SAGE Project Proposal

Parson’s Programming Puzzles

Improved Scoring System

Jeffrey Fabian
Andrew Gorovoy

Abstract

The goal of the SAGE project is to help develop computational thinking among 6th - 8th graders through gameful learning. One method of completing this objective is through the use of Parson's Programming Puzzles which have been implemented within the SAGE system; however, the feature currently lacks adequate scoring baseline metrics and scoring advancement mechanisms that emphasize and reward computational thinking. For this semester, we plan to focus on the Parson's Puzzle 1.2 Feature in the Gameful Direct Instruction Epic with a specific focus on the Scoring Baseline and Scoring Advancement Stories.

Introduction

Despite its inevitable association with software engineering, the field of Computer Science (CS) has grown beyond its professional potential and into a universally applicable attitude and analytical skill set in the form of Computational Thinking (CT). That is to say, the problem-solving strategies often utilized by computer scientists — such as problem decomposition, thinking at multiple levels of abstraction and a separation of concerns — have become vital to human functioning in a world of ubiquitous computing (Wing, 2006). Alongside the conceptualization of CT, educators and technologists have become widely interested in cultivating CT in K-12 education. In particular, SAGE aims to infuse CT in grade 6-8 curricula using an intelligent game-based learning and assessment system (Bender, 2015).

A promising route to teaching CT is analyzing how students solve Parson's Programming Puzzles (PPP): a family of code construction assignments where lines of code are given, and the task is to form the solution by sorting and possibly selecting the correct lines of code (Ihantola and Karavirta, 2011). The current SAGE project provides an abstract coding block environment in which these programming puzzles can be completed. With the primary functionality of the programming puzzle implemented and working, other factors, such as scoring, will now be of focus. A strong scoring mechanism can provide a scaffold for student thinking and encourage more pragmatic CT (Garner, 2007). We propose a new scoring mechanism that is not only based on block values and correct moves, but further evaluates and supports student CT.

Related Works

An Exploration of How a Technology-Facilitated Part-Complete Solution Method Supports the Learning of Computer Programming **(Garner, 2007)**

The author of this paper identifies five distinct levels of CT that each require a progressively greater level of preplanning and visualizing of code functionality. The classifications provide logical metrics that should be searched for when trying to identify CT concepts taking place and scoring. Furthermore, the author identified that (college) students required the most support with semantics, followed by algorithms and code structure. A robust scoring system should take into account these areas of greater difficulty.

How Do Students Solve Parsons Programming Problems? — An Analysis of Interaction Traces **(Helminen et al. 2012)**

Because the process of problem solving is generally invisible to the teacher, Helminen et al. (2012) investigated the steps that students take (individually and together) throughout the course of solving simple PPPs. Their primary focus was to provide a degree of transparency to the problem solving process by analyzing programming behavior based on recorded system interactions. Results, categorized through common patterns of a system interaction graph, characterized problem solving patterns such as backtracking, in which a student undoes the operations exactly in reverse order -- implying intentional corrective behavior. Backtracking, alongside the other four identified graph patterns, could provide insight into how we can characterize a subset of interactions into a student's problem-solving intention and score accordingly.

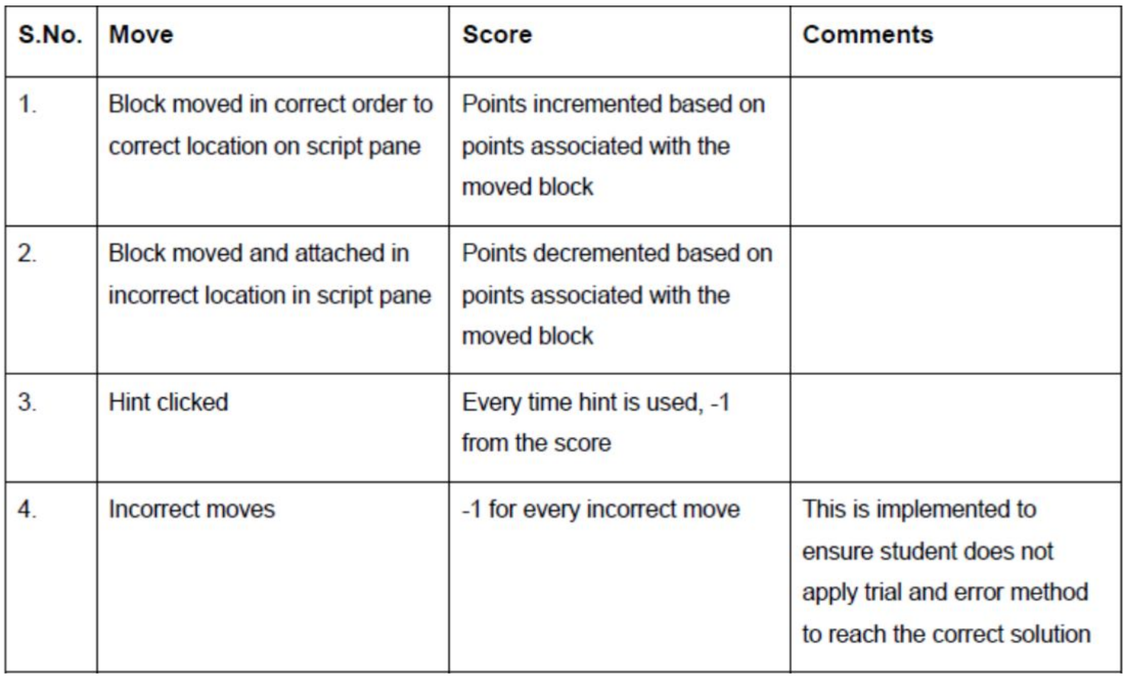
Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods

(Kiesmueller et al. 2010)

Kiesmueller et al. (2010) identified four problem-solving strategies that students (aged 12 to 13) employ in a visual programming environment: top down, bottom up, hill climbing and trial and error. Consequently, researchers built a flexible learner-system interaction (LSI) model that utilized statistical automata (HMMs) in order to probabilistically categorize interactions into the aforementioned problem-solving strategies. Accounting for expected variation of interaction sequences, their model successfully categorized 93% of LSI and noted that most learners seemed to prefer the bottom up strategy. While ambitious, a similar modeling approach for SAGE PPP could be the foundation for a scoring system that is both flexible and accurate enough to capture problem-solving strategies and higher-order CT concepts.

Proposal

The current scoring system (Table 1) only takes into account (in)correct positions, hint usage and distractor block inclusion. The current scoring methodology does not reward nor identify solutions that take into account different levels of CT. As most computer science problems can be solved with solutions of varying levels of abstraction and planning, PPPs should be as well. The current scoring mechanism is limited to assessing only one solution as inputted by the teacher. We propose adding a number of more sophisticated metrics to be detected and accounted for by the scoring system in order to *motivate* and *quantify* CT.



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: Existing scoring system

New Baseline Scoring System

In order to align CT concepts with scorable interaction, we've provided a categorization table (Table 2) that aligns CT conceptual groups with their associated PPP use case. The first, and perhaps most feasible, of these improvements would seek to identify how *deliberate* a student's approach to the problem is by imposing a limit on the number of moves. Each PPP would have an ideal number of moves for good, better, and best solutions. Moves would be counted as removing a block from a palette and dropping it into the environment, or moving a block to a new location within the environment. This new scoring methodology is intended to motivate and quantify the degree in which a student "planned" their block placements (and effectively de-incentivize a trial and error approach).

CT Concept	Description	Use Case	Explanation
Deliberate Approach	In accordance with Garner (2007), higher levels of cognitive strategy is characteristic of CT	Good, better and best solutions would be characterized by a limit to the number of moves for each rating	A limit on the number of moves would incentivize students to plan and decide on a particular approach
Abstraction	Enable users to chunk code snippets and create logical groups with the given lines of code; effectively "black-boxing" functionality	"More Blocks" section could be useful for grouping given code snippets into user-defined blocks	Being able to recognize and group sub tasks (by defining a custom block) would show an intentional understanding
Iterative Design	Think and work in small, repeatable steps. CT revolves being able to break the problem down into these manageable steps.	Special distractors that try to get students to not loop and instead explicitly code each step.	Use of loops and iterations, show more CT insight than repeating work (also exposure to DRY code)
Learning Strategy	Employing learning strategies outlined in Kiesmueller et al. (2010), would demonstrate good (or bad in the case of T&E) practice	Top down, bottom up, Hill climbing or Trial and error behavior	Student learning strategies could both inform their use of structured reasoning or lack thereof.

Table 2: Score categories aligned with CT concepts

For example, when a student begins the puzzle, he or she is presented with a max number of moves to achieve a “best solution” score. If a student surpasses that number of moves, then the maximum number of moves for the “better solution” is displayed and so forth. Yet, it must still be finalized if the current scoring mechanism with the implemented deductions and additions will remain. Having a move count and as well as score could create confusion or possibly another level of engagement. We see a similar scoring mechanism present in mobile games such as Candy Crush, and seek to harness the same level of engagement in a learning context.

Flexible Scoring Advancements

While the Deliberate Approach and Iterative Design dimensions can be implemented alongside the existing top-down-enforced scoring system, higher-order CT concepts that incorporate *patterns* of interactions (namely Abstraction and Learning Strategy) require more flexibility to allow more than one viable approach. A simple solution to the current inflexibility would be to add to the parsonsLogic algorithm to only deduct points upon recognizing block combinations rather than just correct and incorrect blocks — e.g. have the included “Control” block (if-else and loops) only be counted as correct or incorrect once its inner statement is attached.

Proposed Timeline

Week 1: Complete review of existing code base and complete detailed design document for new features.

Week 2: Begin developing and experimenting with structures and algorithms and business logic necessary for implementing the new score range system and flexible scoring advancements.
Week 3: Finalize business logic and specifications while continuing to develop infrastructure for new features.
Week 4 - 9: Maintain two week sprints developing the product following SCRUM methodologies. Intended to complete core functionality of new features during this period.
Till End of Semester: Begin unit testing and system integration. Fix bugs.

Discussion & Future Work

We sought to present the proposed improvements with respect to both feasibility and extensibility in mind. Our proposed scoring baseline includes both a feasible starting point through an imposed move count limit, but also elaborated on future conceptual groups and how their uses may manifest themselves in a PPP environment. Likewise, scoring advancement flexibility implemented with block combinations and tagging provides a clear first step before deep-diving into using interaction tracing or Hidden Markov Modeling — powerful, but timely, approaches of which are best to be considered after we confirm, or debunk, our assumptions during our PPP field studies.

References

Bender, Jeff. *Tooling Scratch*. 2015, *Tooling Scratch*.

Garner, Stuart. *An Exploration of How a Technology-Facilitated Part-Complete Solution Method Supports the Learning of Computer Programming*. vol. 4, Issues in Informing Science and Information Technology, 2007, pp. 1–5, *An Exploration of How a Technology-Facilitated Part-Complete Solution Method Supports the Learning of Computer Programming*.

Helminen, Juha. *How Do Students Solve Parsons Programming Problems? — An Analysis of Interaction Traces*. ICER, 2012, pp. 1–8, *How Do Students Solve Parsons Programming Problems? — An Analysis of Interaction Traces*.

Ihantola, Petri, and Ville Karavirta. *Two-Dimensional Parson's Puzzles: The Concept, Tools, and First Observations*. vol. 10, Journal of Information Technology Education, 2011, *Two-Dimensional Parson's Puzzles: The Concept, Tools, and First Observations*.

Kiesmueller, Ulrich. *Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods*. ITiCSE, 2010, pp. 1–5, *Online Identification of Learner Problem Solving Strategies Using Pattern Recognition Methods*.

Wing, Jeannette. *Computational Thinking*. Communications of the ACM - Self Managed Systems, 2006, *Computational Thinking*.