instructors with this creation process.

Meanwhile, within the Constructionist Video Games condition, puzzles did not have correctness feedback or point updates as the student went through the solution; rather, ”say” blocks in objectives were used to provide guidance towards the correct condition, or even offer alternative ways of building the blocks to their correct solution. We considered configuring the CVG condition so that explicit step-by-step instruction is not necessary if game objective feedback offers sufficient progress and completion guidance, and move-based PPP feedback is disabled. This configuration would optionally facilitate an alternative approach to looping instruction in which the participant is encouraged to construct a sequence first, then focus on the loop. In PPPs, it is less confusing when the participant is guided to build the puzzle from top to bottom, since when puzzles are solved that way, the result is a positive flow of correctness feedback and an increasing score. Objectives across conditions were at first differentiated, but after receiving feedback, the same objectives were used across conditions for puzzles. There is also a possibility for objectives for Constructionist Video Games to be different, as we aim to guide students towards the different constructionist methods to approaching the correct solution for the Parsons Puzzles in SAGE. This idea could be developed in future iterations of fs3.

ii. Surveys Surveys for the fs3 study guide, looping content pre-test and post-test were created, available within the links. The qualtrics fs1 pretest posttest and fs2 pretest posttest are available, but are arguably programming-oriented. For the fs3 pre-test and post-test, we aimed to emulate the following principles to highlight computational thinking assessment over programming skills assessment. The qualtrics surveys are currently not on the premium level required to give them full functionality; this needs to be completed by a user with a premium account.

1. No blocks and only text
2. Screenshots of stage with sprite and question related to number of blocks required to reach designated spot
3. Probe understanding of value (select reason for using loop out of incorrect reasons)
4. Identify puzzle in which looping would be useful (two types, 1 text, 1 picture)

iii. Preliminary User Centered Design Sessions

During the creation of the Parsons Programming Puzzles in the earlier half of the semester, I tested the looping content puzzles with two adult learners who I’ve identified as potential test users for SAGE. The class and roster creation functionality will be implemented and the learners will be given separate logins to assess the platform and give feedback on the looping content across conditions. Learners will begin with PPP (with a palette provided) and also be assessed on the use of objectives within the looping content, and any difficulties with either the platform, curriculum, or conditions will be noted. These preliminary user centered design sessions will act as guiding forces for more involved user-centered design sessions once looping content across conditions is finished being developed.

Feedback from the earlier testing sessions is available here, while feedback from later testing sessions is available here. Furthermore, many bugs appeared when testing the puzzles with users, which are documented within the bugs file.

IV. Future Work

For future work regarding fs3, comprehensive testing will need to be conducted on individuals end-to-end using the fs3 study guide. This would include the participant engaging with the study from the pre-test, to the puzzles, to the post-test, to exit surveys. Puzzles with objectives will only be able to be tested comprehensively when the bugs outlined in the earlier section are resolved, as they provided too much obstruction to user testers in being able to solve the problems. The qualtrics surveys also need to be appropriately configured and completed,

with links in the study guide to accompany each question.

REFERENCES

[1] 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."

[2] K. J. Harms, N. Rowlett and C. Kelleher, "Enabling independent learning of programming concepts through programming completion puzzles," 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Atlanta, GA, 2015, pp. 271-279.

[3] K. J. Harms, E. Balzuweit, J. Chen and C. Kelleher, "Learning programming from tutorials and code puzzles: Children's perceptions of value," 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Cambridge, 2016, pp. 59-67.

[4] David Weintrop, Nathan Holbert, Michael S. Horn, and Uri Wilensky. 2016. "Computational Thinking in Constructionist Video Games." Int. J. Game-Based Learn. 6, 1 (January 2016), 1–17. DOI:https://doi.org/10.4018/IJGBL.2016010101