Parson’s Programming Puzzle Scoring System Improvement

SAGE Project Proposal

Haoxuan Huang (hh2773)
Zijie Shen (zs2411)

Content

1. Abstract

Through the development of the Social Addictive Gameful Engineering (SAGE) project, Parson’s Puzzle has been already pretty well developed and integrated. However, some core functionalities, such as scoring system, have not been well implemented yet. In addition, features like score persistence still have space for improvement. 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 two stories: Parson’s Scoring and Parson’s Score Persistence.

2. Introduction

Social Addictive Gameful Engineering (SAGE) is a project that aims to build a collaborative game-based learning and assessment system that infuses computational thinking in grade 6-8 curricula.

For Gameful Direct Instruction approach, SAGE embedded Parson’s Programming Puzzle authoring and gameplay in Scratch which is a visual programming language for children developed by MIT Media Lab. Parson’s Programming Puzzle is a game that allows students to drag code fragments to build the correct program[1]. In SAGE, teachers design the puzzles and evaluate students based on their scores. While solving the puzzle, students receive immediate feedback of each move as a guiding.

Even though SAGE has already provided many functionalities, there are still various suggestions from students that require corresponding actions. So in this proposal, we will focus on improving existing Parson’s scoring system and providing Parson’s score persistence.

3.Related Works

3.1 Scoring System Design Principles

The scoring system used in any kind of game can have considerable influence on the satisfaction of players during gameplay. Scoring acts as a type of positive feedback and reward system capable of spurring players on toward greater challenges. Scoring often serves as a bridge between games and players, and thereby provides an indication of the degree to which game players are intent on achieving the objectives of the game. In other words, scoring is a way of measuring success.

Three aspects of scoring systems should be considered in the design of games: perceivability, controllability, and relation to achievement. Perceivability refers to the extent to which players are aware of the existence of the scoring system. This affects how immersed players become in the game and what gaming strategies they develop. Controllability affects the freedom of players to manipulate their own score and whether they can employ multiple game strategies. Relation to Achievement affects the lifespan of the game. A player who no longer feels challenged to achieve something in a game is significantly less willing to continue playing the game. The greater the level of achievement offered by the scoring system, the greater the level of challenge.[2]

Therefore, we aim to redesign or improve the current scoring system by following these three aspects.

3.2 Parson’s Programming Puzzle

Parson’s Programming Puzzles is an automated, interactive tool that provides practice with basic programming principles in an entertaining puzzle-like format. It comprises a set of drag-and-drop style exercises designed to provide students with the opportunity for rote learning of syntactic constructs in Turbo Pascal. It is a wonderful replacement of the common drill exercises which are boring and there’s difficulty of removing a single syntactic unit from the logical context in which it occurs.

In each problem context, the student is given a selection of code fragments, some subset of which comprise the problem solution. The student needs to drag their choices of the draggable fragments into the indicated answer locations. The student is encouraged to repeat the problem until 100% accuracy is achieved.[1] In general, for each attempt, a score should be computed and shown to the student. The score is output by the scoring system, which is one of the main focus in this proposal.

3.3 Computational Thinking

Computational thinking is a set of problem-solving methods that involve expressing problems and their solutions in ways that a computer could execute. It requires thinking at multiple levels of abstraction and many other aspects, which are important skills not only to scientists but also everyone else. The influence of computational thinking on other disciplines is already witnessed including statistics and biology. So it is necessary to expose pre-college students to computational methods and models[3].

Computational thinking also complements and combines mathematical thinking given that computer science inherently draws on mathematical thinking[4]. Therefore, it would be most effective and efficient if students could improve both at the same time. An experiment with sixth-grade students seems to indicate that Scratch can be used to develop both students' mathematical ideas and computational thinking[4]. So it is reasonable to choose Scratch for computational thinking development.

4.Proposal

4.1 Parson's Scoring

In Veronica Woldehanna's final project report in Spring 2019, she also touched this topic about improving the scoring system. In the report, she mentioned she used a heuristic based on Manhattan distance to differentiate between lines of code that have errors ranging from few to numerous. In the previous implementation of the scoring system, for each line which is out of order, the student would get a point deduction the same value as the reward from each of the line which is correct. Veronica’s improvement is to implement a scale scoring system. In other words, how much point is deducted is based on how far away each line is from the correct position. However, this implementation based on the idea of manhattan distance has some flaws. For example, the lines which should be at position 3, 4, 5 get placed wrongly by the student at position 7, 8, 9. According to Veronica’s implementation, the student would be deducted 3*4 = 12 points. However, this method of deduction totally ignores the relatively correct order of these three lines themselves. Although, in general, the student’s solution is wrong. But at least for those three lines, the student gets the logic correct. Therefore, we’re thinking a better scoring system which can include the consideration of the relative correct order in the answer. In addition, we also want to implement some other scoring system features such as score based on difficulty levels.

4.2 Parson's Score Persistence

Currently, the scores of students’ Parson’s Puzzle are not saved. Instead, we only have feedback to students when they are working on the puzzles, which limits teachers’ access to each student’s Parson’s Puzzles results. Therefore, we need to save necessary information to database when students finish each puzzle. After that, teachers can evaluate students based on individual’s results and objective of each Parson’s Puzzle. The evaluations from teachers should be an important factor to student’s CT score.

In mongoDB, we can have a collection for puzzles’ results in which document contains score, start and end time, hint usage, and puzzle_id which refers to a specific Parson’s Puzzle. Each user document should store result_id as a list for all puzzles solved by the user. Therefore, teachers could have access to all puzzle results of a specific student.

5.Timeline

Sprint 1: Play around with SAGE and read the source code.

Sprint 2: Implement Scoring System

Sprint 3: Use the newly implemented Scoring System to improve score persistence

Sprint 4: Survey about the best scoring system and debug the code

6.References

[1] Parsons, D. and Haden, P. (2006). Parson's Programming Puzzles: A fun and effective learning tool for first programming courses.

[2] Lee, C. , Chen, I. , Hsieh, C. and Liao, C. (2017) Design Aspects of Scoring Systems in Game. *Art and Design Review*, **5**, 26-43. doi: 10.4236/adr.2017.51003.

[3] J. Wing. Computational Thinking. Communications of the ACM 2006, 49 (3), 33-35.

[4] Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2019). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. Interactive Learning Environments, 1–12.