Gameful Direct Instruction (Parson's Puzzle)

Sudhanshu Mohan

COMS E 6901, Section 28

October 1, 2017

Table of Contents

1. Introduction

The motivations behind this project are to enhance Computational thinking and Parson's Programming puzzles with visual progress for assessment in SAGE (Bender, 2015) and improve students' understanding of computational thinking (Barr & Stephenson, 2011).

Computational thinking is an approach to solve problems using concepts of abstraction, recursion, and iteration to process. Such concepts foster critical thinking in students at an early age. Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science. In today's competitive world, a key part to be taken into consideration is how people think, understand and perceive things. Such techniques are not bounded to the four walls of the classroom, but can be applied at various places beyond that. Computational thinking inculcates engagement with the students.

In the past few years, a lot of interest has gone towards understanding how people think. Researchers have focused on design based learning activities using interactive media to keep students engaged. People have focused on honing the computer programming skills of students for introductory Computer Science courses as well as advanced courses.This has risen interest in Parson's Programming puzzles. Parson's programming puzzles are a fun and interactive way that allows teachers/tutors to touch upon important topics and syntactic and logical errors in a program. Such puzzles expose students to short pieces of code that help students construct the logical order of statements for any Parson coding puzzle. *S*ince each puzzle solution is a

complete sample of well-written code, use of the tool exposes students to good programming practices.

Parson's puzzles are built on the idea of a drag and drop concept where a student puts the block with the code piece in logical order by dragging and dropping at the correct order. A family of code is given where lines of code are provided and the goal of the students is to reach the final correct solution by sorting and possibly selecting the correct code lines. This reduces the cognitive load for the student as the code is already present. At the same time, the intent of the teacher is to help the student and guide him/her through the solution and reach the final answer. This leads to active learning rather than a passive learning.

We propose to include the concept of Parson's programming puzzles for SAGE. SAGE (Bender, 2015) extends Scratch, which is a programming language and community that allows teachers and tutors to author gameful puzzles for students. The language is drag and drop based that provides functionalities of media, stories, animation and much more for authoring puzzles (shown in Figure 1).The Scratch community is very active and has millions of users across the globe. It's a powerful learning tool that has wide audience using it. SAGE extends the idea further by proving game based-learning techniques and instills motivation for computational thinking at an early age. SAGE takes advantage of the collaborative nature of the online Scratch community and traditional classroom environments to encourage students to compete with each other while playing the games and, most importantly, learning important concepts. Using, SAGE the teacher drives the learning for the student.

2. Related Work

2.1 Hot Potatoes

The original Parson's problems (Parsons & Haden, 2006) were created using a generic drag-and- drop exercise framework called Hot Potatoes**.** Exercises created with this tool can be exported to HTML and JavaScript pages. Exercises are solved by dragging lines from right to left (shown in Figure 4). When feedback is requested, lines in (absolutely) correct positions are highlighted. One problem of this UI is that inserting a line between two existing lines is cumbersome. Student may need to move all lines after the insertion point to create a free slot where the new line can be inserted.

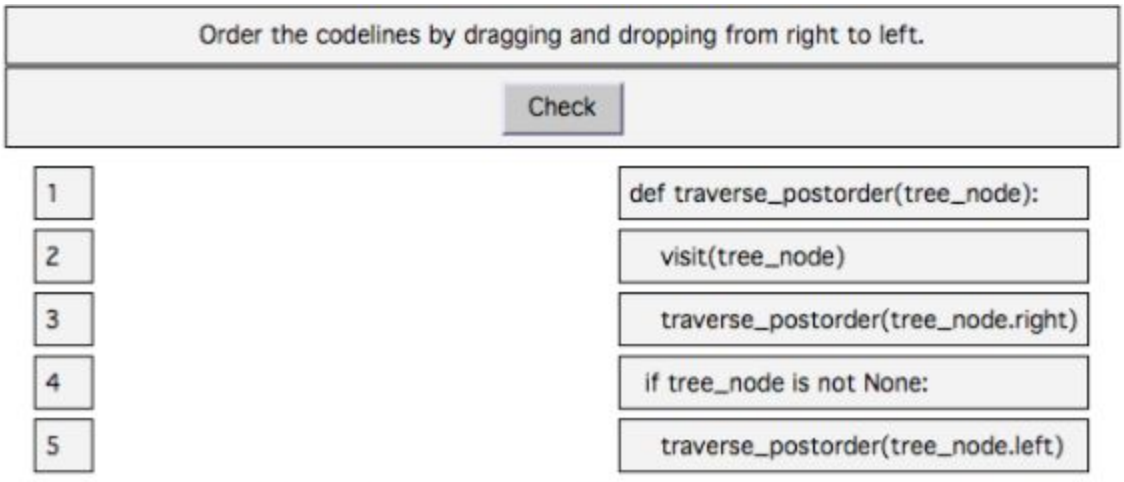

Figure 4: A Parson's problem in Hot Potatoes

2.2 ViLLE

ViLLE (Rajala, Laakso, Kaila, & Salakoski, 2007) is a Java application/applet which allow context to be created around the editable code. Distractors are not supported. The feedback is an error message in case the resulting code does not compile or a number of points in case it

does. In this case, the student can also step through a visualization of the execution of his/her solution. An example of the interface is shown in Figure 5.

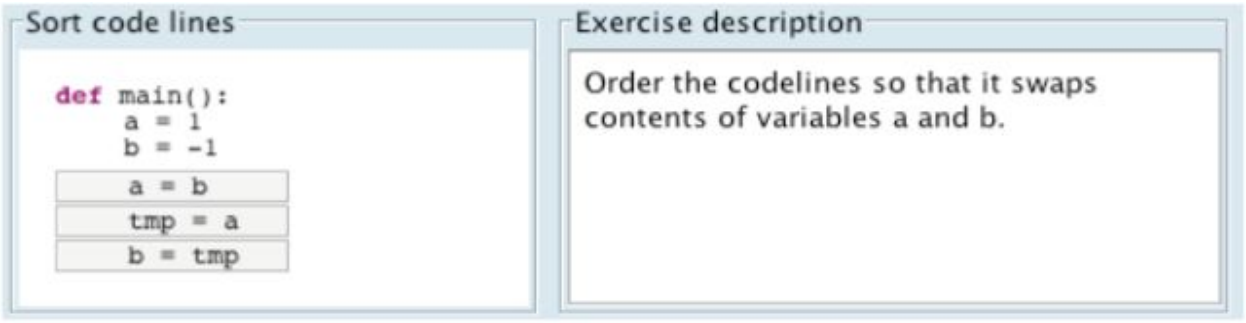

Figure 5: ViLLE

2.3 CORT

CORT (Garner, 2007) has been used with Visual Basic programs so that students move lines from left to a part-complete solution on the right. Moving the lines is done by selecting a line and clicking arrow buttons to move it left or right. To get feedback, student can copy the code into Visual Basic interpreter and execute the code. CORT supports both distractors and context.

2.4 Dr. Scratch

Dr. Scratch (Moreno-Leon, Robles, & Roman-Gonzalez, 2015) is a web application where Scratch project files can be uploaded for automated analysis. It serves as an analytical tool that evaluates Scratch projects in a variety of computational areas. After a project file is analyzed, a scorecard is presented that indicates how well the project uses a variety of computational thinking concepts. An example of this score card is illustrated in Figure 6. Dr.

Scratch helps educators in the assessment of student projects and encourages students to improve their programming skills in a fun way. Feedback from Dr. Scratch enables students to understand that successful completion of an assignment includes more than just completing a set of tasks. Successful completion also includes mastering computational thinking concepts that improve their ability to complete similar assignments in the future even if they are not in the same domain.

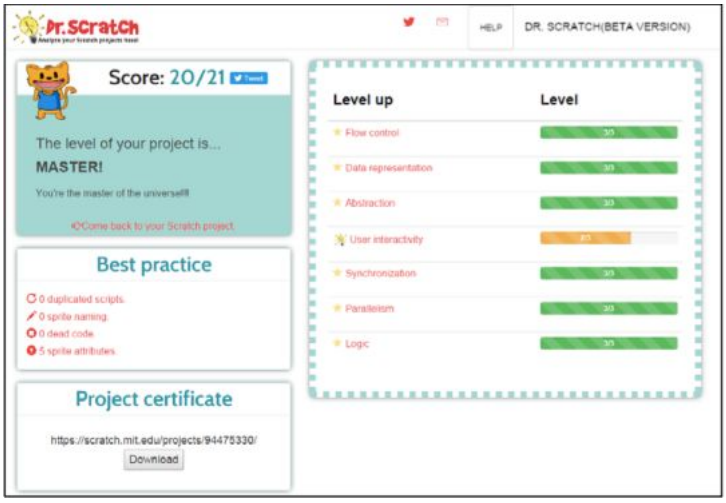

Figure 6: Dr.Scratch scorecard

The proposed project can benefit from Dr. Scratch by taking inspiration from the approach of using a scorecard to present students with visual feedback on the progress they are making on the assignment. The scorecard is concise and clear. Students immediately see where they have mastered a concept and where they need to improve. Badges and scores in Dr. Scratch helps keep students engaged because it introduces an added incentive of social competitiveness among classmates. And the progress bars allow students to quickly understand their progress towards completing their assignment the way it was really meant to be done by their teacher.

3. Proposal

This section will discuss the functionalities to be added for the teacher/tutor to design and author Parson's programming puzzle for students.

3.1 Integrated Parson's Puzzles

The teacher will be facilitated to create Parson's puzzle and Constructionist video games so that students can take guidance from the direct instruction and learn the concept. Basically when a teacher opens a Parson's puzzle in Design mode, the puzzle should appear in the script pane for the teacher to make modifications. But when opened in Play mode, the puzzle should not appear in the script pane. The stage content and Constructionist video game will appear in any mode.

3.2 Student, Assessment, & Video IDs via Query String

Presently, when Scratch launches it opens up a modal section asking the 'Student ID' and 'Assignment ID'. A new functionality is proposed where Scratch should identify the user as well as any assessment and video associated with the user. The student should not be asked for his ID. It could be implemented by passing the student, assessment and video ID information via Query String. Once implemented, the modal window asking the ID will be eliminated.

3.3 Parson's Assessment Scores

Once the Parson's puzzle is created or edited, SAGE should be able to calculate the maximum score for the puzzle and also prompt the teacher to enter two lower scores which will act as cut-offs. These cut-offs scores will be then used by the teacher to differentiate

performances between students by comparing their scores to the maximum scores. This information will help the system assign appropriately difficult in-Quest Constructionist Games.

3.4 Parson's HUD

A new Parson's Heads-Up Display (HUD) will be designed which will display the student's score in relation to the two cut-offs score (mentioned in section 3.3) and the maximum score. This will help student to gauge his progress during the play.

3.5 Parson's Self-Explanations

The students should be asked to explain their approach to solve a particular puzzle prior to the submission of the assignment. This will give an opportunity to students to reinforce learning by self-explanation. In the current implementation, a comment box appears before the submission of the assignment. This box could be repurposed to implement the self-explanation feature.

3.6 Parson's Submissions

Upon submission of the assignment, a bunch of submission messages are displayed. The teacher should be given the ability to author these messages to provide an improved student experience so that they are motivated to play more and thus stress on the 'Addictive' component of SAGE.

4. Timeline

Table 1 outlines the various milestones and their estimated completion dates. Dates are also defined for midterm and final project deliverables.

S.No	Milestone	Date
1.	Project Proposal	October 1, 2017
2	SAGE Scratch Setup	October 12, 2017
3.	Integrated Parson's Puzzles	October 26, 2017
4.	Student, Assessment, & Video IDs via Query String	November 9, 2017
5.	Parson's Assessment Scores	November 9, 2017
6.	Parson's HUD	November 23, 2017
7.	Parson's Self-Explanations	November 23, 2017
8.	Parson's Submissions	December 7, 2017
9.	Final Presentation	December 15, 2017 (Tentative)

Table 1: Proposed Milestones completion dates

5. Future Work

The future work outlined here is possible and will be worked upon contingent on the successful completion of the proposals ahead of schedule stated in the previous sections.

5.1 Parson's Scoring

The scoring scheme presently implemented in Parson’s puzzle need to be revisited. The incorrect move presently penalizes one point. In future, the penalty of the incorrect move should depend on the block point value. The scoring system should support multiple solutions to the puzzle and should be able to give more points to the elegant solution. Also, there should be a way to differentiate use of distractors from the use of solution blocks placed in incorrect positions.

5.2 Parson's Score Persistence

The scoring system mentioned in the previous section should be stored so that they can be evaluated. The score should be persisted and associated with the student and puzzle. Puzzle start and end times along with the usage of hints could also be persisted.

5.3 Parson's Coaching System

The information and metrics stored during the gameplay should be used to design an efficient Coaching system. The teacher should be able to author messages that would be presented to students during the puzzle. SAGE should provide preset configurations of coaching messages and an option to choose a default from them. Instructor should also have the freedom to modify the messages loaded via a preset configuration. In case a student is struggling a lot while solving a puzzle, a message to watch the video should pop up.

6. References

Bender, J. (2015). *Developing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula.*

(2016, January 25). Retrieved from Scratch - Imagine, Program, Share: https://scratch.mit.edu/

(2016, September 29). Retrived from Dr.Scratch - Analyse your Scratch projects here: http://drscratch.programamos.es

Parsons, D. and Haden, P. (2006). *Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses*. In Proc. Eighth Australasian Computing Education Conference (ACE2006), Hobart, Australia. CRPIT, 52. Tolhurst, D. and Mann, S., Eds., ACS. 157-163.

J. Helminen, P. Ihantola, V. Karavirta, and L. Malmi. *How do students solve parsons programming problems?: An analysis of interaction traces*. In Proceedings of the Ninth Annual International Conference on International Computing Education Research, ICER '12, pages 119–126, New York, NY, USA, 2012. ACM.

L. Seiter and B. Foreman. *Modeling the learning progressions of computational thinking of primary grade students*. In Proc. Ninth annual Int'l. ACM Conf. on Computing Educ. Research - ICER '13, page 59, New York, New York, USA, Aug. 2013. ACM Press.

Denny, P., Luxton-Reilly, A. and Simon, B. (2008). Evaluating a new exam question: Parsons problems. *Fourth International Workshop on Computing Education Research(ICER '08)*, Sydney, Australia, 113-124

Barr, V., & Stephenson, C. (2011, March). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM InRoads*, pp. 48-54.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired Static Analysis of Scratch Projects. *SIGCSE.* Denver: ACM.

Brennan, K., Balch, C., & Chung, M. (2014). *Creative Computing.* Harvard Graduate School of Education.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagen, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. A Multi-Institutional, Multi-National Study of Assessment of Programming Skills of First-year CS Students. *SIGCSE Bulletin*, 33(4).125-140, 2001.

Moreno-Leon, J., Robles, G., & Roman-Gonzalez, M. (2015, September). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *Revista de Educación a Distancia*.

Murphy, C., Kaiser, G., Loveland, K., & Hasan, S. (2009). Retina: Helping Students and Instructors Based on Observed Programming Activities. *SIGCSE* (pp. 178-182). Chattanooga: ACM.