Parson's Programming Puzzle Scoring System Improvement

SAGE Final Report

Haoxuan Huang (hh2773)
Zijie Shen (zs2411)

Content

1. Abstract

In the final report, we will talk about the entire progress we've made this semester. Our work for this semester includes improving both the scoring system and the score persistence feature. The relevant Epic Gameful Direct Instruction and Feature is Parson's Puzzle 1.2. For the first half of the semester, we focused on Story Parson's Scoring. We successfully designed and implemented an improved scoring algorithm. For the second half of the semester, we focused on Story Parson's Score Persistence and also made the scoring system more perfect.

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.

3.4 Feedback Algorithm: Longest Common Subsequences

"How to both provide meaningful feedback on (partial) solutions to Parsons problems and discourage undesired problem solving behavior?" is an important question to figure out. The paper "A Mobile Learning Application for Parsons Problems with Automatic Feedback" talks about this issue and discusses about how to improve the Parson's

feedback. The paper presents a feedback algorithm implemented based on longest common subsequence (LCS). To give feedback about the order of the lines, searching for a longest common subsequence shared by the model and the student solution that has the greatest number of consecutive code fragments in the student's solution. The problem can be reduced to finding the longest increasing subsequences[5].

4. Design and Implementation

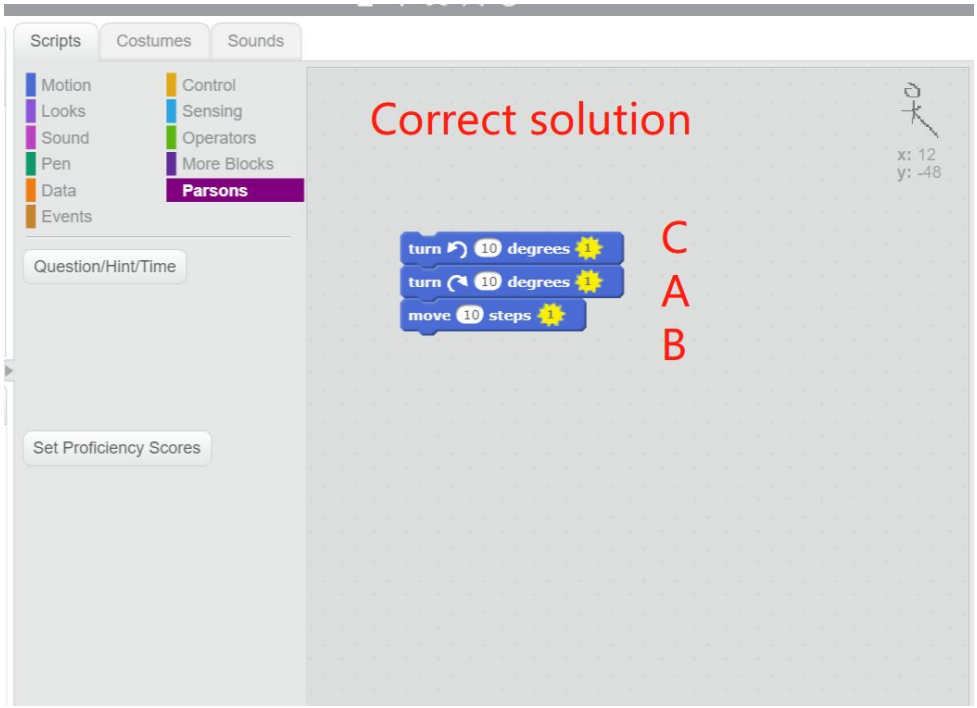
4.1 Old Design

The original scoring system was mainly based on Manhattan distance idea. For each operation in student's answer, the system tried to align it with its position in the solution by padding Null value in front of the student's answer. Then the system had a function which utilizes the Manhattan distance idea while taking the above processed student's answer as an input parameter. The system then multiplied the manhattan distance of each operation to its correct position in the correct sequence by points of the operation and added them together. Finally, the system used the total sum of all operation points in the correct sequence times the length of the correct sequence to minus the above lowest calculated sum to get the highest final result as the score. As shown above, the logic behind the original scoring system is pretty complex and we decide to design an algorithm which is more concise, easier to understand and performs better in real scenarios.

4.2 Our New Design

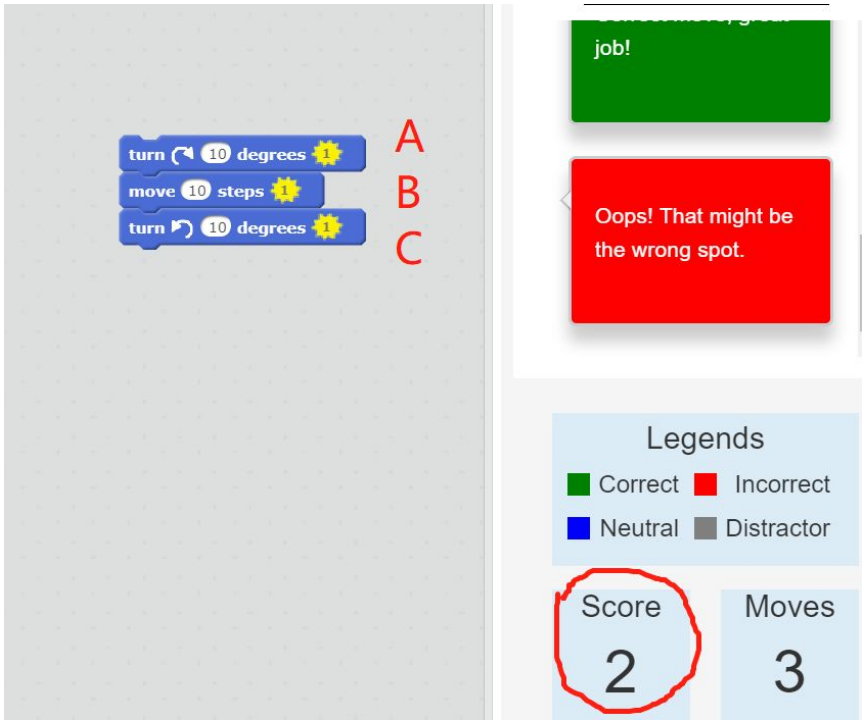
Our new design could be simplified into a simple idea: the closer the student is to solving the puzzle, the higher the score will be. The calculation mainly operates on consecutive matching subsequences (in a student response with blocks "ABC" to a puzzle with solution "CAB", a consecutive matching subsequence is "AB"). We identify each consecutive matching subsequence from the student's answer one by one from top to bottom. We score each consecutive matching subsequence by using the point

value of each block inside the student's answer times the length of the subsequence, and then sum the matching consecutive components of the subsequence while also deducting the distance between the first component of the subsequence in the student's answer and the first component of the subsequence in the real solution to take account into the error of the absolute position. The sum of the scores of all identified subsequences (with deduction already taken) is the total score.

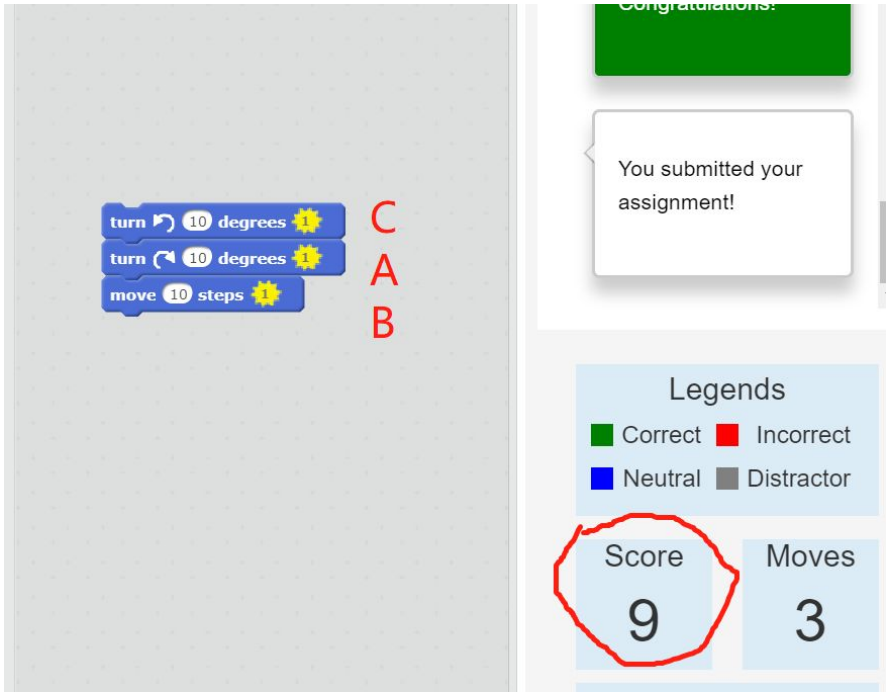


A more detailed explanation would be: For the first 2 operations in the students' answer, "AB", has a match in the solution's 2nd to 3rd operations. Assume each operation has pointValue 1. We calculate the score of this consecutive matching subsequence by using each operation's point value times the total length of this subsequence and summing them altogether. So we have 1*2+1*2 = 4. And because the absolute position differs by 1, we have to do 4-1 = 3. And next we have the 3rd operation in the student's answer, "C", which matches the first "C" in the correct solution.So we have 1*1 = 1 for

this consecutive matching subsequence. And because the absolute position differs by 2,

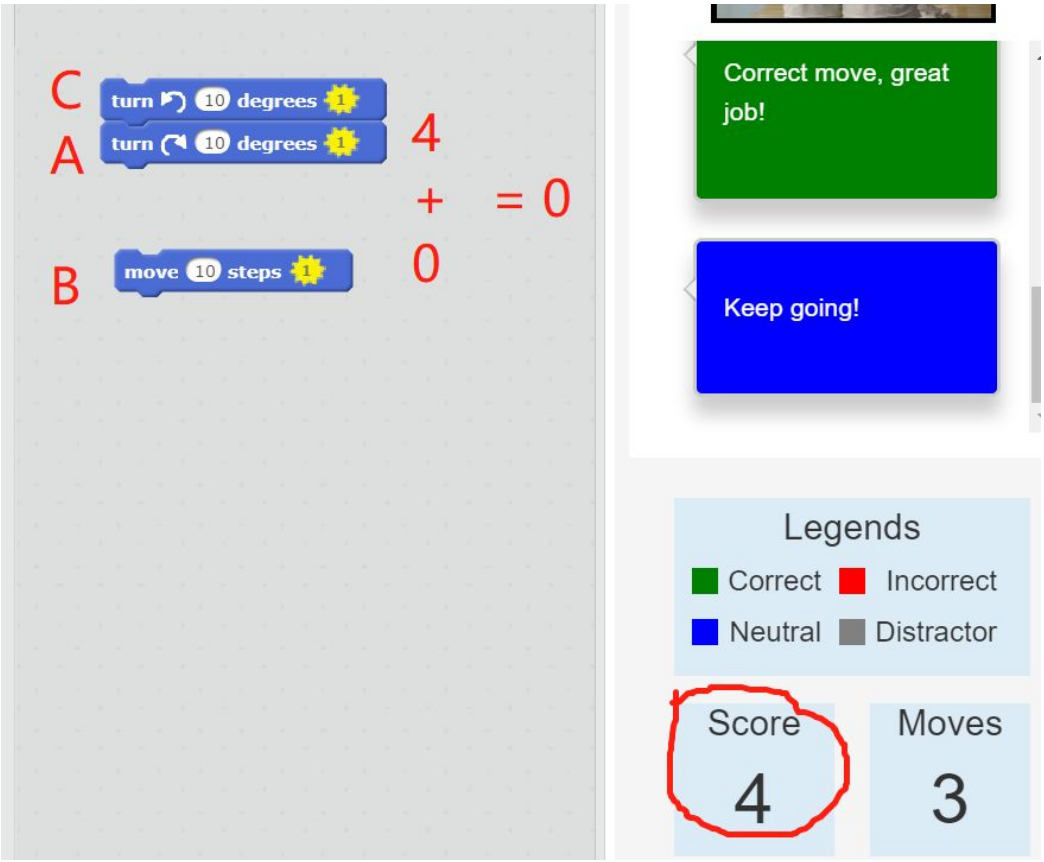


we have to do 1-2 = -1. Therefore, the final total score would be 3+(-1)=2. If the correct solution is still “CAB” and the student gets it right with answer “CAB”, then the score would be 1*3+1*3+1*3 = 9, which means the maximum possible score is 9.



4.3 Separate Chunks

We found in actual scenarios that students would very likely put some blocks on the panel first and then put several others as well but not connect them together for a while. This is due to the fact that the student has not already thought through the whole logic and decides to choose the blocks which they think make sense and put them separately on the panel first. We call this situation as “separate chunks”. The original scoring system just picked the chunk which has the maximum score as the score to show. But we thought this not very appropriate because other chunks should also contribute to the total score as well. For our scoring system, we dealt with the “separate chunks” by summing the scores of all chunks together. However, before summing them, we made sure each chunk’s score not lower than 0. It is possible for a single chunk to get a negative score because its absolute position differs from its true absolute position too much. If a chunk’s score was lower than 0, we just set it to equal 0 because we didn’t want to decrease the total score just by putting a single component of the solution separately from the main chunk. Consider an example: given solution “CAB”, student puts “CA” and “B” separately on the panel, which form two separate chunks. “CA” would give a score of 4, from the scoring algorithm we talked about in the last subsection. However, for “B”, its score should be 1-2 = -1. The deduction is because its absolute position differs from the true absolute position by 2. In this case, we don’t want to let the total score to be 4-1 = 3, which is less than the case that there’s only “CA” on the panel. Therefore, we decided to promote all chunks with score lower than 0 to have a score of 0. Then the total score would become 4+0 = 4, which is much more reasonable than 3.



4.4 Meaningful Moves

Preventing students from abusing the scoring system is another important consideration. In practice, students may simply try different combinations of blocks to finally get the correct answers, like trial-and-error. We have to stop students from doing that since it does not help students perform computational thinking at all. Initially, we tried to combine this mechanism with the scoring algorithm, but we found it not robust to some certain scenarios. Then we decided to have a separate metric called Meaningful Moves as a negative indicator, so students will have to solve the puzzle with as few meaningful moves as possible, which could prevent trial-and-error behavior.

The move count will increment when the following three scenarios happen: 1. Drag a block from the palette to the panel; 2. Connect existing sequences together; 3. Disconnect a sequence into separate sequences. We implemented the calculation by comparing the

configuration of students' current answer with that of the previous step. Each sequence from last step is saved to a map and then we can check if the composition of sequences remain the same or not. If the configuration changes, the count move will increment. If students drag blocks back to palette or only move existing sequences around on the panel without connecting or disconnect, the move count will not increase.

Besides trial-and-error, other considerations include distractors and efficiency could be solved by Meaningful Moves. Students who avoid extra moves by avoiding dragging distractors to the panel might better understand the CT concept; they might also exhibit more efficient learning of the CT concept.

4.5 Paron's Score Persistence

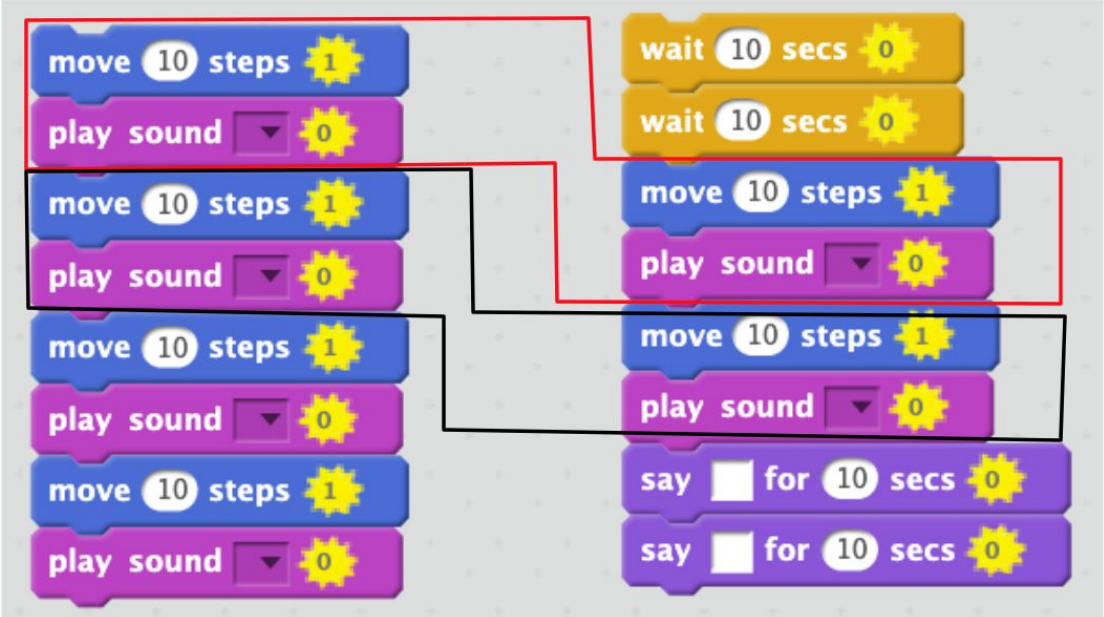
Since the Parson's score has already been persisted in MongoDB collection 'studentsubmissions', we performed several tests of it with the new scoring system to make sure that every submission could be saved to the collection with new scores and relative information, like start and end time, hint usage, etc. For future data analysis, we persisted the aforementioned Meaningful Moves and maximum scores of each game to the collection as well. Currently, each submission results in a record like the following:


```
"startTime":"1576485334698",
"score":"175",
"hintUsage":"0",
"remainingSeconds":"271",
"submitMsg":null,
"endTime":"1576485663631",
"meaningfulMoves":"5",
"maxScoreForGame":"175",
"studentID":"5db383c977376c28c4eb0614",
"assignmentID":"5db4ed770857c75ac46c588d",
"objectiveID":"5cb74d0aa8a08a1aec196d33",
"selfExplanation":null,
"sb2File":"PK\n\b!\u0085\u0081O%H\u009aB_\tÄproject.jsonÕYÝoã¸Î¦
```

5. Future work and Limitations

5.1 Duplicate Blocks Scoring

The new scoring algorithm will match duplicate blocks from top to bottom and label matched blocks in correct answers to prevent them from matching more than one time.



The order of blocks is important so instructors may need to explain the mechanism to students that the algorithm will not automatically choose the maximum score for different match possibilities but only match in order. We consider the importance of order but more discussion may need for different scenarios of duplicate sequences.

5.2 Separate Chunks

According to 4.3, we take the sum of all nonnegative scores of each chunk in order to give students as many instructions as possible. However, the instruction is only for “local correctness”, which means the increase of score could only indicate that the move in the specific chunk is considered as a good move, but it could be a bad move when the students connect chunks together. We designed the system in this way since separate chunks do not have order relations to each other. Instructors may need to inform students about the logic here.

5.3 Meaningful Moves

According to 4.4, Meaningful Moves is a metric to prevent students from abusing the scoring system. The three scenarios aforementioned are treated equally now. However, the importance of each scenario could be different so they may have different weights. For example, sometimes dragging from palette is not as important as connecting existing sequences together. Also, more scenarios could be considered beyond the three. Instructors also need to carefully consider how to use the Meaningful Moves to evaluate students.

6. Conclusion

For this semester, we mainly designed and implemented a new scoring system. Compared to the old system, the new one primarily mitigates the scoring algorithm so it is easier for instructors and students to have a better understanding. Also, the new scoring system provides students with more feedback dynamically when they work on separate chunks. Moreover, the new system adds a new measurement to prevent trial-and-error and evaluate CT for instructors.

7. 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.

[5] Karavirta, V., Helminen, J., & Ihantola, P. (2012). A mobile learning application for parsons problems with automatic feedback. Proceedings of the 12th Koli Calling International Conference on Computing Education Research - Koli Calling 12. doi: 10.1145/2401796.2401798