User Interface, Learning Metrics

SAGE Spring 2019

Cherie Chu, Zoë Gordin, Eleanor Murguia

Table of Contents

Introduction

Social Addictive Gameful Engineering (SAGE) is a research project that utilizes the Scratch programming language to encourage and enhance the computational thinking skills of students between grades 6 and 8. A key element of SAGE is the assessment of student's computational thinking, done through the Learning Metric Analysis. SAGE utilizes the Hairball-Kurt library to determine mastery of key skills by analyzing game results.

In this report we will discuss functionality improvements made to SAGE Learning Metric Analysis as well as improvements to the user interface and the app's overall functionality. We will also discuss future work in these areas and potential technical limitations.

Related Work

Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking (Moreno-León, Robles, Román-González, 2015) [1]

This paper discusses the impact of Dr. Scratch, a Scratch assessment feedback device, on the computational thinking and coding skills of middle school students. 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 (e.g. 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). Experimental data demonstrates that this feedback is most effective in improving the computational thinking ability of students starting off with a lower Computational Thinking 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.

Understanding Computational Thinking Before Programming: Developing Guidelines for the Design of Games to Learn Introductory Programming Through Game-Play (Kazimoglu et. al., 2011) [2]

This paper discusses how it can be beneficial for students to use games to learn Computational Thinking without actually writing code. In these games, there are exercises that are based in computational skills, and have clear links to actual coding exercises, but are more gameful than typical exercises that are involved in an introductory computer science course. The authors describe how "games designed to encourage players to think computationally through puzzle-solving gameplay, in an environment contextually based on CS programming concepts, are conducive to learning programming"(48). Clearly, there are ways other than learning programming languages to learn how to code or simply to learn Computational Thinking skills, and the authors of this paper explore games similar to Scratch games that accomplish this goal. The paradigm discussed in this paper is directly connected to the goal of SAGE, and shows how essential it is to accurately and clearly calculate students' Computational Thinking scores.

Implementation

User Interface Bug Fixes

Login

Two minor fixes have been made to the login page to facilitate the students' and instructors' sign-in process.

First, error messages have been corrected to display at the appropriate time. Before, as shown in Figure 1, the error messages "Please enter your email" and "Please enter your password" are displayed upon any interaction with the input boxes, even when a valid email and password combination are entered.

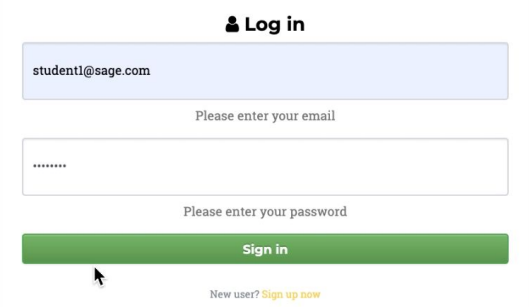

Figure 1. Error messages displayed when input boxes are filled.

As shown in Figure 2, that error has been corrected to display the error messages only when a box has been filled, and then deleted. When both input boxes are filled, these errors are not displayed.

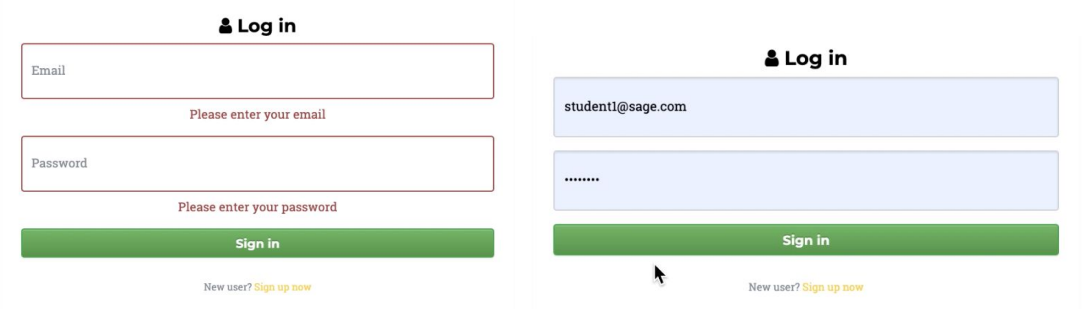

Figure 2. Error messages displayed when filled input boxes are deleted, and disappear when email and password are entered.

Secondly, there was also a login issue where, if a user entered their credentials incorrectly once, the login button would be disabled until refresh. This has been corrected, and users are able to retry their login if they enter it incorrectly.

Instructor Mission Management

We made several important usability improvements to the instructor mission management UI. The first fixed issues on the mission management page, shown in Figure 3.

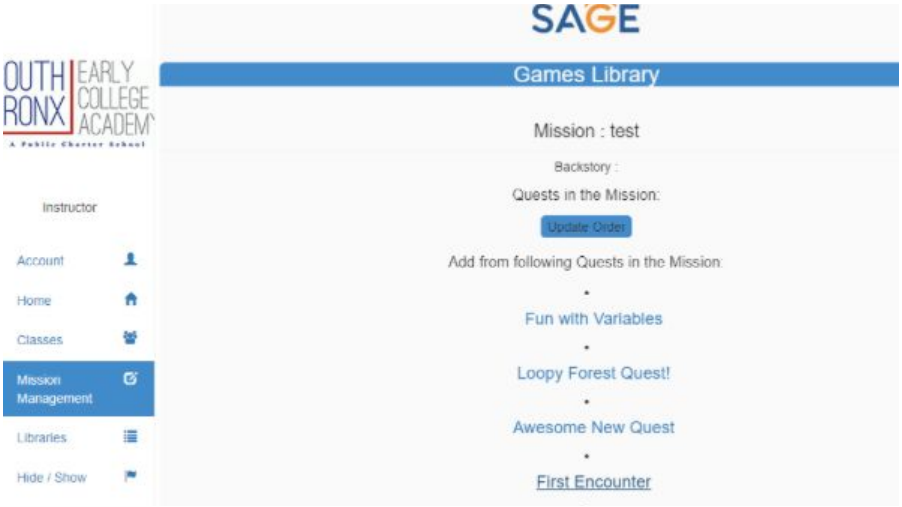

Figure 3. Original mission management page.

The page was confusingly laid out and would delete all the quests from a mission if a user accidentally added a quest that was already in the quest. We re-formatted the page, removing the stray bullets and adding an “add” button for quests not in the mission. Once a user has added a certain quest to the mission, the mission will be removed from the list of available quests, therefore avoiding any user error and preventing the bug that causes all the quest to erase. The updated user interface can be seen in Figure 4.

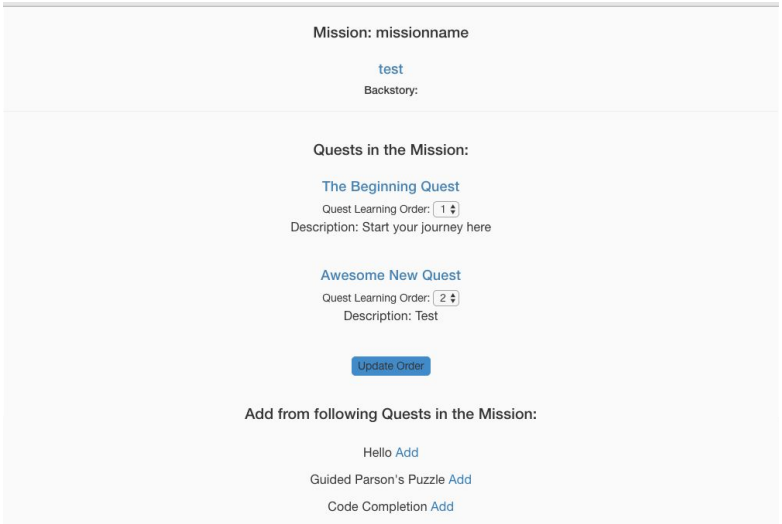

Figure 4. Updated mission management page.

We also made a basic usability fix to the main mission page. The original main displayed the entire quest and mission description, causing large overflow, as seen in Figure 5.

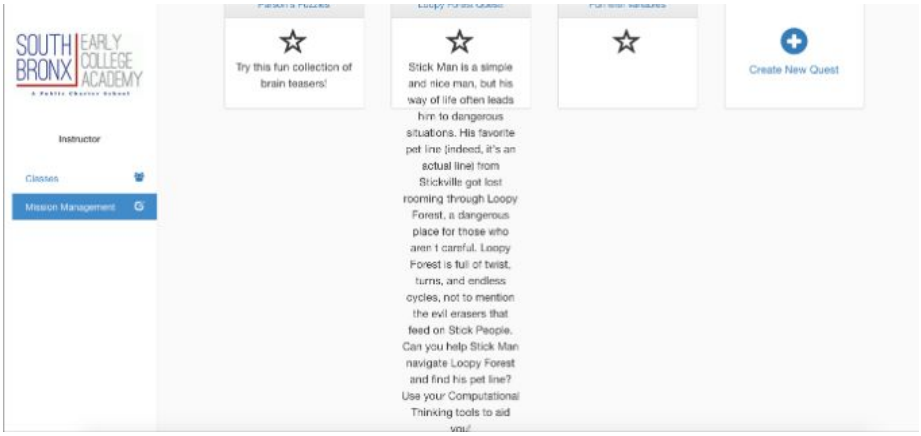

Figure 5. Original main mission page.

We updated this by limiting the descriptions to three lines and adding an ellipses to indicate the continued text, as seen in figure 6.

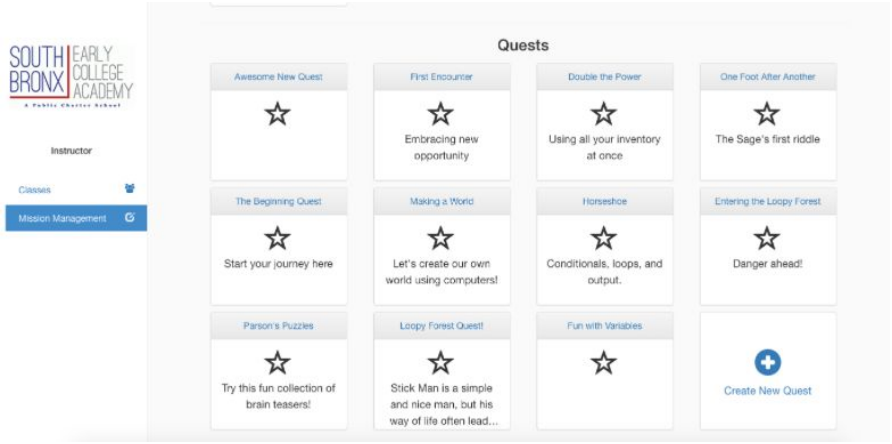

Figure 6. Updated main mission page.

Student Missions/Quests

The student Missions and Quests page was reformatted and fixed to resemble the structure of the Missions and Quests in the instructor Mission Management page. As shown in Figure 7, the page before was titled Missions in the sidebar, Quests in the header, and had Missions and Quests laid out in list format. However, clicking on the links led to an invalid pathway, as denoted by the empty highlighted portions of the URL and empty pages in the lower two photos of the figure. The highlighted portion should have had the mission or quest ID.

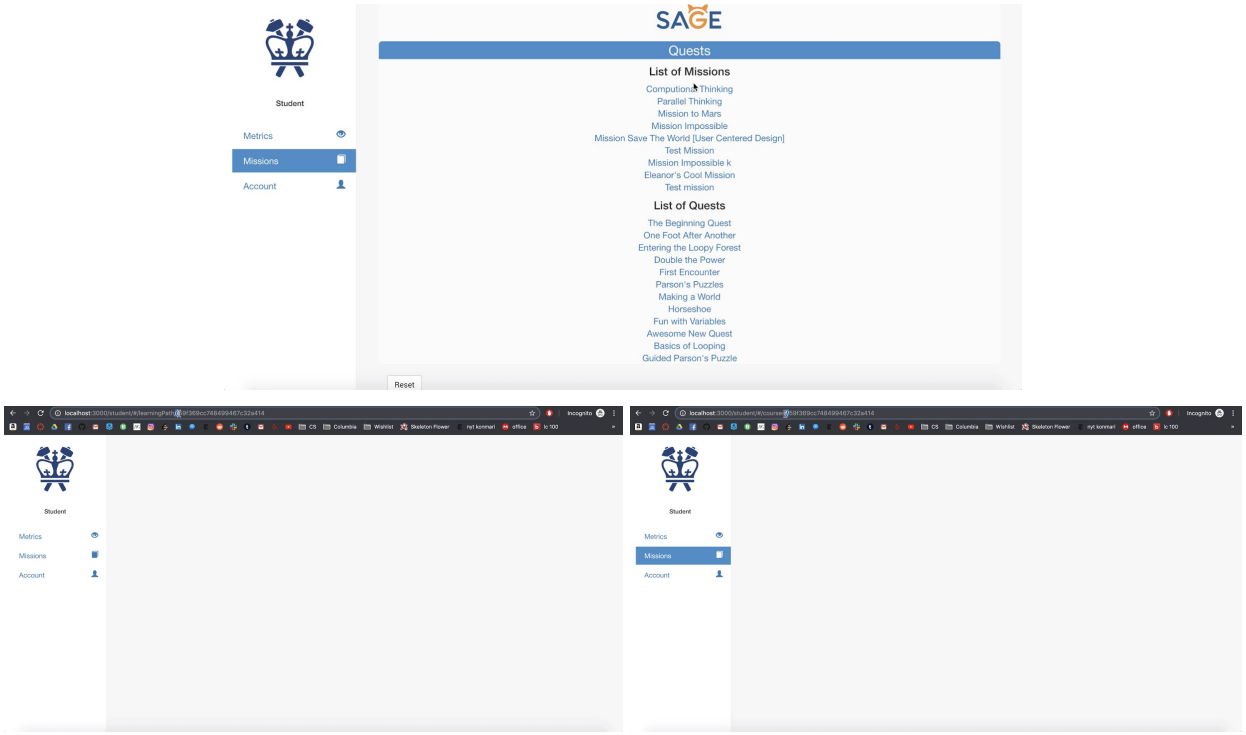

Figure 7. The original student Missions home page, followed by clicks to links under "List of Missions" and "List of Quests." Both lead to invalid pathways and empty pages.

Figure 8 shows the student Missions page after the bug fix. The structure of the page is now nearly the same as that of the instructor Mission Management page, with separate icons for each mission and quest. Clicking on any of them leads to the correct mission or quest page.

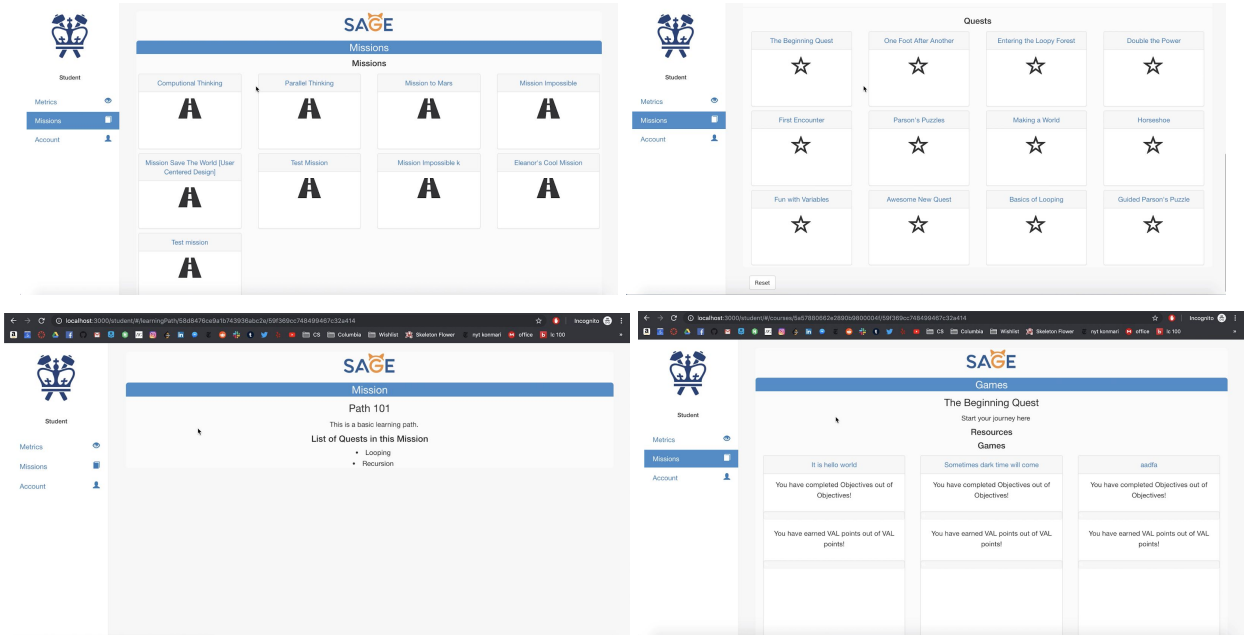

Figure 8 The updated Missions home page, restructured to look analogous to the Instructor Mission Management counterpart, with valid links for both missions and quests.

Game Objective Editor

We worked on improving the usability of the game objective editor, focusing on the id strings that were displayed. The original objective editor design, seen in Figure 9, showed the game id, a long alphanumeric string, at the top of the page. In order to select an objective to edit, the user had to scroll through a list of objective ids, also long alphanumeric strings, and select one to edit.

Figure 9. Original objective editor with non-human readable id in the title.

In order to display a name rather than a id, we had to edit the Game and Objective models in sage-node. When these models were first implemented, no name fields were in the models, hence the displaying of ids. We added name fields to the models and reworked the sage-node controllers to work with the new models.

In the new design, the name rather than the id is displayed at the top of the page. We added a placeholder for objectives that are currently unnamed, [Name], as seen in Figure 10. A user can now enter a name in the objective name field and the name will be saved. Rather than scrolling through a list of ids the user will be able to scroll through a list of names/placeholder names. Ideally the objectives will all eventually be named and there will no longer be placeholders.

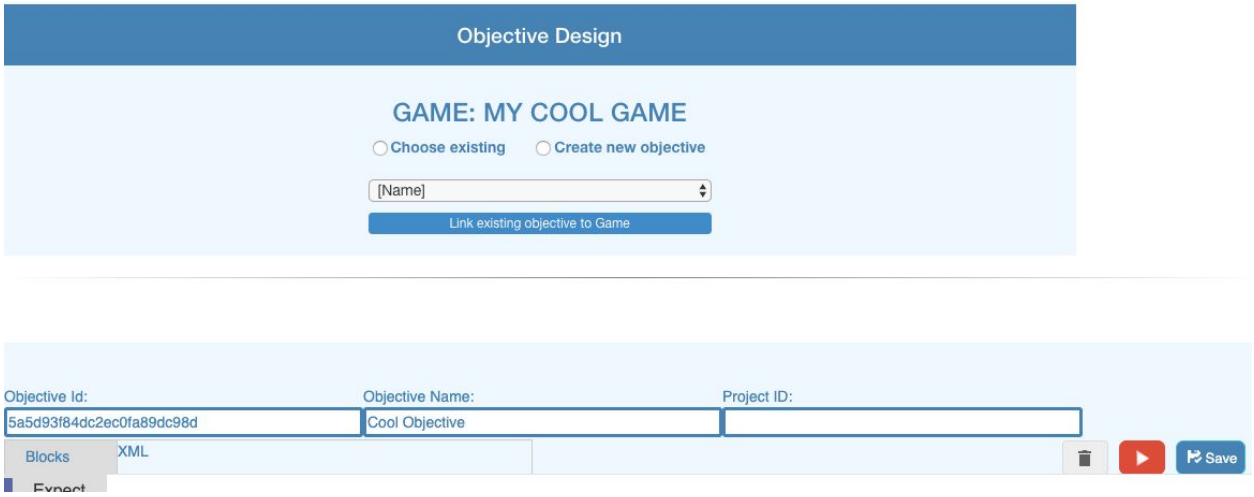

Figure 10.New objective editor with game name and objective name.

Learning Metrics Analysis

Hairball

Hairball-Kurt is the Python package we used to analyze the sb2 data from student game submissions. Our goal was to gain access to the student game data (stored in the sage-login database) from sage-node (where hairball is run) upon submission, analyze the data with hairball, and insert the output into the sage-login database so the results can be visually displayed by sage-frontend. Our attempt was partially successful; the progress will be documented in the following two sections, Database Manipulation and Result Storage and Display, and shortcomings will be described in the Learning Metrics Analysis sections of Limitations and Future Work.

Database Manipulation

In order to show the results from Hairball analysis of a students' game, it was necessary to run Hairball on the students' submitted game, parse those results, and store them in the MongoDB. In order to access the pertinent data from the games, such as game and student IDs, we decided to connect the sage-node layer of the architecture to the sage-login database. This was a bit of a work-around, and we recommend that this information be stored and accessed from the sage-node database in the future. Following this, we used existing code to run Hairball on the students' submitted game file (an SB2 file).

Result Storage and Display

After running Hairball, we parsed the individual Computational Thinking scores for each category (Data Representation, Abstraction, Parallelization, Logic, Synchronization, Flow Control, and User Interactivity). This data is then saved to sage-login, along with the student and game IDs. Here, we saved empty information for game objectives, as they are not as relevant to the Parsons Puzzle games. When the game data is saved, it is saved along with an identifying student ID, and each entry with a student ID has all of the Hairball scores for the games they have played as of that entry. Because this data is saved in sage-login, sage-frontend is able to access it and populate Computational Thinking score spider graphs for the student at the Mission, Quest, and Game level.

Shown in Figure 11 is an example game that a student submitted, which features an Event block (when the green flag is clicked) and a Control block (a for loop).

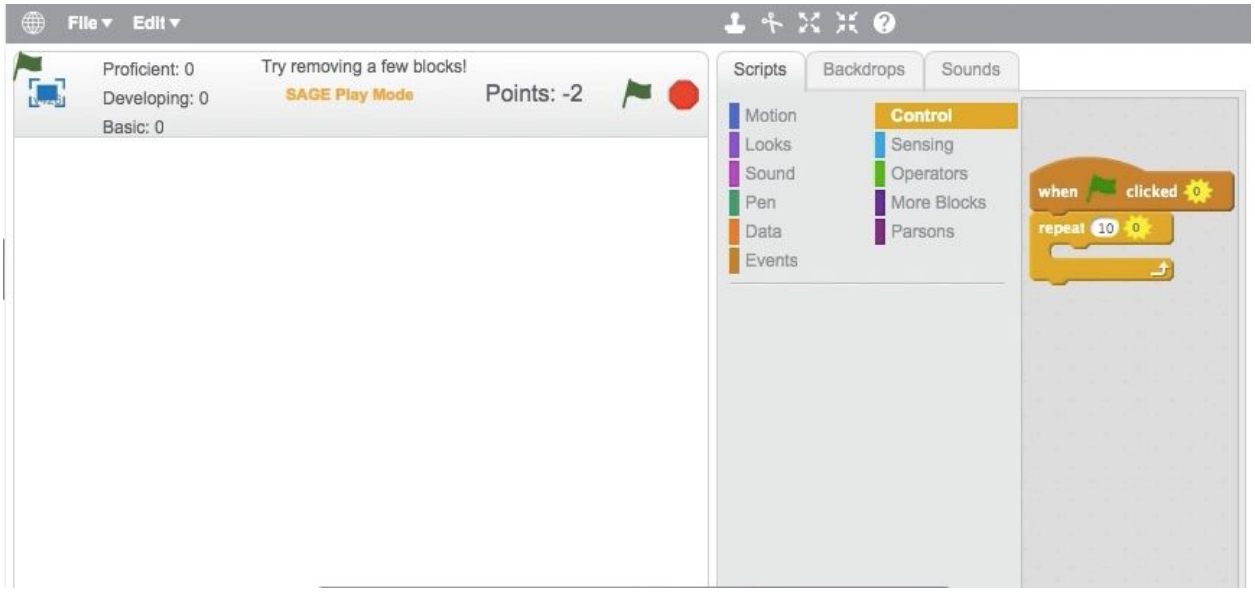

Figure 11. Student playing a game and adding event and control blocks

Once this game has been downloaded, Hairball is run on it to analyze what kinds of Computational Thinking skills the student displayed while playing the game. Shown in Figures X and X are the spider graphs that display the student's Computational Thinking scores in relation to the blocks they played in the game in Figure 11.

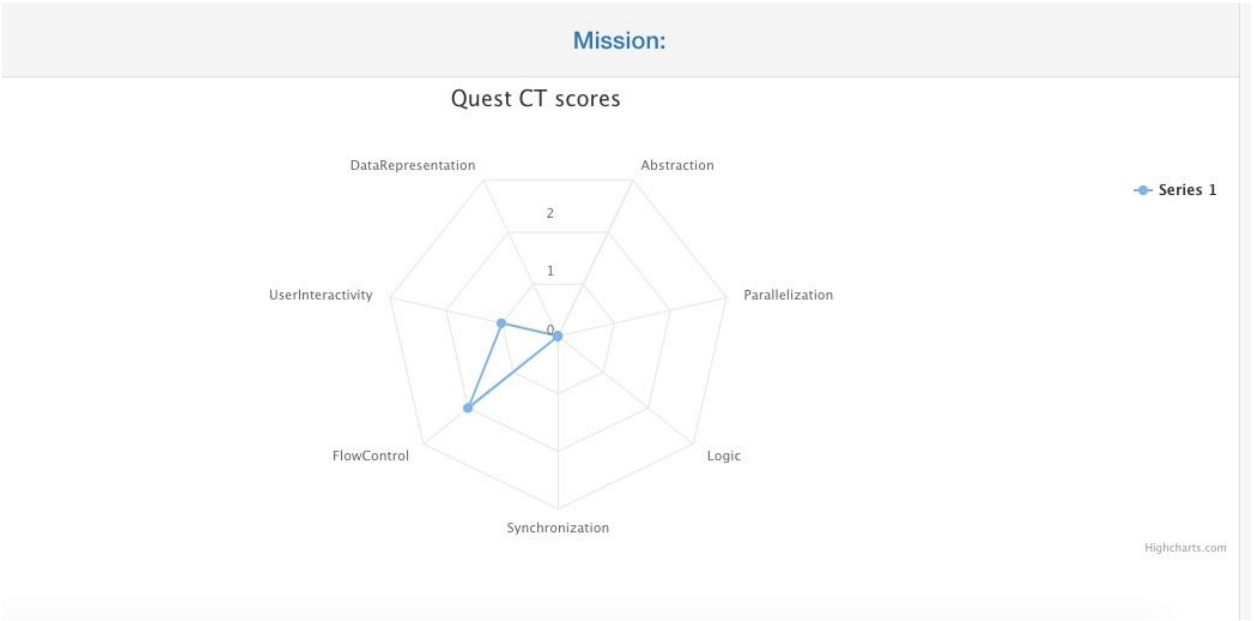

Figure 12. The student's Computational Thinking scores based on all games played in this mission.

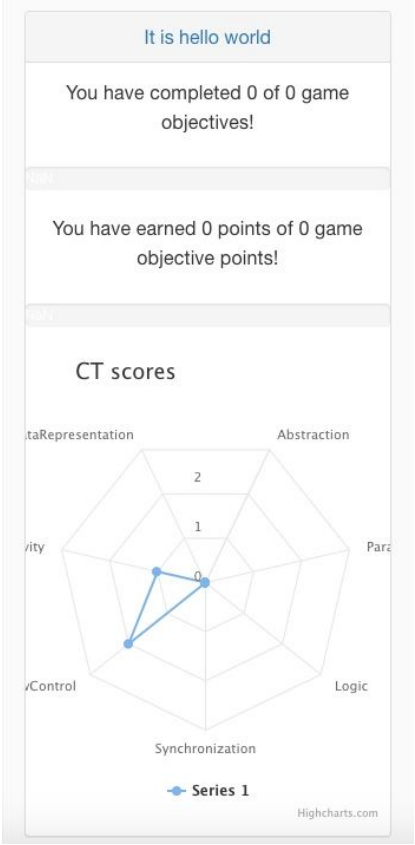

Figure 13. The student's Computational Thinking scores based on the game played in Figure 11

As can be seen in Figure 13, the student received points for Flow Control and User Interactivity, which are Computational Thinking categories that correspond to the blocks they played in the game in Figure 11.

Limitations

User Interface

The most prominent user interface limitation was the data models for games and objectives in sage-node. The models lacked name fields, meaning we had to create the fields and modify many of the backend functions in order to display basic data like names on the frontend.

Learning Metrics Analysis

We faced a number of complications during our work in learning metrics analysis.

The folder that contained the code running hairball (sage-node directory) was not connected to the database (sage-login database) that held the game submission data and analysis results to be displayed (sage-frontend directory). We fixed this by connecting sage-node to the sage-login database, so that it could retrieve game data, run hairball, and input the results into the database.

We had difficulty producing a valid SB2 file from the data already in sage-login. After connecting sage-node to sage-login, we tried to manually download the SB2 game file in its binary form, but running it in hairball proved that files produced this way were incompatible with the analysis program. Currently, our way of producing valid SB2 files is to manually download them from the scratch game right before submission.

Even with a correct SB2 file, it took a while to be able to run Hairball successfully on it. Attempts to recreate the hairball results produced in Ruimin Zhao and Neng Chen's Learning Metrics [3] report from Fall 2018 were unsuccessful until we not only ran the code in sage-node, but also altered the Kurt library in our local machines to account for the discrepancies between conventional Scratch SB2 files and SB2 files created by sage-scratch.

After we had resolved the previous two issues (we were able to get hairball running without error on valid SB2 files), our goal was to be able to download or access the game information upon submission without a manual download. Our attempts to do so via local download and GET/POST requests to and from the server have not been successful as of now.

Future Work

User Interface

When creating new data models for SAGE, we recommend always creating a field for a human readable name alongside the unique id entry. While slightly more work in the beginning, this will enable users to easily interact with the data from the start. It is certainly more difficult to go back and add this human-readable name after the entire model has been created, especially if the team that wrote the model is not the team modifying it.

We recommend that when field tests are conducted, the experimenters specifically test the user interface. The user interface can determine whether or not a user can interact with or understand an app and can be vital to a student or instructor's successful usage of the app. We suggest study participants go through the flow of the app and note pain points or points of confusion to the experimenters in order to generate an interface that is easy and intuitive to use, and does not detract from the computational learning goals of SAGE.

Learning Metrics Analysis

The goals in creating clear and consistent Learning Metrics analysis for SAGE students is to be able to obviously and dynamically display their progress to them, as well as their instructors. Moving forward with this, the dynamic aspect of this process must continue to be improved. Currently, the Hairball analysis and display of accurate Computational Thinking scores relies on the manual downloading of the game SB2 file. This is not sustainable, and the next steps in this project should be to conceive of a way to trigger SB2 download to the SAGE server when the student submits their game — this would allow for Hairball to run on the correct file and calculate the student's Computational Thinking scores.

Based on the work discussed in the Learning Metrics Limitations section, it appears necessary to download the SB2 file through sage-scratch — though this poses difficulties as the game submit button is located in sage-frontend. This process of determining how SB2 files can be downloaded in real-time is essential for rounding out the Learning Metrics display for SAGE students.

In addition to real-time download of SB2 files, another aspect of future work with Learning Metrics is to ensure that future researchers are working with the version of the Hairball-Kurt package that has been modified to work with SAGE. There may be some refactoring in installing necessary to ensure that, when running Hairball, the researchers are not running the original version.

Conclusion

In this report we show our work on the User Interface and Learning Metrics for SAGE. We first discuss related work and then detail the features we implemented and bugs we fixed in these two areas. With the goal of preparing SAGE for a field study, we focused on user interface issues that hindered basic usability of the application. We addressed issues with login, Mission Management, quest view, and the Objective Editor. We applied the Hairball metrics package to the student game files, and displayed those Computational Thinking scores directly to the student. Our contributions this semester helped make SAGE into a more robust application and brought it closer to being sufficient for a field study.

References

[1] Moreno-León, Robles, Román-González, *Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking*, 2015

[2] Kazimoglu, Cagin, et al. “Understanding Computational Thinking before Programming.” *International Journal of Game-Based Learning*, vol. 1, no. 3, 2011, pp. 30–52., doi:10.4018/ijgbl.2011070103.

[3] Chen, Neng; Zhao, Ruimin. *Learning Metrics*. December 19, 2018.