Project Proposal

Parson's Puzzles, Parson's Coaching, and Objective Feedback

Eleanor Murguia
Zoë Gordin
Cherie Chu
Veronica Tassew Woldehanna

Contents

Abstract

This paper discusses features under Gameful Direct Instruction and Gameful Constructionism epics. We will be discussing the Parson's Coaching and Parson's Puzzle 1.2 features under the Gameful Direction Instruction epic and the feature Objective Feedback under the Gameful Constructionism epic.

Introduction

Social Addictive Gameful Engineering (SAGE) is a platform that builds upon the Scratch programming language to encourage and enhance the computational thinking skills of students between grades 6 and 8.

SAGE also incorporates Parson's programming puzzles in Scratch. Parson's puzzles is a game that allow students to reorder blocks of code to get the desired solution (Parsons & Haden, 2006). It prevents syntactical issues from blocking a student from learning concepts. In Sage, it's been modified to constrain the logic of a potential solution so students are guided to the solution as well as to make it engaging for students to learn. It also allows instructors to author the puzzle, has "Self-Explanations" for students that help reinforce learning, animates sprites to indicate success or failure, and allows the teacher to differentiate between student performances and assess mastery of a concept.

In this paper, we will explore the topics of progress logging and performance feedback, and propose methods of implementing these features in SAGE. We will also be focusing on operationalizing parson's puzzle for field study.

Related Work

Micro-adaptivity: protecting immersion in didactically adaptive digital education games (Kickmeier-Rust & Albert, 2009)

This paper discusses the importance of tailoring gaming experience to the individual student. The authors propose the idea of micro-adaptivity, a version of adaptive learning approaches specialized for digital educational games, or DEGs. The goal of micro-adaptivity is to guide the student's learning process, by doing things like alerting students of their mistakes or providing relevant hints. The article outlines different types of interventions

and how when to use them. This discussion will be helpful for Parson's Hint and Feedback Presentation.

Lessons Learned and Best Practices of Stealth Assessment (Wang, Shute, & Moore 2015)

This paper discusses best practices for and challenges with stealth, or implicit, assessment. The conceit of stealth assessment is that many traditional forms of assessment, like standardized exams, "often measure superficial skills and are stripped of the context in which knowledge and skills are applied" (Wang, Shute, & Moore 1). A common solution to the issue of standardized exams is to use games as assessments, rather than more common exams. Wang, Shute, and Moore bring to light a few essential elements of stealth assessment. These include choosing external measures that align with the stealth assessment, using performance-based assessment over self-reporting, and ensuring that the external measures of success align with the in-game measures of success. This study examined stealth assessment within games that they did not create themselves, such as *Portal 2* and *Plants vs. Zombies* — developing stealth assessment in SAGE will benefit from this team's ability to have control over the game's interface itself, as well as the fact that individual educators can create whatever challenges they wish for the game. Overall, the principles of stealth assessment will be essential in effectively developing a method to assess the students' PACT score based on the objectives they complete.

Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking (Moreno-León, Robles, Román-González, 2015)

This paper discusses the impact of Dr. Scratch, a Scratch assessment feedback device, on the computational thinking and coding skills of students ages 10 to 14. Dr. Scratch, a free open source web tool, analyzes computational thinking of users by assessing programs they create on Scratch. In its Scratch program analysis, Dr. Scratch detects errors or bad programming practices (for example, code repetition or incorrect object initialization), as well as evaluates seven facets of computational thinking (abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of flow control, user interactivity, and data representation). Experiments show that this feedback is most effective in improving the computational thinking ability of students starting off with a lower CT level, as well as secondary students. Nevertheless, both educators and learners can view their own performance and improve their programming based on the feedback given.

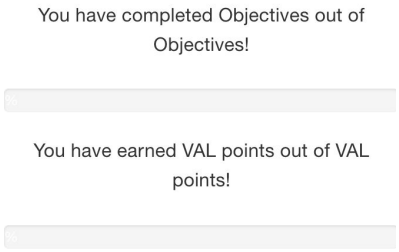
Proposal

Progress Bar

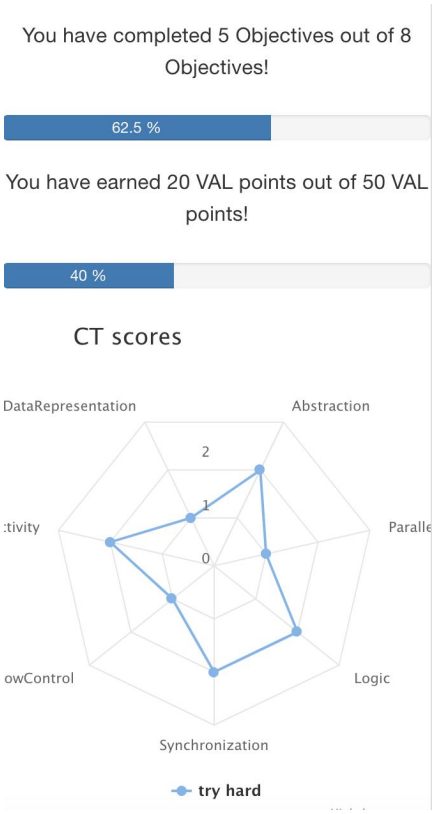
In the current state of quests in SAGE, students cannot be completely sure how close they are to completing a game. Additionally, it is not completely evident how many objectives are within a singular game. It is difficult to understand how much progress you have made when there is no clear graphical representation of that progress, so we plan to implement a progress bar for students that will show, based on how many objectives they have completed, how close they are to completing that particular game. This progress bar can also be beneficial to educators, as they will be able to understand, based on the progress the student has made, what elements of computational thinking they understand. Through this, it will be essential to be able to align the objectives of each game with aspects of computational thinking. Additionally, it will be important to be able to understand how students' actions display computational thinking — hopefully being able to display for the educators if the student is showing their understanding, or simply guessing multiple answers until they find the correct one.

Objective Points Display

For most of the quests available, the objectives completed and VAL points are not evaluated/visible:



For the few games that do display these values, very little information can be learned from the bars:



Separate goals should be visible and accessible for each game so that players can check up on past performance, gauge difficulty, and focus on the areas that need work. Breaking down these numerical values into more bite-sized goals will not only help players track their progress, but also highlight patterns in their play to bring awareness to possible improvement methods.

Parson's Coaching

Currently, most of the key features of Parson's Coaching are implemented, specifically hints, feedback, and objectives. However, the current implementation relies on modal popups that disrupt the flow of game play (see Figure 1).

Figure 1: Current implementation of hints as modal popup in Parson's Puzzles

We will focus on removing these coaching features from modal popups in order to better support students' learning. We will be moving these features to be presented by the Avatar (see Figure 2).

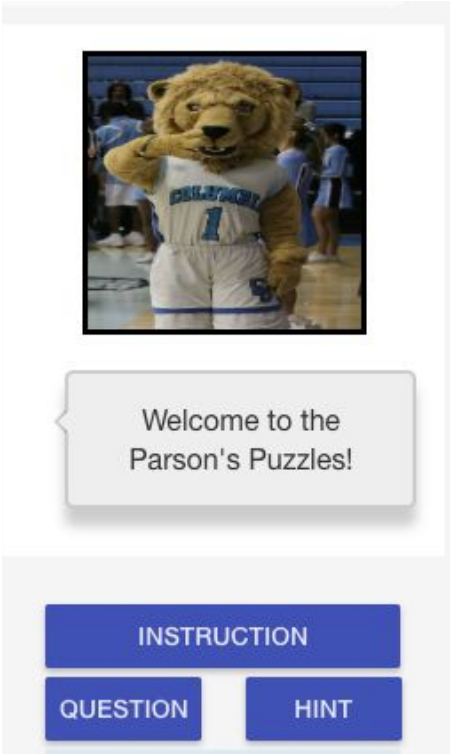


Figure 2: Current implementation of the Avatar.

Specifically, we will be moving feedback, objectives, hints, and submit to be presented by the Avatar. By doing so, we will be centralizing all coaching the students receive and allowing students to view this coaching without being navigated away from the main Parson's Puzzles page.

Operationalizing Parson's Puzzles for field study

This section will focus on improving features that are already implemented as well as increasing the efficiency of parson's puzzle game so that both instructors and students can have a better user experience.

Currently, when a student re-opens a game they have already started in the same login session, it takes a while for the scratch game to load. However, if the student refreshes the page the game loads immediately but also loses their progress. This is either a problem when saving/retrieving student information or faulty communication between sage-scratch end sage-frontend. Our plan is to remove this lag as well as properly retrieve and display student progress.

In addition, parson's puzzle at its current state shows a puzzle's solution to students whenever they open the game. Since this has been a recurring problem, we plan on fixing this bug and implementing measures that will help prevent this bug from easily reappearing in the future as other researchers contribute to the code base.

Another task is to make palette and block restriction less complex. We want the teacher to be able to add/remove a block from parson's palette without having to worry other palettes' or blocks' accessibility to students. We plan doing this by removing the option to add or remove a palette in design mode and adding the ability to remove a block from parson's palette easily from parson's palette itself without that action causing problems for that particular block in the future.

Finally, we plan on doing more testing to make sure that playing parson's puzzle is operational and easy, and making improvements and modifications as necessary. So far testing has been done mostly using the student1 and instructor1 accounts. Currently, some problems arise when loading the game if two students are playing a game at the same time. Our plan is to find and fix these kinds of problems by testing different accounts and different scenarios.

Improving on features from last semester

This section is to be done provided we have completed all that is necessary to make parson's puzzle operational during field study and we have remaining time left.

Operationalizing peer feedback in parson's puzzle: peer feedback was a feature implemented last semester in an effort to integrate student collaboration and detailed feedback in to parson's puzzle. If a student manges to correct their mistake in one try, a student is able to provide an explanation, for a point back, that will be available to other students if and when they make that same mistake. However, there are currently a few limitations to this feature. More information can be found in fall 2018's final_sage_gdi_coaching.

One limitation is such that these available explanations from other students all become available immediately once a student makes a mistake, provided their mistake has an explanation from other students. However, this might not always be desirable since not every student explanation is helpful. Our plan is to find an effective way to filter student solutions and display only a few relevant explanations to the student. In addition, if this is a mistake that a lot of students seem to be making, we can provide that data to teachers and give them them the ability to add their own explanations in addition or instead of the student explanations. This will give instructors some view of students' solving process. We also plan to make it easier for instructors to enter explanations, a task that is not easy to do in this feature's current implementation.

Another limitation is that this feature depends on the assumption that students will more or less be solving the puzzle in order of the solution. This assumption also underlies the scoring system. Since we want to give students flexibility, we want to remove this assumption from how the student feedback system works. Another team is working on removing this assumption from the scoring logic.

Finally, in line with operationalizing this feature in parson's puzzle, we will make an effort to make this feature self-contained and easy to turn on/off by the instructor. This can be helpful when conducting field study given that this feature might or might not be desirable in different cases.

Operationalizing code-visualization feature: Auto-initialization and auto-execution were also features implemented last semester in an effort to help students avoid misunderstanding of how initialization works, something that is common when students learn programming in scratch because there is carrying of state between executions, and give them the ability to better visualize their code in the context of scratch. This was done by removing carrying of state between executions and running their code with every change they make. However, this feature also has some limitations. More information can be found in fall 2018's final_gdi_sage_coaching.

Currently when a student's code is executed the sprite returns to its initial position and is then manipulated by the student's code. However, this process happens extremely fast and it might not be noticeable to the student that the sprite had gone back to its initial position. One solution can be marking the initial state of the sprite by its watermark so

students can have a constant reference to it. In addition, we might also want to trace the sprite's path from this initial position to its final position better visualizing their code by tracing out how exactly it's manipulating the sprite. Ofcourse, this might clutter the sprite stage especially for complicated solutions so we would hide this until a students feels they want to see it.

Proposed Timeline

Sprint 1: Review codebase, get development environment up and running

Sprint 2: Work on Parson's Coaching and Parson's Puzzles user interface stories; Conduct research on how to implement progress bar

Sprint 3-4: Implementation of progress bar and objective points display

Sprint 5: Complete code review, squash bugs, and integrate

References

Wang, Lubin, et al. "Lessons Learned and Best Practices of Stealth Assessment." *International Journal of Gaming and Computer-Mediated Simulations*, vol. 7, no. 4, 2015, pp. 66–87., doi:10.4018/ijgcms.2015100104.

Moreno-León, Jesús, et. al. "Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking." *RED. Revista de Educación a Distancia*, núm. 46, 2015, pp. 1-23

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