Project Proposal

Parson's Programming Puzzle Scoring System

Junyi Wang
Yilan He
Ningchao Cai
Yiming Guo

COMS 6901 SECTION 028

Sep 21, 2018

CONTENTS

1. Abstract

Throughout the development of the Social Addictive Gameful Engineering (SAGE) project, Parson's Puzzle has been successfully developed and integrated. However, some core functionalities such as scoring system are not well implemented. Additionally, the current system's user interface has much space to improve. In this project, we aim to focus on the Parson's Puzzle 1.1 Feature in the Gameful Direct Instruction Epic. We would put special emphasis on the scoring system and the user interface.

2. Introduction

Parson's programming puzzle is an entertaining way for students in the basic programming level to learn how to code. And it is also a great way for students to get in touch with various programming topics and typical errors. Teachers can embed this type of game in the student system to expose students to coding practice, and to teach students better in the future based on the performance of the students from different backgrounds and levels. Therefore, it is worthwhile for elementary and middle schools to explore the way to develop an entertainment system with this interactive tool. Currently, the SAGE project has utilized this idea and developed a such system. In our project, we aim to further improve the system by focusing on its scoring system and user interface.

While the system's main component, namely the Parson's Puzzle, is functioning, one of its most critical functionalities, the scoring system, still requires more work. According to Turner (2011), the teacher's knowledge of how students learn can make the difference between effective and ineffective teaching. One of the sources of such knowledge is the performance of students. In our system, it could be quantified as the score. Therefore, it is crucial to have precise and comprehensive metrics to evaluate students' performance, or teachers' teaching plans and strategies would be off in the first place.

User interface will also be emphasized in the project. As the puzzle is targeted at students in grade six to eight, clear instruction and interface are required for smooth usage of the system. Confusion about the system would be a confound variable and in turn would add noise into the feedback collected by teachers and researchers. Great interface design would not only allow students to use the system without obstacles, but also positively reinforce the students to continue the usage of the system.

In the following sections, we will first discuss works that are related to our project. Then we will dive into detail regarding the issues with the current system and our approaches to resolve the issues.

3. Related Work

Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses (Parsons & Haden, 2006)

In this paper, Parsons and Haden proposed the Parson's Programming Puzzles. The paper talked about the design of the puzzle in detail. Unfortunately, the paper does not talk about how score could be inferred from the puzzle. However, the paper does suggest that the order, the selection of distractor, time and trials are critical factors in evaluating the performance. Additionally, the study conducted suggested that better user interface such as positive rewards would make the puzzles more entertaining.

Student-Centered Instruction: Integrating the Learning Sciences to Support Elementary and Middle School Learners (Turner, 2011)

The author reviews the knowledge bases of the learning sciences and presented what researchers and teachers have learned in the past 30 years. The author proposed an instructional guide for researchers and teachers to follow when creating learning environments. In the paper, the author emphasized the importance of feedbacks. The author mentioned that the educators should ensure that the feedbacks are utilized to improve teaching. The author also mentioned that measures of knowledge transfer should be included in the assessment.

Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods (Kiesmueller, Sossalla1, Brinda & Riedhammer, 2010)

The paper describes a way to identify the students' problem-solving strategies, namely Top down, Bottom up, Hill Climbing and Trial and Error, using hidden Markov models. The paper mentions in addition to the learners' solution attempts and quality of attempts, the solving strategy should also be taken into consideration.

4. Proposal

● Scoring System

Stage 1: Develop comprehensive metrics

Currently, the basic system uses a rudimentary scoring mechanism as shown below in Table 1. The current metrics leave out much nontrivial information. To start with, the metrics could only capture whether the block is at the correct location or not. It could not tell how wrong the current answer is. For example, having two correct blocks placed in wrong orders would yield the same result as having two random blocks placed in wrong orders.

Secondly, as suggested in Parsons and Haden's work (2006), factors such as time and number of trials should be taken into consideration. Other than these, if multiple answers exist, the final score should end up different based on the code's style. For example, use a for loop to execute an event 10 times should gain more

S.No.	Move	Score	Comments
1.	Block moved in correct order to correct location on script pane	Points incremented based on points associated with the moved block	
2.	Block moved and attached in incorrect location in script pane	Points decremented based on points associated with the moved block	
3.	Hint clicked	Every time hint is used, -1 from the score	
4.	Incorrect moves	-1 for every incorrect move	This is implemented to ensure student does not apply trial and error method to reach the correct solution

Table 1: Current Scoring Metrics

scores than repeating the event 10 times manually. As suggested by *Kiesmueller, Sossalla1, Brinda & Riedhammer* (2010), the problem-solving strategy should also play a role in scoring. For example, dragging the correct blocks into the correct position in random sequence should earn fewer scores than solving the problem with a top-down approach. The current mechanism does not capture this.

Moreover, the learning system should detect students' actions when they try to answer a question. In the previous version of the system, it supports pausing and undoing actions. However, we know that these actions are important aspects of scoring. These indeterminable factors should be taken into consideration. On one hand, the system should notice students' actions and modify their scores accordingly. On the other hand, the basic rules can’t be broken when the system modify student’s score, such as scores can’t be negative, and so on.

Lastly, for a specific quiz, there can be more than one programming question. How to make sure that each question has a score that is consistent with the corresponding difficulties is a tricky problem. Similarly, we need to ensure that for each question, every student is graded with the same criteria and we would like to minimize as much subject scoring as possible.

To approach these problems, we propose the following actions:

a. Analyze the current data retrieved from the previous system. We want to find out whether there exist some patterns that can help us improve the current scoring system and, if necessary, revise the scoring mechanism.

b. Develop comprehensive metrics. Take time duration, trails, problem solving strategy and students’ actions into consideration. We would need to figure out the weight for each additional metric.

Stage 2: Use statistical analysis and machine learning to improve the system

Another approach we could take to improve the scoring system is to utilize the data we have. To begin with, we could take statistics approaches to adjust the scores among the students, in order to rule out noise and potential unfairness. Additionally, we could use machine learning and deep learning to learn the weights for each metric, if we have enough data. We would try out different models to improve the handpicked weights of the metrics.

● Recommendation System

For students, they might need to do different problems to practice programming. After finishing a quiz or a puzzle, unless the students are required to do the quizzes in a specific order, students can then choose any other quizzes they want in the system. What we want to do is to provide some recommendations after a student successfully finish a quiz based on his or her score and the concepts that might need to be covered in the next quiz. We hope that we can do this by using some prediction models.

To achieve this, we have the following action items:

a. Build up our recommendation system on the basis of the revised scoring system. This means that our recommendation system will partly rely on the students' scores.
b. Figure out a way to quantify the difficulties of each puzzle (or just based on the ones that are provided in the current system). If possible, we might try to figure out the difficulties automatically by using machine learning models. However, this is tricky since the weighing process is mostly subjective.
c. Explore several recommendation algorithms to provide automatic recommendation for students. We first need to decide whether our system should be factor-based or content-based. Then the most important part in this step is to adjust the weight of each feature in the selected model to improve the

performance of our model. And if time allows, we could try out some different algorithms to show the accuracy of our model.

- **User Interface Improvement**

We have studied the current SAGE site for the Parson's Puzzle (Figure 1). The current system already has the basic function contains the timer, user panel etc. However, these functions are not well implemented. For example, the time is not used in scoring and the instruction is not finished well.

Based on these drawbacks and the feature backend will add, we will implement the front end in following aspects.

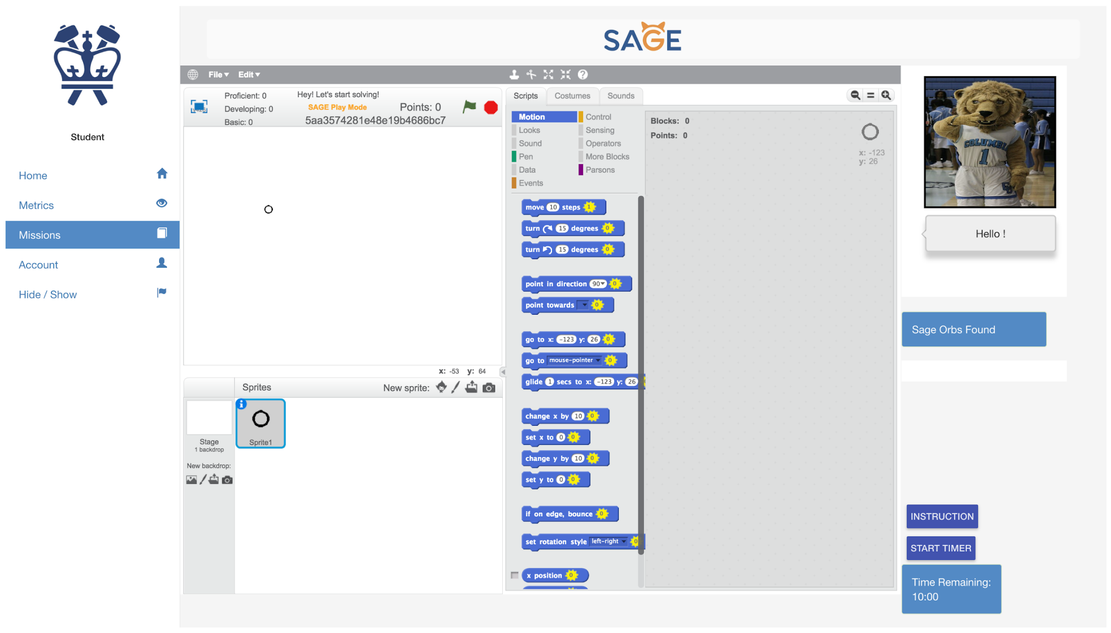

Figure 1: Current web design

- **Popup instruction**

  Although current page has the instruction button, the function of this button is not obvious nor intuitive. When we first come to this page, it took a long time trying to figure out how the system works. That is user unfriendly, especially for students. Therefore, we suppose adding a more interactive instructional panel or tutorial might be a great help. Technically speaking, this part could be done using a front-end modal panel which pops up necessary information. We can highlight important details in CSS.
  The current system has very few interactive components except for the main Scratch block.

  For the students who come to this page for the first time, we can pop up basic steps to tell the students follow, it would be better for them to catch up with the main idea of the game. We also noted the question setting is not clear and the students might not know what to do at the first sight. A pop-up window for the questions could also help with this problem.

  For the students who have used the system before, they could obtain information about the buttons and icons' functionalities by checking the "?" besides them.
  With the popup instruction, the puzzle game will be more user friendly.

- **Scoring**

  The UI component in the current interface is quite simple and has little interaction with the plug-in Scratch block. We are wondering if we could get the scoring parameters right from the Scratch API if it exists. We will focus on the parameter passing from the frontend to the backend as part of the scoring and hint systems that we aim to produce. Also, we want to check whether we have any way to make the design of this compound system look more consistent in visual style. We plan to design the interface using a more user-friendly approach by using the usability principles and if possible, adding some animations.

Also, in terms of the functionalities, we could add more complex factors to the grading. For example, the scores should vary with student's response time, mistaken moves, and number of hints used. To accord with the backend functionalities better, we will make interactive components such as progress bar and charts.

In addition, as an incentive of learning, we plan to add a result board showing a percentage indicating the scores of the current student compare to others. For example, a message like "Congratulations! Your scores beat 70% of the players.". This approach is adopted by LeetCode as we can see from the figure below.

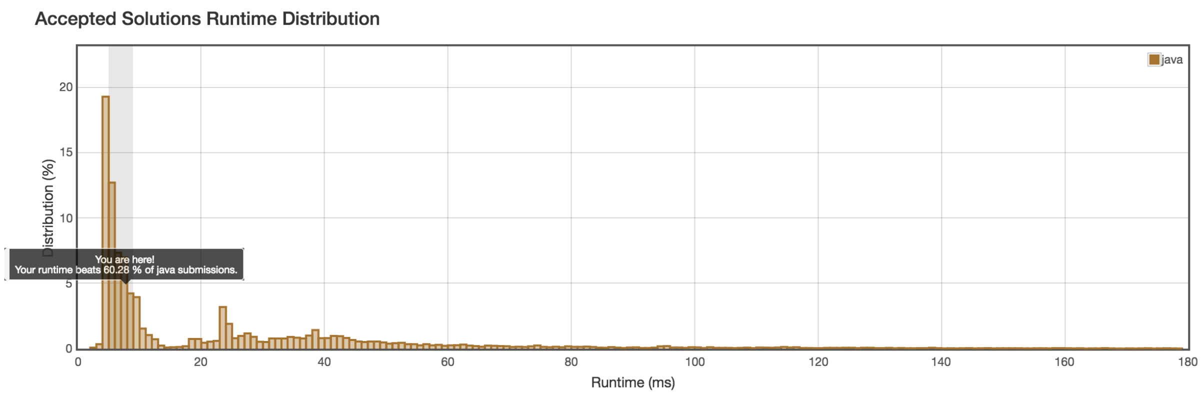

Figure 2: LeetCode Percentile

This could serve as an encouragement for students who have demonstrated a certain level of skills. According to Tomlinson (2005), it is important to promote opportunities to build student self-worth and self-efficacy. From a psychology perspective, motivations like this serve as positive reinforcements, which would encourage the students to participate more frequently in the current activity as well. For beginners, we could adjust the message as needed.

- **Difficulty levels**

  We are also concerning the difficulty setting of the game. From the current interface, we find it hard to adjust the difficulty level. Thus, it might be good to either allow the students to toggle the difficulty modes before rendering the quiz or after entering the main quiz screen.

  Besides, we will give the suggested difficulty level to the students after they finish one puzzle based on the score they got. This interface will be based on how backend implemented the recommended system.

- **Timer setting**

  The original time has the advantage of giving the user some degree of freedom to control it. The basic function of the timer is satisfied. Though, in some ways, the place of the timer is not obvious enough. We hope to give the students some senses of the relationship between the time elapsed and the performance they reflected on the quiz, which is measured by the scores. At the stage, we hope to do some minor animations as the time reaches a range where the score will be changed. Thus, the student could get the idea of the time used will affect their scores, thus encourage the efficiency of their problem solving.

Technically, we will be developing the frontend system in Node.js and Angular.js as in previous versions. We will also look into the Scratch component and see whether we could do some modification on its UI parts to ensure more consistency across the whole system. Potential frameworks and tools that could be used include Bootstrap and Sass.

5. Proposed Timeline

- The 1st week: go through the current version of system, understand the existing functions in the system, and inspect the code
- The 2nd week: for the frontend team, start designing the user interface; for the backend team, retrieve the previous data, find some useful metrics from the data and start model selecting.
- The 3rd week: for the frontend team, experiment on the communication between the Scratch component and the outer UI; for the backend team, choose metrics by analyzing the importance of each metric in the scoring system.
- The 4th week to the 8th week: for the frontend team, making the actual components including the pop-up window, instructional panel, and the new scoring section; for the backend, the scoring model should be trained with a decent accuracy rate.

- The 9th week to the 12th week: for the frontend team, enhance the system function and interactivity; for the backend team, start designing recommendation system, choose an appropriate algorithm / model and improve the recommendation results.
- The 13th week: Integrate the whole system and test the system.

6. Reference

Kiesmueller, U., Sossalla, S., Brinda, T., & Riedhammer, K. (2010, June). Online identification of learner problem solving strategies using pattern recognition methods. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 274-278). ACM.

Parsons, D., & Haden, P. (2006, January). Parson's programming puzzles: a fun and effective learning tool for first programming courses. In Proceedings of the 8th Australasian Conference on Computing Education-Volume 52 (pp. 157-163). Australian Computer Society, Inc..

S. L. Turner. "Student-centered instruction: integrating the learning sciences to support elementary and middle school learners," Preventing School Failure, vol. 55, no. 3, pp. 123-131, 2011

Tomlinson, C. A. (2005). Differentiating instruction: Why bother? Middle Ground, 9(1), 12–14.