PROJECT PROPOSAL
Parson Programming Puzzles and Field Studies

October 3, 2019

Rebecca Cawkwell rgc2137
Emily Li el2895
Sandy Zhao bz2328
Juvaria Shahid js5160

Contents

0.1 Abstract

SAGE's mission is to create a program to develop computational thinking (CT) skills in 6-8th. In order to do this, there is a need to create Parson Puzzles for various CT concepts and at different levels of difficulty. It is also important to test our progress of these puzzles and of the overall program of SAGE by completing field studies with middle school teachers. For this semester we plan to focus on the building of Sequences and Loops Parson Puzzles and the Field Studies.

0.2 Introduction

Computational Thinking (CT) describes the problem-solving skill of thinking in the way a computer would solve a problem. Today, there are more computing jobs than any other type and not a sufficient number of graduates to fill those jobs. It is important to create educational programs that will teach the next generation of students CT in order to meet the needs of our economy. These educational programs need to be accessible, usable and inclusive to students of all backgrounds.

SAGE is a variation of Scratch that provides teachers a systematic way to build students' CT abilities. CT is broken down into concepts that describe various computational methodologies to solving a problem such as loops, conditionals, etc. Each of these CT concepts needs to have various Parson Puzzles designed in SAGE for students of different difficult level.

In order to know the effectiveness of this program, it is important to evaluate the effectiveness and usability of the program. Teachers can be evaluated to find gain insights into how students may respond to the puzzles as well as how usable the program is. Students can be evaluated to find the effectiveness of the designed Parson Puzzles and the usability of the system.

0.3 Related Work

0.3.1 Parson Puzzle Design

How Do Students Solve Parsons Programming Problems? An Analysis of Interaction Traces (Helminen, Ihantola, Karavirta, Malmi)
This study primarily focused on understanding the process of how students work through

and solve Parsons programming puzzles. The study was conducted on university students in Finland who were asked to solve five different programming puzzles. Interestingly enough, it was found from this study that the students used the automatic feedback function very sparingly while solving the assignments. The research also provided insight on common patterns that students used when solving problems, such as adding all the code fragments in a linear top-down order or first adding code fragments that defined a block or control structure. Through this study, they were also able to observe common difficulties among the students and noticed that their solution paths often had loops. The study categorized this looping student behavior as backtracking, circular loops, separate sidetracks, concentrated sidetracks, and jumbled combinations.

Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking (Moreno-LeÃşn, Robles, RomÃąn-GonzÃąlez)
In this study, they proposed a web application named Dr.Scratch to facilitate teachers' evaluation of their students' CT proficiency and to provide effective and comprehensive feedback for students in order to motivate learning. Result has shows, through the incorporation of Dr.Scratch, student's overall CT scores were improved. Consequently, their coding skills are also improved. One of the ways that Dr.Scratch use to give effective feedback is by detecting insufficiency of logic. Thus, even students are able to complete the current task, they can still have not mastered the CT concept due to the use of repetitive codes. Dr.Scratch is able to detect that and further enhance the learning experience beyond correctly completing a task. Furthermore, Dr.Scratch, through analyzing the Scratch project, gives different amount of information to students at different levels since they have found too much information can be counterproductive for beginners.

Understanding Computational Thinking Before Programming: Developing Guidelines for the Design of Games to Learn Introductory Programming Through Game-Play (Bacon,Kazimoglu,Kiernan,MacKinnon)

This paper primary focuses on improving CT for first year undergraduates. It emphasises the difference be learning and gaming in GBL and the importance of designing challenges based on individual abilities. Bacon et al. believed that GBL should take into account students' prior knowledge, learning progress, preferences, etc to provide an individualized learning experience for all. They also made the distinction between learning CT concepts and CS concepts, in that mastering CT should enhance one's critical thinking and problem solving skills in any discipline. Moreover, they stressed the importance of receiving clear feedback to help beginner learn from their mistakes, such as learning early on the re-usability of logic and code. They claimed meaningful feedback that are different from compiler messages would also motivate students to reach their learning goals. Lastly, they thought it's crucial to make all GBL gender and expertise neutral to encourage learning for all audiences.

0.3.2 Field Studies

Distractors in Parson Puzzles Decrease Learning Efficiency for Young Novice Programmers (Harms, Chen & Kelleher)
In this study, Harms, K. et al. studied the difference between learning PP with and without distractors. Overall they found that distractors did not increase their "transfer task performance" and decreased their learning efficiency. They studied three types of distractors: extra noise, familiar suboptimal paths and only one possible solution. They found that extra noise was easy to ignore. Suboptimal paths led to students not creating the correct solution and not understanding why their solution was not correct. Only one possible solution was the answer to this problem. However, in analysis they found that distractors created only extraneous cognitive load which is harmful to learning. They suggest that these results may be different for students who have prior programming experience and in different educational contexts. They also suggest the study of the impact of distractors on the motivation of students.

Enabling Independent Learning of Programming Concepts through Programming Completion Puzzles
This paper focuses on formative evaluation techniques. The group iteratively tested and developed a puzzle interface within a block based environment. It found that participants who learned using the puzzles did 26 percent better on transfer tasks and 23 percent in the learning phase.

Their formative evaluation was conducted on 23 users (age 10-15). Each session lasted 30 minutes.
The evaluation was broken down into two parts:

1. Designing the programming puzzles Through ten iterations of formative testing
23 users (1-15 years age) were evaluated. Each session was 30 minutes.
Some lessons learned regarding the format and interface:

- Only show the user's work in the program's output
- Limit the editable dimensions of the puzzle
- Limit distractions and focus on the user's attention on the output
- Provide user with ambitious and incremental feedback towards the puzzle's solution

2. Creating the programming puzzle curriculum
Focused on Repeated execution, parallel execution, parallel nested within repeated execution, and repeated nested within parallel execution.
Evaluated 21 participants ages 10-15. Each session was 90 minutes.

0.4 Proposal

Preparation

We will build upon the surveys and templates created by Ben and Jady this summer. We will design the field studies based on methodologies that gain the most amount of feedback from teachers. Additionally we will use our resources to connect with potential schools and teachers that we can talk with.

We could potentially work with teachers from the Upperline School of Code. Upperline School of Code employs around 20-30 teachers each summer to teach software engineering boot camps. Most of the teachers teach in schools in New York City. If we are able to secure a couple of teachers, we will have access to several classrooms. In the past, SAGE has connected with PS175, so these could be secondary locations to conduct our surveys for the pilot release.

Parson's Puzzle Design

For our first iteration of puzzles, we will design 10 puzzles focusing on testing loops. Using this initial set of puzzles, we will work with and observe 4-5 professors and teaching assistants as they work through these puzzles to gain insight on their thought process. In doing so, we hope this will inform us of more difficult sections that may need more assistance and feedback and also help us understand how experienced programmers structure their problem solving approach. This will allow us to refine our puzzles for the next iteration.

We want to design multiple themes and characters for the puzzles to ensure all students are able to choose their preferred option. This would also ensure the design is friendly and motivating for all students. In addition, we will design multiple versions of the same problem by randomizing the parameters for each problem, so students would be able to practice with different problems at the same level of difficulties.

We have also decided to not use distractors for the more basic levels of the puzzles, because the paper by Harms, Chen & Kelleher suggested that distractors are not improving students learning outcome at an early stage. However, as we get into designing more difficult puzzles, we would like experiment if distractors enhances learning at a more advanced level.

Field Study Implementation

As we send out emails to teachers that could potentially work with us, we will provide two options: either they can choose to give us their teacher-side feedback or they can help set-up a study with their students.

If the teachers choose the first option, we will hold a one to two hour session with them in which we walk through SAGE and sample puzzles. Before the walk-through, we will give a survey that gauges their experience teaching CT, Parson Puzzles, programming languages (specifically block-based languages such as Scratch). As we walk through SAGE and our current parson puzzles, we will gather their reactions and comments. At the end of the orientation, teachers will fill out a post-session survey to verbalize their feedback.

If the teachers choose the second option, we will file an IRB if needed and schedule our study for after the creation of a satisfactory version of the loop puzzles. Teachers will be asked to pick 15-20 students who are willing to participate in the study. We will host an orientation session with the teachers. Then, we will conduct our study over two days. On the first day, we will conduct a pre-test (to assess the student's computation skills) and present a tutorial on SAGE. On the second day, we will teach the concept of loops with a short presentation and then provide the loop SAGE puzzles for the students to work through. After, we will ask the students to complete a post-test survey.

Analysis and Incorporation of Feedback

After gaining feedback, we will analyze the results of the various surveys and observations we make. In turn, we will use this feedback to make recommendations for changes of the SAGE system and for our own design of Parson Puzzles. After gaining these insights, we will make changes to our Parson Puzzles.

0.5 Timeline

Date	Milestone
Sprint 1	Proposal and Pitches, Set up Environment, Send out Field Study Emails
Sprint 2	Finalize PP Design Process and start building + Finalize field study design
Sprint 3	Continue to work on PP + Complete field studies and analyze results
Sprint 4	Incorporate field study feedback into Parson Puzzle Design