COMS 6901 - Final Report

SAGE Scratch: Parson's Puzzles and Mission Management

Sanil Shah | ss4924 | Summer 2018

1.0. Introduction

The goal for this paper is to describe the work done on the project to add functionality to SAGE Scratch which enables instructors to more easily create new Parson's Programming Puzzles and to view and update existing ones. Further, it details the changes to make solving these same puzzles using SAGE Scratch easier and more user intuitive for students using the system.

The motivations behind this project are to make the creation and editing of Parson's Puzzles in the SAGE Scratch environment as intuitive as possible for instructors while simultaneously making the student experience less confusing by removing extraneous elements during while they attempt to solve Parson's Puzzles.

The project primary covers the Gameful Direct Instruction epic as part of the SAGE [1] project. It will focus primarily on the work done towards the Parson's Puzzle 1.1 feature. This paper will cover the user stories worked on in greater detail including the architecture of the various features added, the technologies used during development and the progress made towards implementing parson's puzzles creation and editing in the SAGE Scratch branch. Finally the paper covers the future work that may be done on SAGE Scratch in order to improve the functionality further.

1.1. Computational Thinking

This project motivates and fosters computational thinking through Parson's Programming Puzzles that can be created and played by students in the SAGE Scratch environment. Computational thinking is the process of solving problems using abstraction, iteration and logical reasoning [2]. It involves creating a solution using a series of logically ordered steps such as an algorithm to solve more complex problems. Parson's Programming Puzzles using the drag and drop block programming offered in Scratch enables students to develop and enhance their computational thinking skills.

Many of today's students will work in fields that involve or are heavily influenced by computing. These students must begin to work with logical problem solving and computational methods and tools in K-12. This can be accomplished by providing K-12 instructors, researchers and students with resources and relevant age-appropriate examples [3]. This is precisely the problem that SAGE Scratch along with Parson's Puzzles support aims to solve.

1.2. Parson's Programming Puzzles

The primary concept of Parson's Programming Puzzles is to provide an interactive method that allows students to practice basic programming principles in an entertaining puzzle-like format [4]. This enables instructors to impart gameful interactive learning to students without demotivating students with syntax and logical constructs that are usually required in order to learn to program. All parson's puzzles are drag and drop style, designed to maximize engagement, model good code, permit common errors and provide immediate feedback. This makes Scratch an ideal tool to build upon to support creating, editing and solving parson's puzzles because of its block based programming interface.

The work done by Gupta and Mohan in Fall 2016 [5] laid the foundation for the features that are described and completed for this project. The team built the fundamentals of Parson's puzzle creation into the SAGE scratch environment. The efforts of this project were to improve upon the work done in prior semesters to make parson's puzzle creation, updating and playing more seamless and user friends for both instructors and students.

1.3. Goals

The primary goals of this project are to make modifications to the SAGE Scratch project in order to improve the user experience and simplicity of creating, updating and playing parson's puzzles games. This will be done by ensuring that the SAGE Scratch environment is aware of the context that it was loaded from.

This involves adding detection within Scratch to determine whether the puzzles being loaded is a parson's puzzle or a constructionist game or any other kind of assignment. This allows the SAGE Scratch environment to make some intelligent design decisions and UI choices to remove any irrelevant details and focus on the important UI elements on the screen. Similarly, this project will add logic to within the SAGE Scratch environment to determine whether it is being accessed by an instructor or a student. This will once again enable the user interface to be specifically catered to the type of account that is loading it.

Once game type and account detection has been added to SAGE Scratch, this project builds on it to allow instructors to create new parson's puzzles and correctly edit and update existing ones. Further it ensures that students have a simplified view when a parson's puzzle game is opened that disables any Scratch functionality that isn't directly relevant to the game, making it easy for students to focus simply on solving the puzzle. We will explore these goals further in section 4.

2.0. Related Work

There have been several implementations of Parson's Puzzles through frameworks like Hot Potatoes, ViLLE, CORT and js-parsons, which provide a drag and drop interface similar to Scratch. The images below show an example of what a Parson's programming puzzle may look like using some of the existing frameworks.

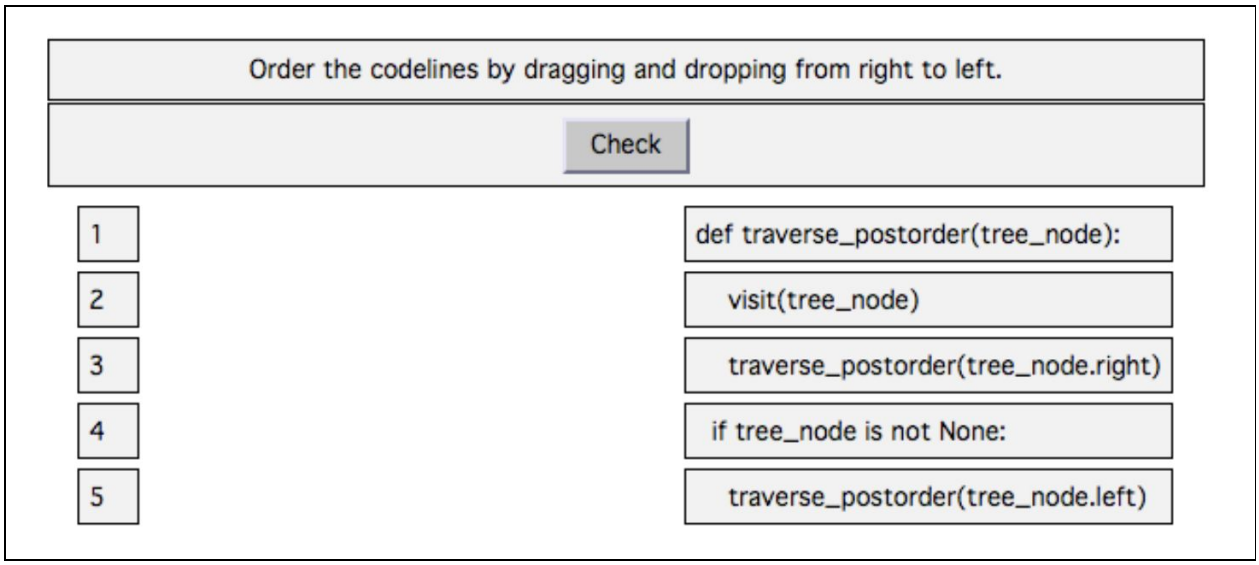

Figure 1: Parson's Puzzle for post order tree traversal using Hot Potatoes

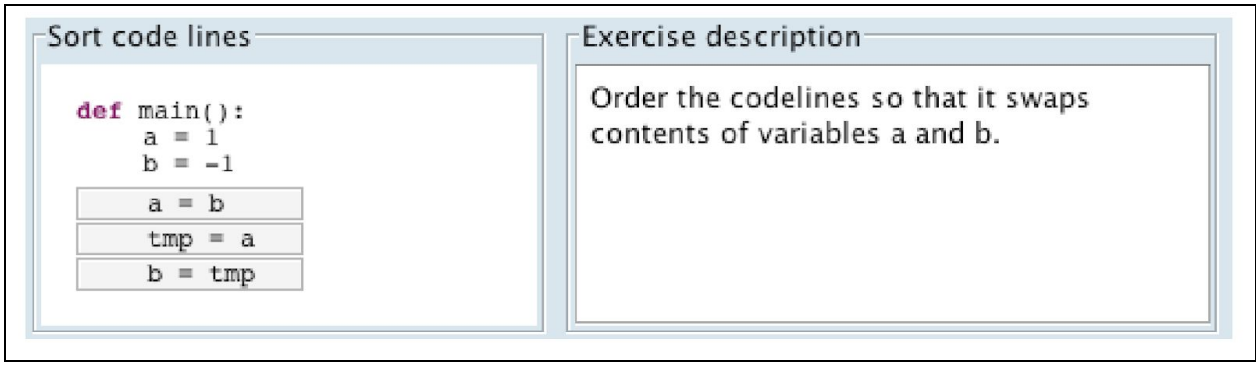

Figure 2: Parson's Puzzle for swapping variables using ViLLE

While the functionality added by this proposal will be similar to that offered by the frameworks above, the value it adds above and beyond the existing research will be integrating with the rest of the SAGE ecosystem which includes the SAGE front end, intelligent hinting system and the web application. This will provide a well integrated learning system that uses Parson's Puzzles to meet the needs of students, teachers as well as instructors.

2.1. SAGE Scratch

The most relevant prior work that this project builds on is the work done in 2016 to build the initial version of Parson's Puzzles support in SAGE Scratch by Gupta and Mohan [5]. The initial work done on this project created the Parson's palette and added checkboxes next to blocks to enable instructors to select the Scratch blocks that were relevant to the parson's puzzle being created or modified. Further it added the functionality to automatically save updates to any parson's puzzle that had already been created while it was being modified. The previous project also added a scoring system and this paper will address that further in section 6 which covers future work.

As described in section 1.3, this project enhances the above work to properly implement the creation of parson's puzzles in SAGE Scratch. Game type detection as well as account type detection is added to improve the user experience and simplicity for both instructors and students. Additional features have been added while updating parson's puzzles to ensure that checkboxes next to Scratch blocks are correctly populated based on blocks that have already been selected to be part of the parson's puzzle. Lastly, we add functionality that differentiates SAGE Scratch behavior when the environment is loaded by student to prevent students from editing parson's puzzles. The parson's palette has been updated to ensure that students are focusing on solving the puzzle created by instructors instead of being distracted by other blocks and features that would otherwise be available in the Scratch environment.

3.0. Initial State

This section describes the initial state of the SAGE Scratch project before the work described in this paper was completed. This motivates the necessity and importance of the work that has been completed during this project. It identifies existing areas of concern to establish the baseline or the starting point for this project and forms the foundation of the work described in section 4.

3.1. Parson's Puzzle Creation

Before this project started, there were some issues with creating parson's puzzles through the SAGE frontend. Figure 3 below shows the errors that were surfaced when an instructor would attempt to create a new parson's puzzle game. This made it impossible for instructors to create new parson's puzzles using the SAGE platform. The error complains that the JavaScript angular framework cannot set the innerHTML property of null which indicated that there were issues

with how the puzzle creation page was structured. This resulted in the SAGE Scratch object never loading and hencing leaving no way to actually create a parson's puzzle.

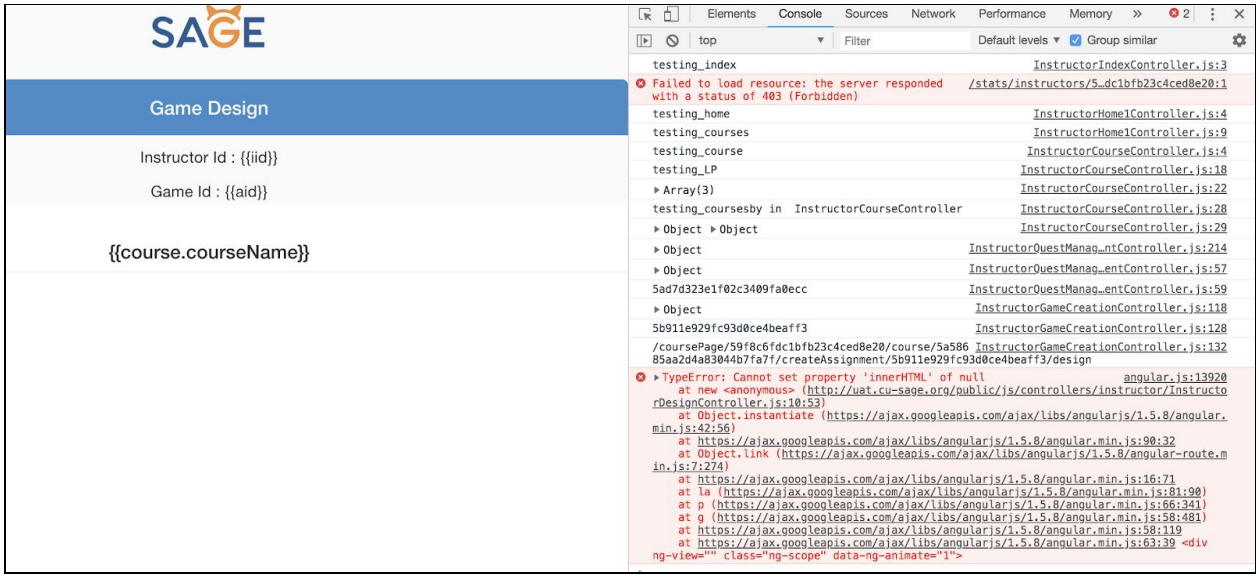

Figure 3: Parson's Puzzle creation showing an error

3.2. Parson's Puzzle Edit

The SAGE platform enabled instructors to view and update games that had already been created. SAGE Scratch automatically saved changes made in the scripts pane directly to the database each time the instructor added more blocks to a puzzle or reordered the solution. However, in the initial state, there were some issues with parson's puzzles being updated as well. As shown in figure 4, the SAGE Scratch environment would open in neither design mode, nor play mode, which left it in an undefined state. The expectation here was that when an instructor attempted to modify an existing parson's puzzle, SAGE Scratch would recognize that action and automatically open in "design mode" allowing instructors to immediately start modifying their puzzles.

The second issue while attempting to update existing parson's puzzles was that the checkboxes next to blocks in the SAGE palettes were not marked as checked even if the blocks were already in the parson's palette. This made it possible to add the same block to the parson's palette more than once and also made it impossible to completely remove an existing block from the parson's palette. This was undesirable behavior and needed to be fixed in order to enable instructors to successfully edit parson's puzzles that they had previously created.

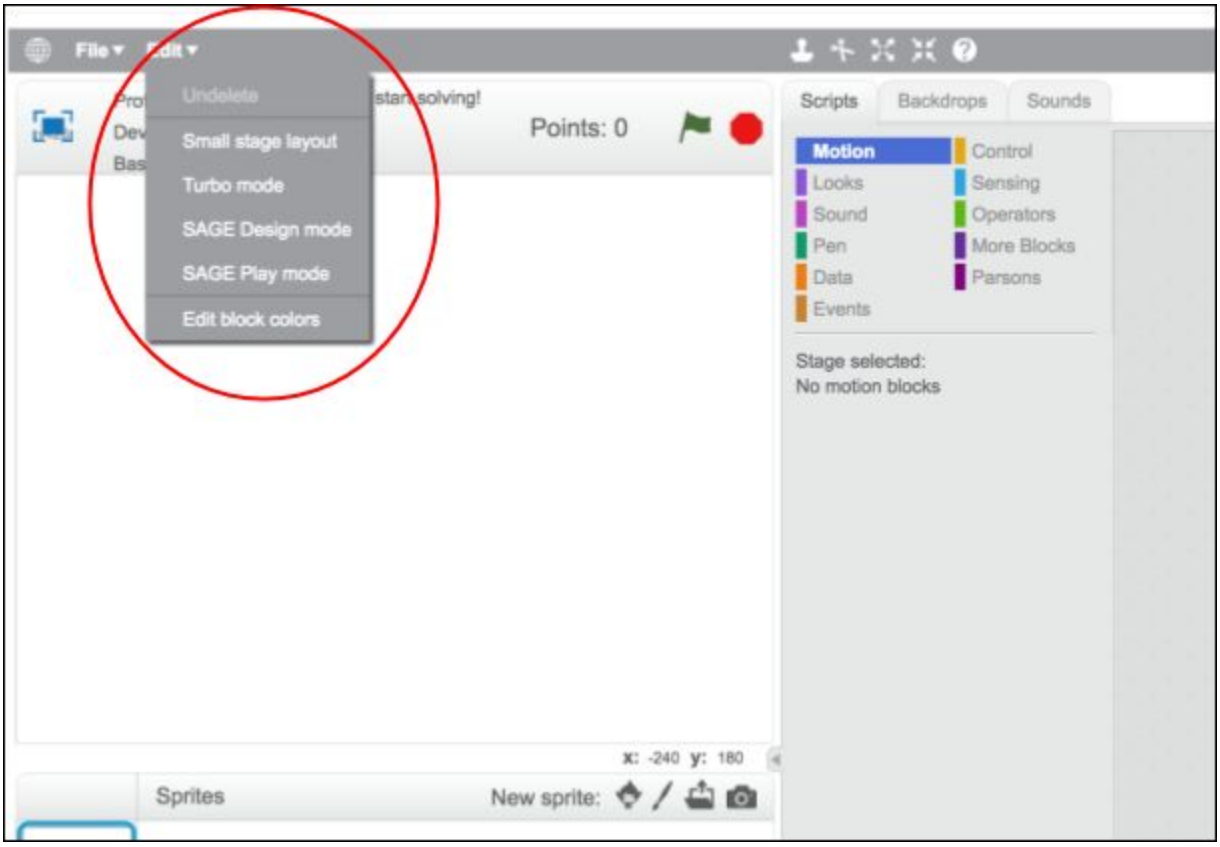

Figure 4: Updating a parson's puzzle did not open in SAGE Design mode

3.3. Parson's Puzzle Palettes

This section describes the state of SAGE Scratch as observed by students using the SAGE platform while attempting to play parson's puzzles. The initial state enabled students to open and play existing parson's puzzle and automatically opened Scratch in play mode. However, there were some issues or limitations with the implementation. Figure 5 below shows that students we also able to change into SAGE design mode from the edit menu. This was undesirable since it allows students to modify puzzles that were set up by instructors.

The right half of figure 5 also shows that students could select between any of the available palettes (Motion, Looks, Sound, Pen etc) while they should be restricted to only the Parson's palette while attempting to solve a parson's puzzle. Finally, the script pane while opening a parson's puzzle would automatically be populated with the solution that the instructor had set up while creating the puzzle. This was obviously not working as intended since it gave the students the solution before even allowing them to attempt to solve it. The work done by this project aims to resolve these issues and make the parson's puzzle solving workflow easier to understand and more intuitive for students.

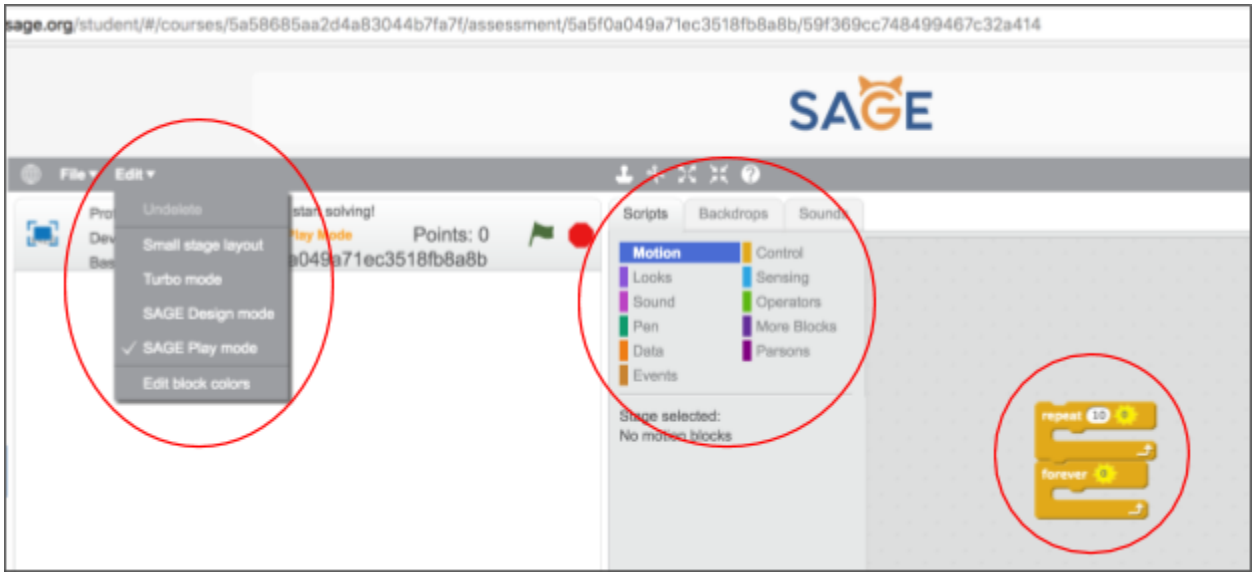

Figure 5: Playing parson's puzzle as a student showing several issues and limitations

4.0. Implementation

This section covers the features that were added to both SAGE Scratch and SAGE front-end through the process of this project. We will first discuss the design and the architecture of the changes being made including the languages and technology used. This paper will then present a detailed account of the changes made during this project along with the improvements that were implemented in the SAGE Scratch project while specifically focusing on parson's puzzle functionality.

4.1. Architecture

This project builds features on top of the existing SAGE Scratch project. This means that the architecture follows the existing design and methodology of SAGE Scratch. Query parameters are used to pass information from the SAGE Affinity Space (front end) to the swf object which allows Scratch to initialize with the correct data. The data tells Scratch about the puzzle being played or edited, the user ID currently using Scratch (instructor or student) as well as whether Scratch needs to be opened in PLAY or DESIGN mode. This data is also used to pass information about instructions to be displayed to the user and the expected solution which can be used in the future for scoring purposes.

This project is built primarily in ActionScript while using Gradle as a build system. The features added to this project are also completed in ActionScript by modifying files of the existing SAGE Scratch branch. Changes were also made to the SAGE Affinity Space code in order to correctly pass the required parameters to the Scratch environment and also to fix the errors described in

section 3.1 that were preventing instructors from creating parson's puzzles. The build process setup in previous years was not modified and most development was done locally by running local versions of the SAGE Affinity Space and the SAGE Assessment Server, as well as a local version of SAGE Scratch.

4.2. Game Type Detection

The first goal of this project was to add functionality to SAGE Scratch to be able to differentiate between different kinds of puzzles (parsons, constructionist games) in order to be able to implement separate behavior within the Scratch environment for each of them. In order to accomplish this the SAGE affinity space (or front end) needed to load the SAGE Scratch swf with the correct URL parameters to indicate the game type.

The URL parameter called "&type" was added to the SAGE Scratch loading URL. This meant that the URL looked something like this:

http://dev.cu-sage.org/public/sampleSWF/scratch.swf?sid=123&assignmentID=456&mode=DESIGN&backend=http://dev.cu-sage.org:8081&type=parsons

Note the "&type=parsons" which was added to the end of the swf loader URL to indicate that the puzzle being loaded was a parson's puzzle. These changes were made by modifying various controllers for both the instructor and the student in the SAGE affinity space such as InstructorDesignController, InstructorGameCreationController, InstructorGamesController and StudentAssessmentController.

This information was then read in the Scratch environment by modifying code in Scratch.as to look for the "type" parameter from the parameters on the loader URL. This is similar to the way the backend URL and the game mode (play or design) was determined in the affinity space and communicated to the SAGE Scratch environment. The effect of these changes can be seen in figure 6 below, where the developer tools on the right clearly shows the game type identified as parsons from within the Scratch environment when a parson's puzzle is opened to be updated by an instructor. Recognizing that the puzzle being opened is a parson's puzzle enabled further enhancements to the user interface as described in sections below.

4.3. Account Type Detection

Similar to the above goal where the type of the game was detected from within the SAGE Scratch environment, functionality was added to also detect the type of account that is loading and running the Scratch program. This is used to implement different behavior for students and

instructors which is extremely desirable since students should be exposed to only a subset of the functionality that is available to instructors. This allows the Scratch environment to hide functionality that students should not have access to while playing a game.

This feature was added by examining the URL of the page which the Scratch swf is being loaded on. If the URL (window.location) of the page contains the word "instructor" the account type is determined to be an instructor and hence the isStudent flag is marked as false within the Scratch environment. This functionality can be seen in action in figure 6 where the console log on the right side indicates that Scratch was loaded by a user who was signed in as an instructor. Alternatives considered for implementing this feature was to add yet another URL parameter as was done for the puzzle type in section 4.2, however this was deemed to be the cleaner solution since it did not require the modification of multiple controllers in the SAGE affinity space. Once again this functionality allows for an enhanced user experience for both students and instructors since behavior within the Scratch environment can be catered to the type of account accessing it.

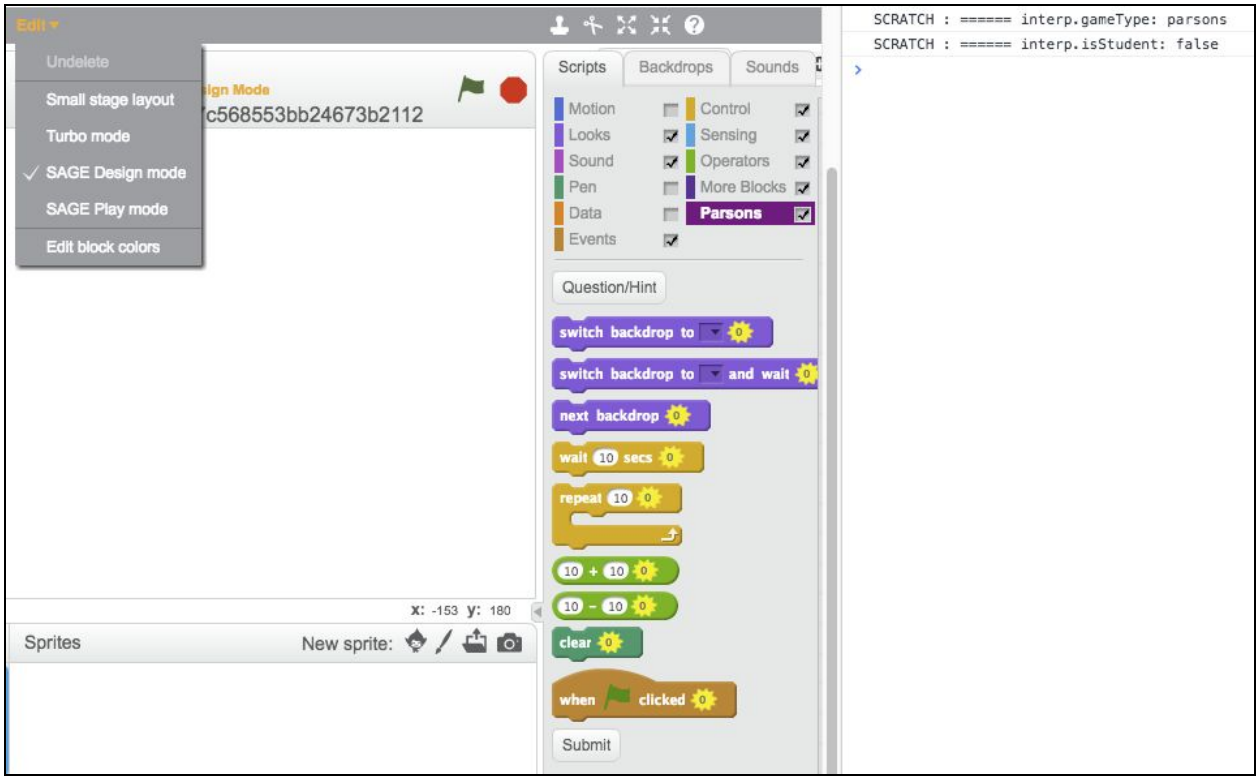

Figure 6: Developer tools showing puzzle type and account type detection in SAGE Scratch

4.4. Parson's Puzzle Creation

The most immediate problem with the initial state was that the creation of new puzzles was completely broken. This was due to issues with the SAGE affinity space and the game creation views that did not seem to work together to correctly load the SAGE Scratch swf. This was fixed by modifying the instructor_scratchdesign.html view in the SAGE front end to ensure that the HTML5 element where the controllers were attempting to place the Scratch swf actually existed.

These changes enabled instructors to create new games without experiencing any errors, and hence also allowed them to create new parson's puzzles. Figure 7 shown below demonstrates that there were no errors after the changes were made as seen on the right, and the SAGE Scratch swf is correctly open in "design mode" on the left.

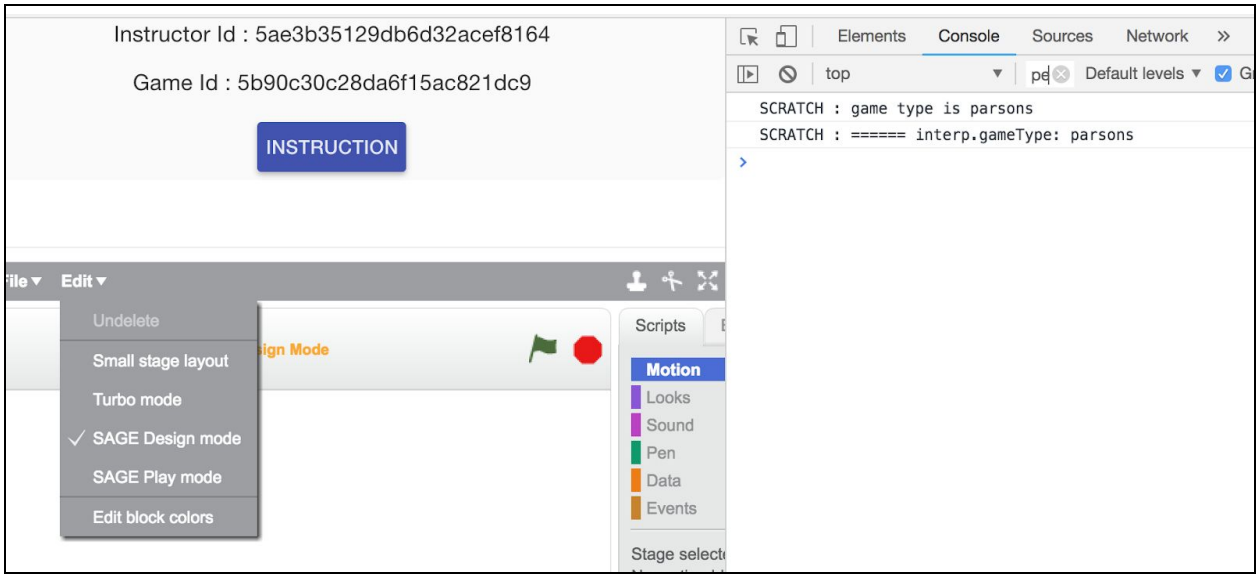

Figure 7: Parson's Puzzle creation working correctly with no errors

4.5. Parson's Puzzle Edit

The ability to update an existing parson's puzzle as an instructor had several shortcomings as described in section 3.2. The work done in this project modified the existing functionality to ensure that a possible opened to be modified from the InstructorDesignController or the InstructorGamesController always correctly opened in SAGE design mode instead of play mode or the undefined state that the initial SAGE Scratch exhibited. This was implemented by checking the "mode" URL parameter on the loader URL. If mode was set to "DESIGN" the SAGE was opened in design mode. The controllers from the SAGE affinity space made sure to add the

"mode=DESIGN" parameter to the swf loader URL when the Scratch environment was being opened in order to edit an existing game.

The effects of this feature can be seen in figure 8 below. Note that the scripts pane on the right is automatically populated with the puzzle that was already created by the instructor as is the sprite pane on the left. The appropriate checkboxes that were previously set up by the instructor are marked as checked and unchecked as well. This functionality enables instructors to immediately start modifying a parson's puzzle when it is opened in order to be updated.

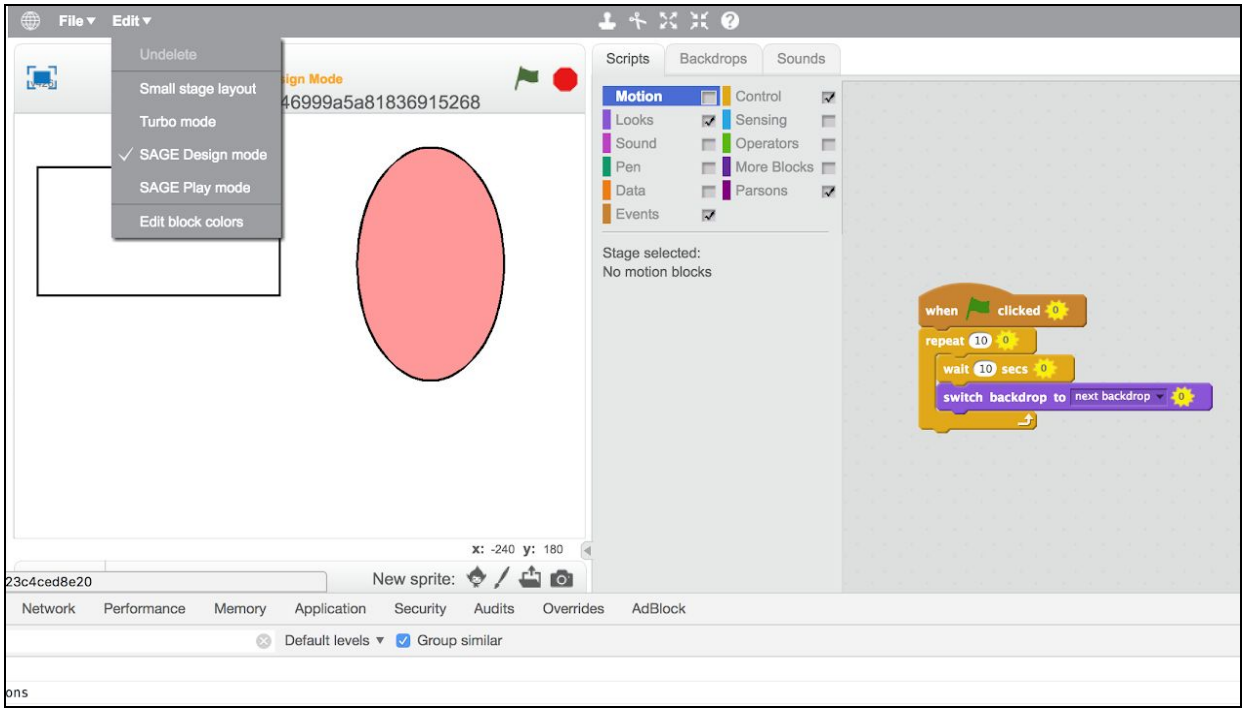

Figure 8: Parson's Puzzle editing opening in design mode

The other issues that needed to be fixed to enable seamless editing of puzzles was the ability to correctly populate the checkboxes on the blocks, based on the blocks that had been already selected for the parson's palette. This was done by modifying the PaletteBuilder code in Scratch to correctly populate the boxes when a parson's puzzle was opened. This is also where the game type detection for 4.2 proved to be especially useful since this functionality was only relevant to parson's puzzles and would not apply for constructionist games. Figure 9 below demonstrates that the blocks from the parson's palette are appropriately checked in their own individual palettes as well. For example, note that the "wait x secs" block is in the parson palette in the extreme right, and it is also correctly has it's checkbox marked in the extreme left in the Control palette as well. This enables instructors to remove existing blocks from the parson's palette and ensure that no duplicate blocks are added to the palette.

Figure 9: Parson's Puzzle editing opening in design mode

4.6. Parson's Puzzle Palette

The final portion of updates made during this project focused mostly on the student experience when opening and playing parsons puzzles. The main goal was to implement different behavior for the students that limited their ability to make changes that should be restricted to instructors. This was accomplished by first completely disabling the SAGE design mode menu option in the edit menu. The work to identify the account type from 4.3 enabled SAGE Scratch to have some context about the kind of user that was logged in and using the environment. This allowed for changes in the Scratch.as file to disable the design mode button as seen in figure 10 below. This also implied that any student account automatically opened in play mode as one would expect.

Further updates to the user experience ensured that when a student opened a game that was parson's puzzle, the palette selector initialized by selecting the parson's palette. In the initial state the first available palette was selected which would be confusing for students since they

are only expected to use the blocks available in the parson's palette to solve the puzzles created by instructors. Hence this made the user experience more intuitive and easy to use for students attempting to solve parson's puzzles.

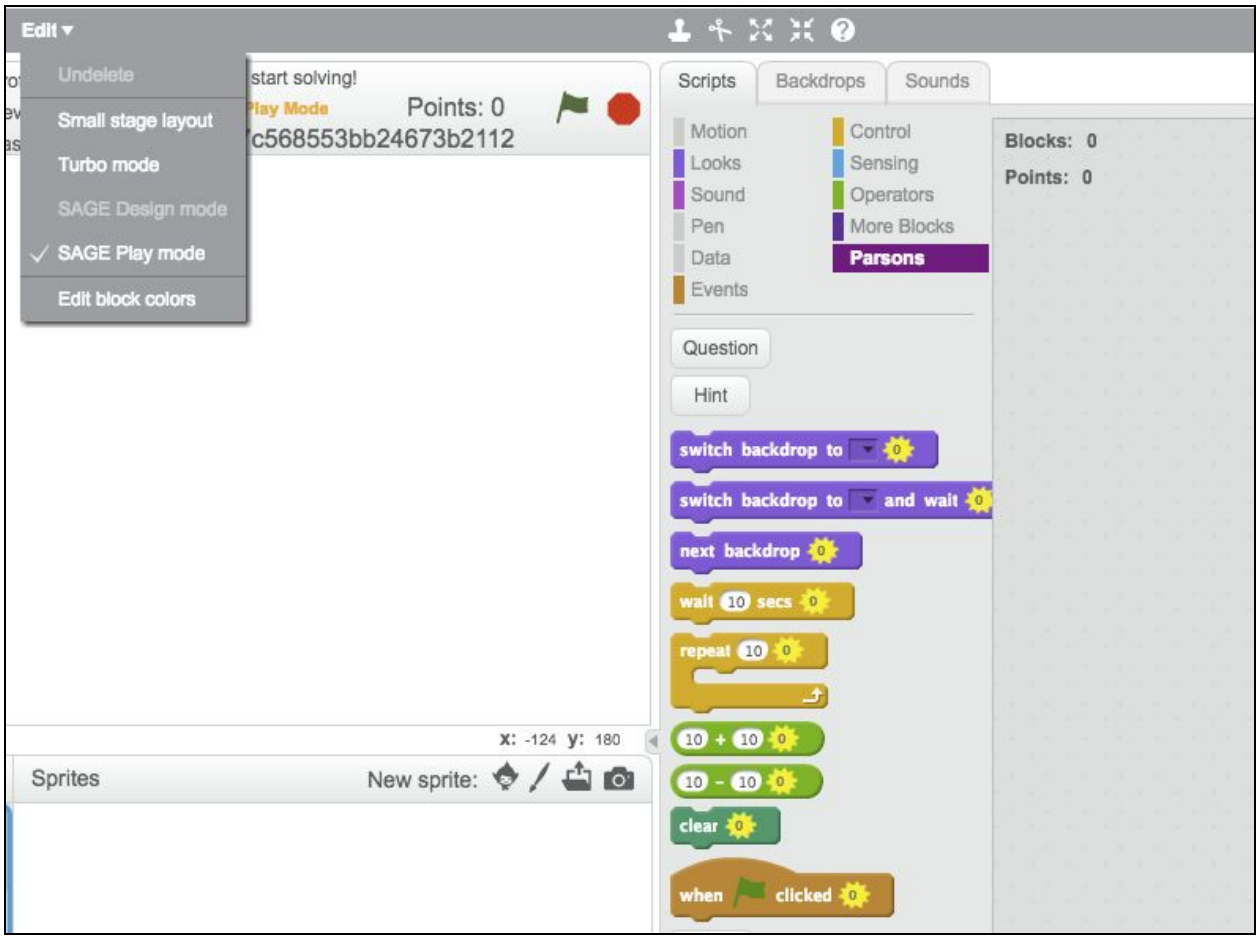

Figure 10: Playing a parson's puzzle as a student

Lastly changes were made to complete disallow students from selecting any other palette other that the parson's palette while they were trying to solve a parson's puzzle. This functionality is intentionally limited only to games that are of type "parsons" since it does not apply for constructionist games where students are allowed to create programs from any of the blocks. Figure 11 below shows that the parsons palette is selected by default when a student opens a parsons game. It also shows that other palettes are grayed out, and those are blocks that have been completely disabled by the instructor. Other palettes like Control and Operators look like they could be selected by the student however specific code was added to the PaletteSelector to prevent students from changing the palette during parson's puzzle solving. This further simplifies the user interface and forces students to focus on the task at hand instead of being distracted by various other elements of the Scratch interface.

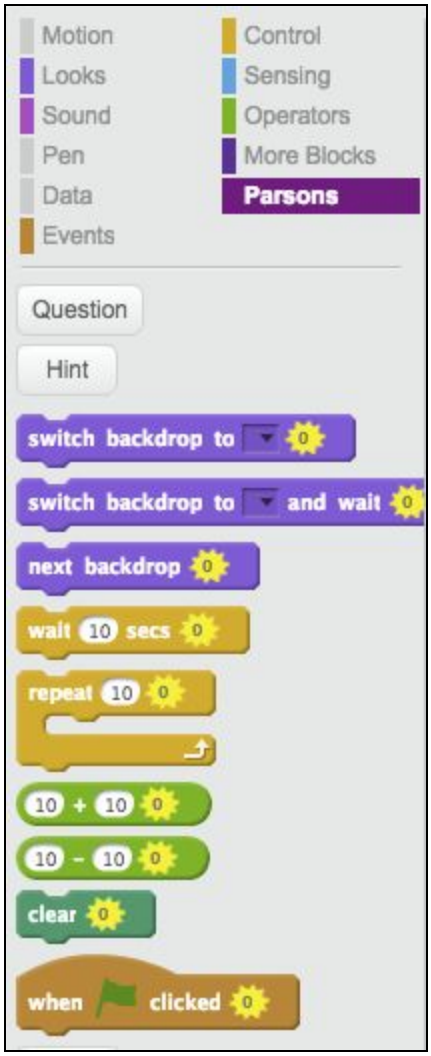

Figure 11: Parson's palette selected by default with other palettes disabled for students

5.0. Code Repository

The code for this project can be found at the links below:

- SAGE front-end instructor controllers
- SAGE front-end student controllers
- SAGE front-end views
- SAGE Scratch
- SAGE Scratch PaletteBuilder
- SAGE Scratch PaletteSelector
- Final Presentation

6.0. Future Work

There were several avenue of future work that can be explored from the work completed during this project and described in this paper. They are listed below:

- Modify and fix the scoring algorithms for parson's puzzles to give students instant feedback about the progress they are making. The current implementation has a basic framework set up for scoring that doesn't seem to give any positive scores to students which seems like incorrect functionality.
- Disable edit mode only selectively based on the game being opened and whether it was originally created by an instructor. This would enable high performing students to create and edit games as well (instead of design functionality being disabled for all students). Currently there is no functionality in the SAGE Affinity Space to allow student accounts to create games.
- Allow instructors to reorder the blocks selected in the parson's palette. This is possible today by unselecting all the blocks and re-selecting them in the desired order. However, this is extremely inconvenient and could be improved to provide instructors with an easier user experience.
- Evaluate the project along with the parson's puzzles functionality in a classroom environment with actual teachers and students to understand further pain points and usability issues.

The features described above would enhance the value of the work done during this project by rounding out and completing the parson's puzzle functionality for both instructors and students.

7.0. Conclusion

In conclusions, updates made to the SAGE Scratch implementation of parson's puzzles was presented in this report. The features added by this project enable instructors to quickly and easily create and update parson's puzzles. They also allow students to focus on the task at hand by providing a distraction free user experience which enables them to concentrate on solving the parson's puzzle. These puzzles ensure a path towards guided learning while teaching students computational thinking skills.

8.0. References

[1] Bender, J. (2015). Developing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula. https://gudangdaya.atlassian.net/wiki/download/attachments/75595790/tooling_scratch.pdf?version=2&modificationDate=1506620159936&cacheVersion=1&api=v2

[2] Grover, S., & Pea, R. (2013). Computational Thinking in K—12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43. http://www.jstor.org/stable/23360476

[3] Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? https://www.researchgate.net/profile/Valerie_Barr/publication/247924673_Bringing_computational_thinking_to_K-12_what_is_Involved_and_what_is_the_role_of_the_computer_science_education_community/links/53e2e8b40cf2b9d0d832c294.pdf

[4] Parsons, Dale & Haden, Patricia. (2006). Parson's programming puzzles: A fun and effective learning tool for first programming courses. 52. 157-163. https://www.researchgate.net/publication/262160581_Parson's_programming_puzzles_A_fun_and_effective_learning_tool_for_first_programming_courses

[5] Gupta, A., Mohan, S. (2016). Formative SAGE Assessments: Parson's Programming. https://github.com/cu-sage/Documents/blob/master/2016_2_Fall/final_report_SAGE_parsons.pdf