SAGE Final Report:
Scratch Experience & UI Improvement

December 17th

Penghe Zhang

Linnan Li

Outline

1. Abstract

SAGE is a website that helps students in grade 6 to grade 8 learning computational thinking. There are two main roles in SAGE: instructor and student. As an instructor, user can design parson games or cvg games and assign them to specific class and student. For students, SAGE will have a virtual instructor in the right panel and give feedback and guides to students. Our work can be divided into two parts: scratch improvements and UI improvements. We created different ways to use blocks in the scratch panel according to different research purpose. We also refine the right panel that displays score and feedback to students, which gives our users better experience.

2. Background

The social Addictive Gameful Engineering research project is a website designed to help 6-8 grade students study computational thinking concepts. Parson's puzzle is a major kind of puzzle that can be designed by instructors. The instructor will create parson game, select several blocks and put those blocks in parsons panel. The student will select blocks inside parsons panel and create block sequence.

The virtual instructor's feedback is an important part of SAGE. On the right side of the webpage, it's the instructor panel, which is designed to display all the information provided to students. For example, students can solve the puzzle based on the feedback displayed on the right panel. The look of right panel also needs to be modified for better user experience.

Since we are going to do research for different modes of parsons puzzle, we need to change the behavior of scratch and feedbacks accordingly. For example, there are two modes of parsons puzzle used in the research: general mode and no feedback mode.

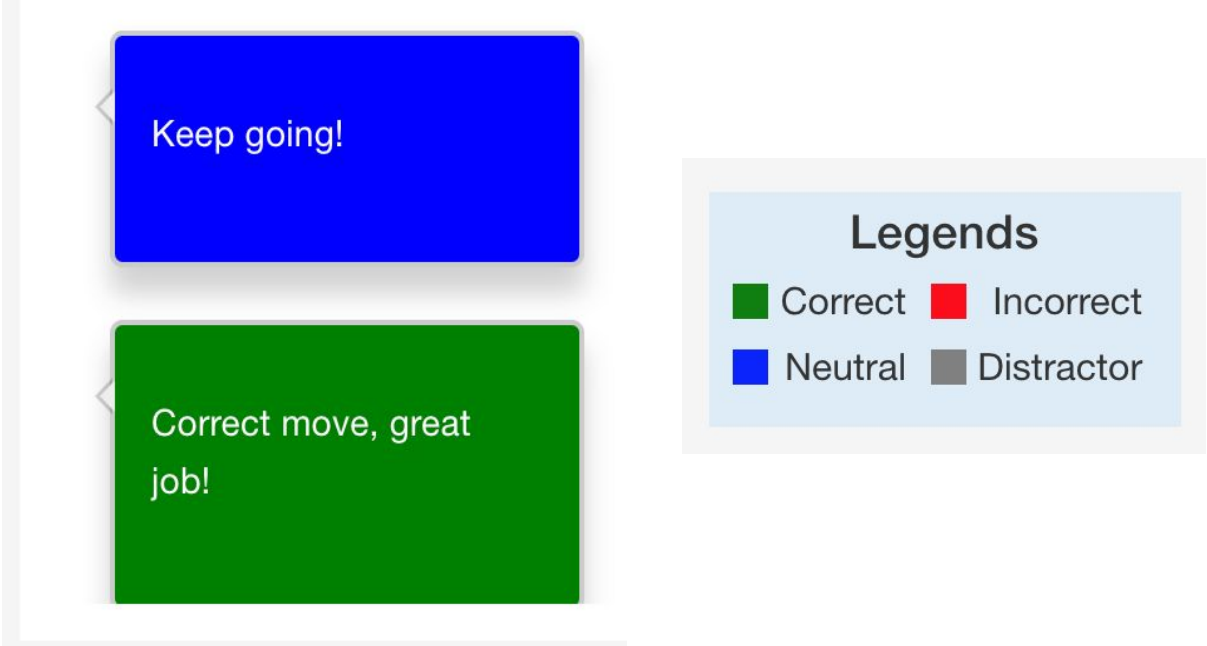
3. Implementations

3.1 UI Improvements

3.1.1 Feedback Improvements

The right panel in a puzzle-solving page is where we display all the feedback and information to students.

We use the concept of virtual instructor who gives students feedback after each move. There are basically four types of feedback: incorrect move, correct move, neutral feedback and distractor feedback. To make the feedback as an indicator for student's solution, we make different types of feedback to be in different colors.

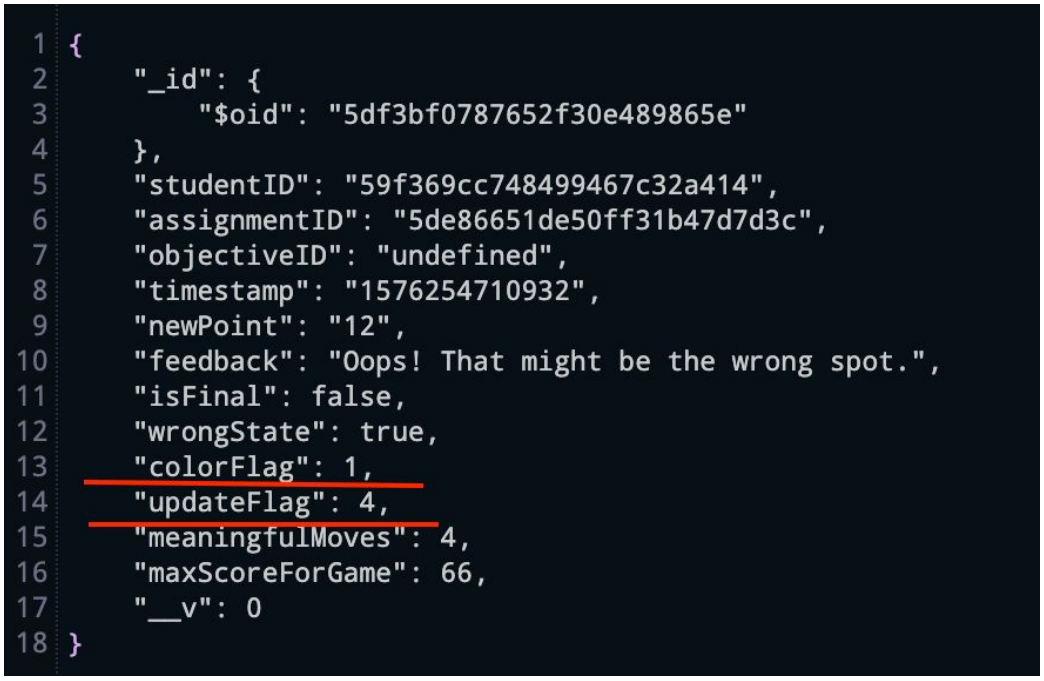


We add an attribute called colorFlag in the database. When the feedback is generated in scratch, we assign different value to colorFlag and upload it to database. While displaying the feedback, we dynamically change the background color according to the flag.

We still use a pull model to retrieve feedback from database per second. To avoid refresh the page or display the feedback multiple times, we add another attribute called updateFlag in database. In scratch server, if a block

has been moved, we plus 1 on updateFlag so that the front end knows there was new feedback generated.

These two flags are in the *tempstudentscores* collection, which is shown below:

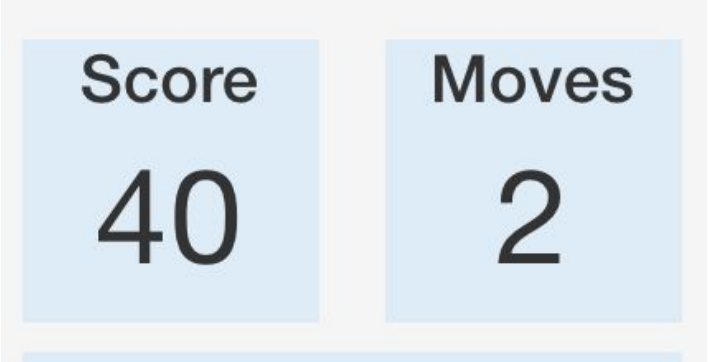


3.1.2 Score and Other Information Improvements

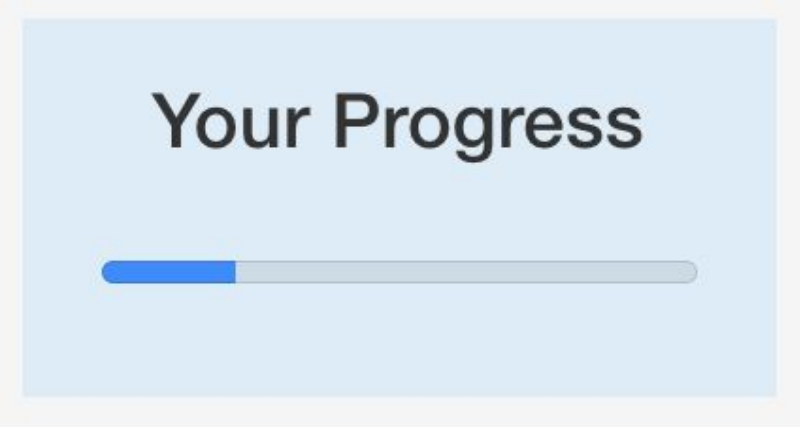
In the previous implementation, score and other information was displayed on scratch palette. We moved them out of scratch and place the information in the right panel.

Each move happens in scratch will be calculated to temporary score and feedback. We use socket connection to transfer these data to database. Therefore, all information displayed on frontend right panel is retrieved from database.

We also use a pull model to get score from database per second. For each time, if the score is different from the previous score on record, we reset the current score. Except score, we also show meaningful moves next to score as another indicator for students. Following is the look for this part:



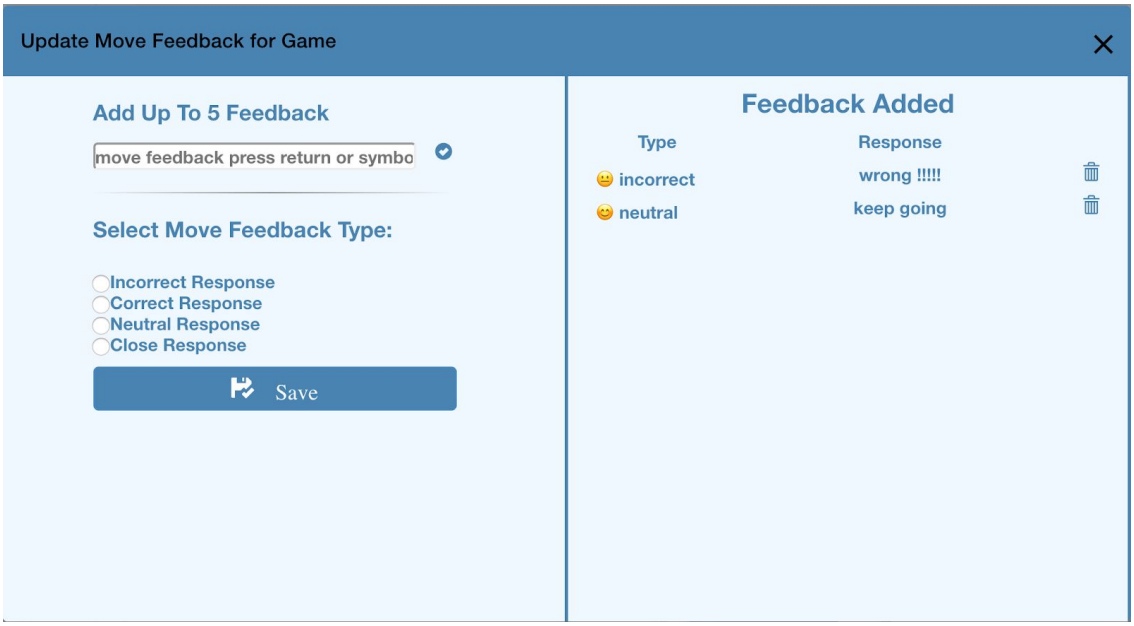
The progress bar is another feature we added to give the student hint. While instructor creating the game, a maximum score will be calculated automatically. Therefore we can use the current score and maximum score to calculate the progress. The formula for calculating progress is simple: *current progress = current score / max score. Following is the look for progress bar:*



3.2 Scratch Improvements

3.2.1 Customized Feedbacks Enabled in Assignment

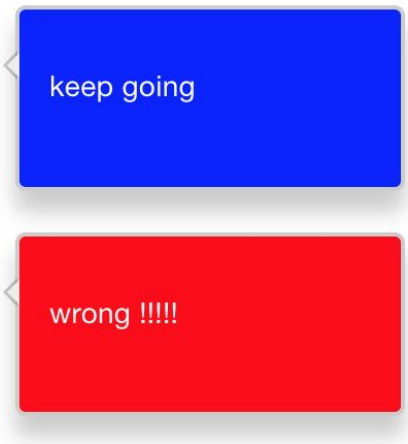
Previously, we can only use predefined comments to give students per-block feedback. Now the instructor can wrote their own customized feedback in the front-end:



And we will save all these movefeedback along with other information of the assignment in the MLab:


```
{
  "_id": {
    "$oid": "5db4ed770857c75ac46c588d"
  },
  "assignmentName": "pz2244 game test",
  "assignmentOrder": 1,
  "courseId": "5cb09fdade42bf36e143d1b5",
  "creatorId": "59f8c6fdc1bfb23c4ced8e20",
  "type": "parsons",
  "assignmentFeedbacks": [],
  "moveFeedbacks": [
    {
      "type": "incorrect",
      "content": "wrong !!!!!"
    },
    {
      "content": "keep going",
      "type": "neutral"
    }
  ],
```


Then we wrote a new API in “sage-node”, “`/:id/getmovefeedback`”, which can fetch all the movefeedback from the MLab. Also, to let scratch read these movefeedback, we also write a new function in scratch code, which is “getSavedData()”. Thus, we will show those customized feedback to students via the right panel in the front end:

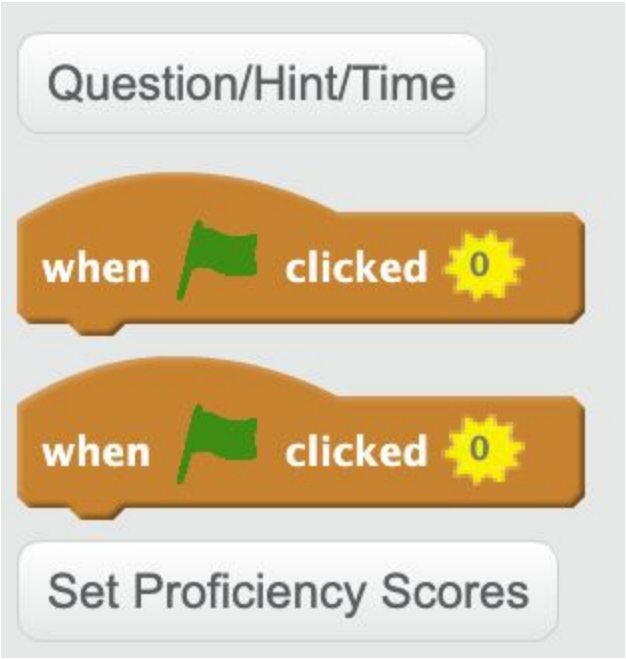


3.2.2 Number Box for Blocks

Previous, there is only a checkbox in front of front end to tell the instructors whether this is the block used in Parson's palette:

And in this way, we will only show one block in the Parson's palette. However, there may be some situations, the instructor used more than one same block in his/her puzzle, if we still show one block in the Parson's palette, the students might be mislead. Thus, we have changed it to a number input box for instructor, where the instructor can wrote exactly number of blocks they want to show in the Parson's palette:

And in the parson's palette, we can see multiple blocks with same content:



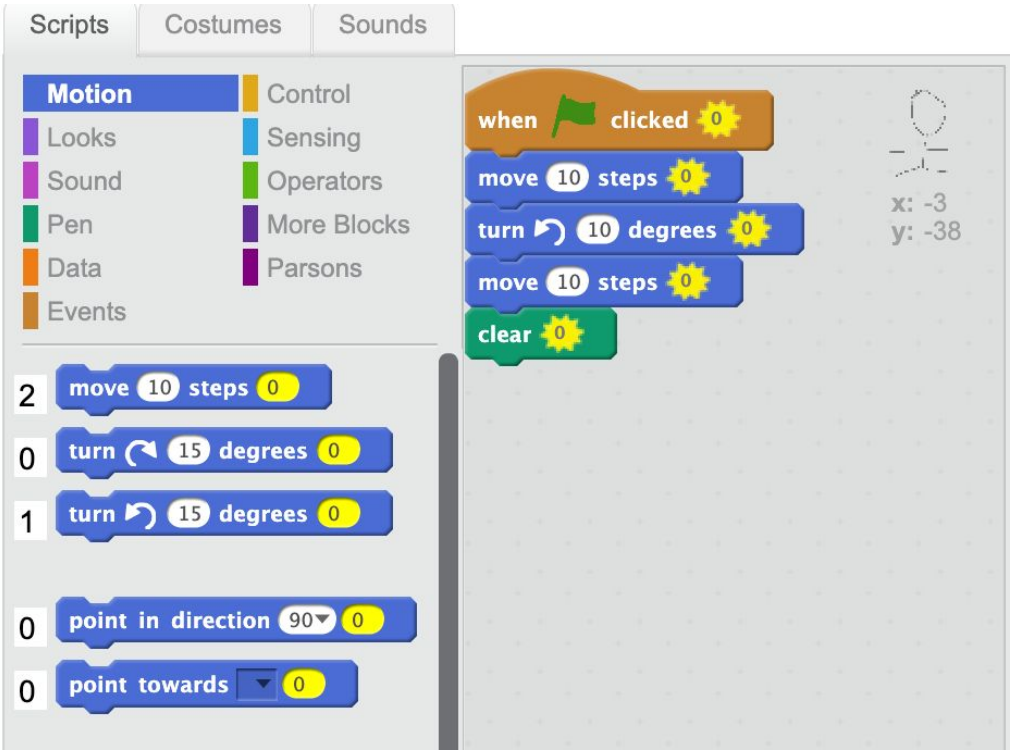
3.2.3 Different Modes for Parson's Puzzle

Since we are going to add a new parson's modes in our parson's puzzle, thus we need to make some differences in different modes which can improve the user experience both for students and teachers.

1. Parson's Normal Mode:

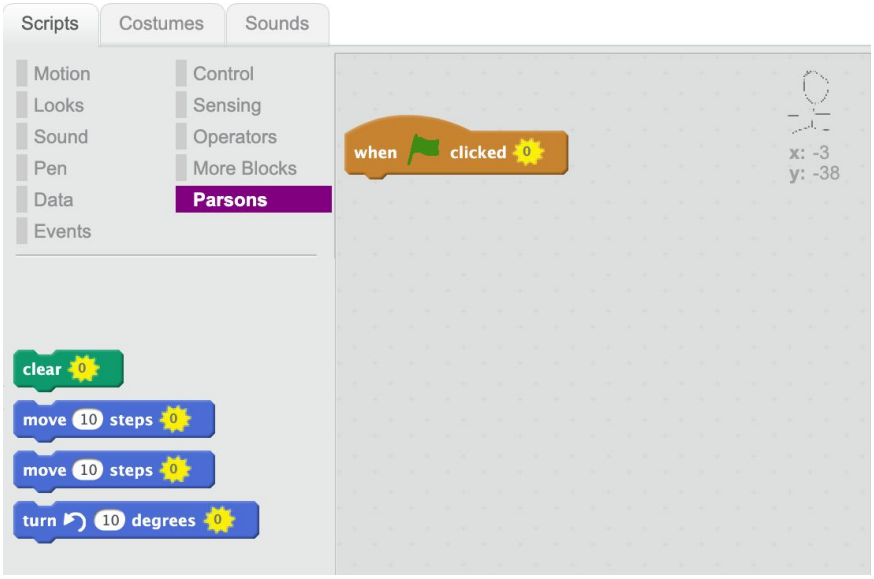
In this mode, everything will be really similar with the Parson's mode we design previously.

Instructor's side:



Instructor can see all the palettes including Parson's palette in this model, instructor can enter how many blocks he/she want to show in Parson's palette.

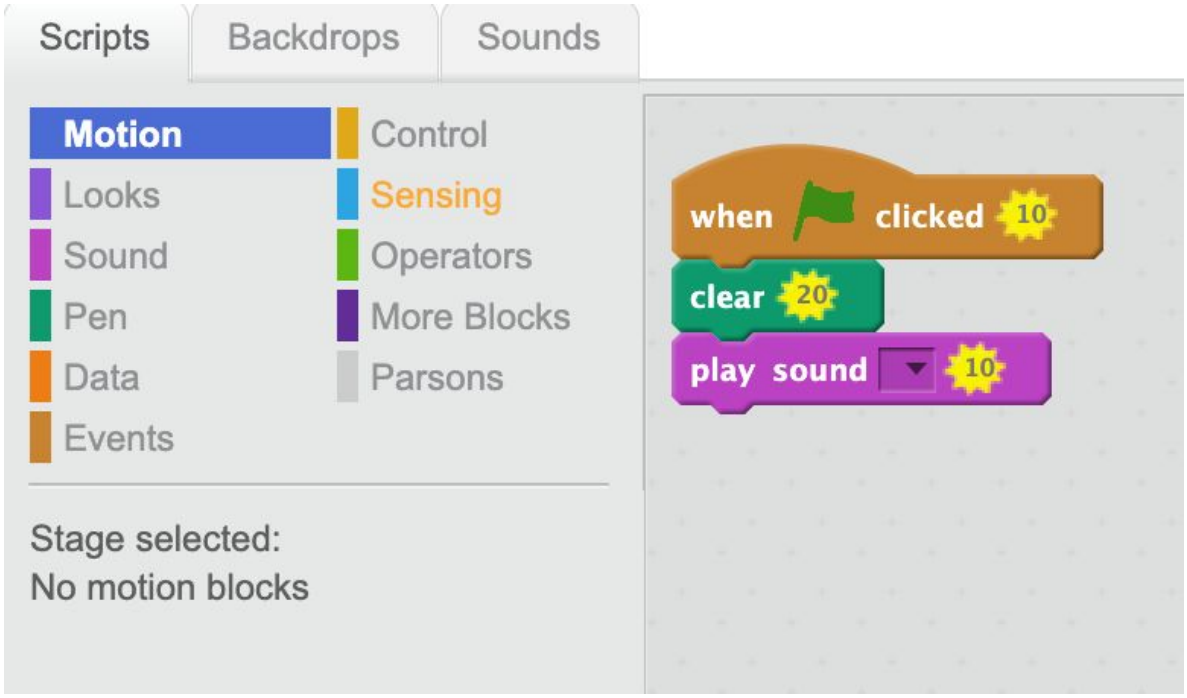
Student's side:



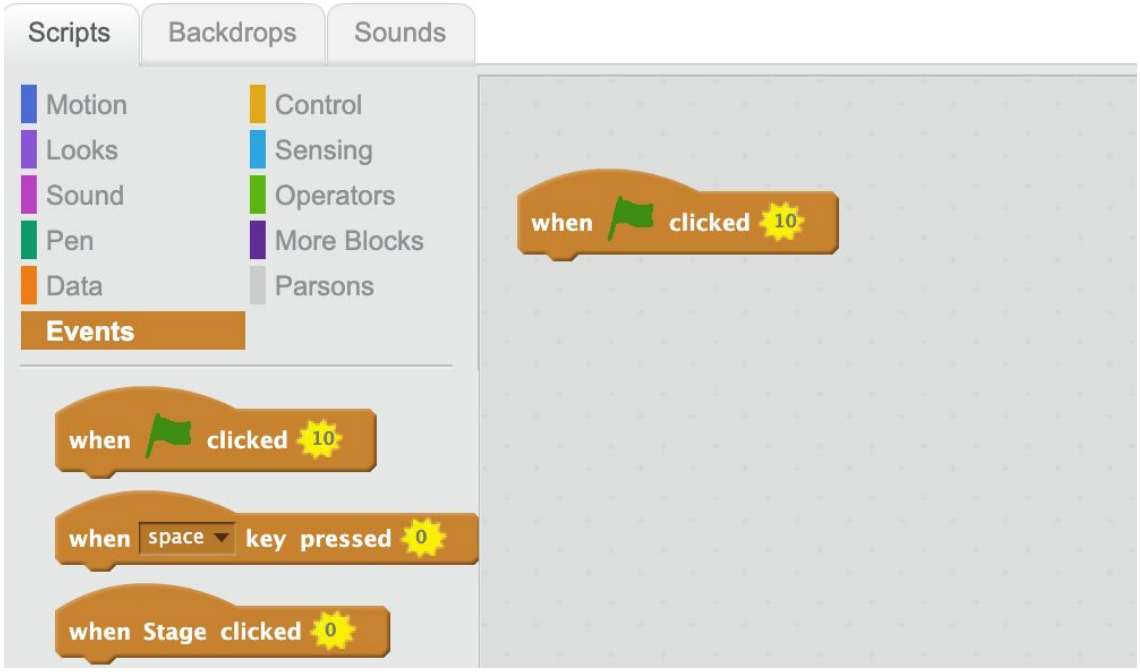
In student's side, we can see all the palettes except the Parson's palette are shown in grey color, which means they are not clickable, it will make students focus on the Parsons palette.

2. Parson's No Feedback Mode:
In this mode, more Parson's palette will be available in the palette pane. Although, parson's is unavailable, we will still show it in a grey color, which can tell students they are still in Parson's puzzle, not CVG mode. Students can drag any blocks in any other palettes in this mode. Also, there will be no more scores or feedback in the front end.

Instructor's side:



Student's side:

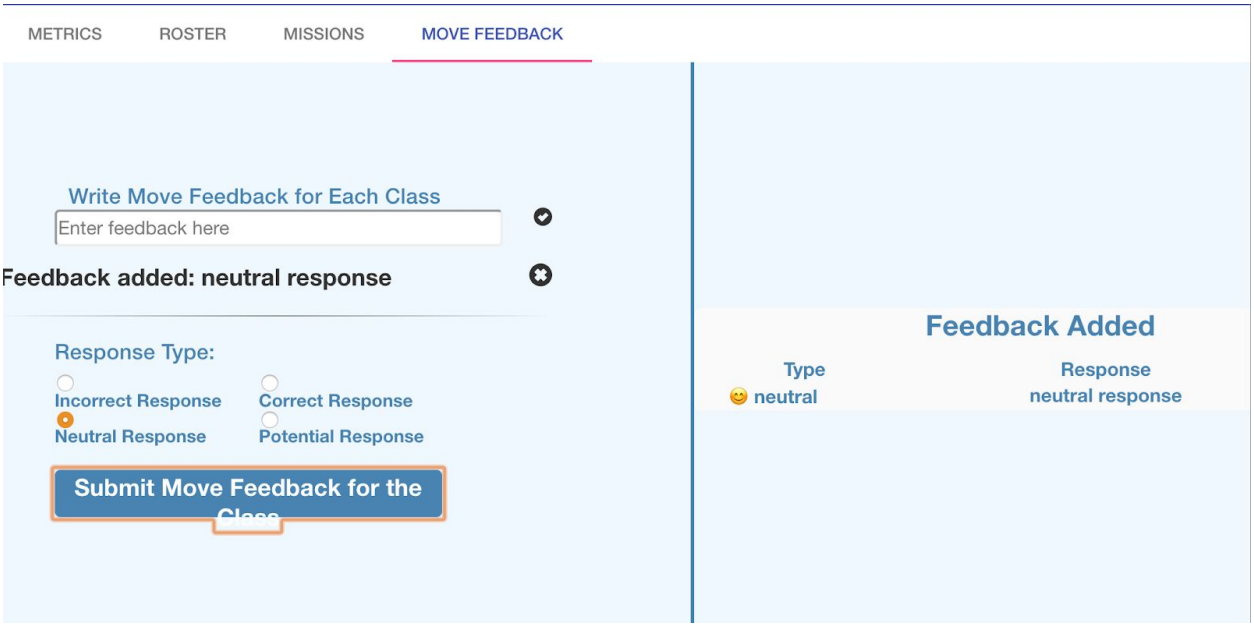


3.3 Bug Fixed

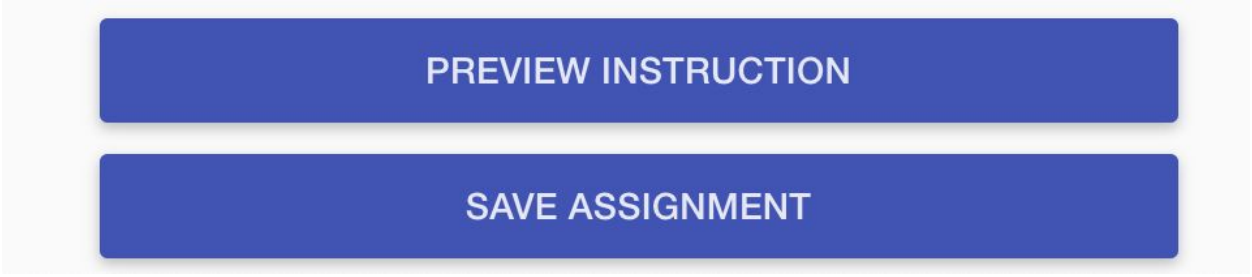
We have fixed nearly 20 bugs in the SAGE system;

3.3.1 On the instructor's side

1. Cannot add customized feedback in course level:
   The "move feedback" tab in the "Class" page was unclickable, we fixed this bug by rewrote the code with new API. Now the teacher can add their customized feedback successfully. The file we changed is "InstructorUpdateMoveFeedbackController.js"



2. Cannot create a quest successfully:
   Previously, when instructor create a new quest, it will not be saved in the MLab, we fixed this by modifying the url in the backend.
3. Tab number bug in the classes' page:
   Now in the "Classes page", no matter which tab you clicked in, you will always see four tabs.
4. Cannot delete a game in the quest:
   Fixed this bug by adding a HTTP delete method.
5. Cannot update instruction in the singe-quest page:
   Fixed this by adding the assignmentID in the http request.
6. Redesign the logic when instructor create a new game:
   Previously, the logic of the game create page is confusing, we delete some useless buttons, only keep "preview instruction" and "save assignment" buttons. After the instructor click the "save assignment" button, it will redirect to the game design page instead of no response.



7. Cannot delete customize feedback in the assignment design page.

8. Only show cvg instruction whatever mode of the game is:
   Wrote a new API in the front end which can judge what type of current assignment is.

3.3.2 On the student's side:

1. Cannot read games in quest page:
   Fixed this bug by using a new HTTP request instead of the previous wrong one. Meanwhile, shorten the time complexity from O(N) to O(1) by removing useless for loop.
2. When submit a game successfully, the front end will crash down:
   Fixed this bug in the "server.js"
3. Cannot update scores in time:
   Upgraded the timeout function and start the timer whenever the game is loaded.
4. Cannot stop timer when finish the game;
   Add a finish flag which can inspect whether the game is finished or not, when the game is finished, stop the timer.
5. Didn't stop the timeout function even submited the result;
   Replace all the old timeout functions in the files with new $timeout function in Angular.js.
6. Cannot redirect to home page after submitting the game:
   Rewrote the redirecting url.
7. Always show the same information in single-course page:
   Fixed a typo in the front end.
8. Add limitation on feedback after submitting the quest:
   Add a max-length attribute in the html file.

3.3.3 On the Scratch's side:

1. Cannot read cvg/parson's mode correctly, always show the Parson's palette:
   Wrote a function in SAGE-scratch and SAGE-node, where we can get to know all the information saved in the "assignment" model.
2. The scratch will crash in play mode if there is no designed script:
   Add a try catch to prevent the situation when there is no corrArr in the script.

3. Cannot send a correct message to front end when finished the puzzle:
   Updated the information sent to the front end.

5. Future Work

1. Transfer the Scratch code from ActionScript to JavaScript (Based on Scratch v3);
   Currently, our scratch platte is implemented with flash, however, flash will not be supported by Chorme. Therefore, it's time to change from ActionScript to JavaScript. With this change, the project will have more consistency as well.
2. Clean out useless and outdated code in the front end code;
   There are some old code has been commented, which make it hard to read and understand the code base. We suggest to sort these code out. For example, remove useless code and add comment before code blocks that could be reused in the future.
3. Per-block feedback logic can be upgraded;
   In our implementation, we didn't change the logic to generate feedback for each move or block. In the future, we can upgrade the algorithm generating per-block feedback to make it more accurate.
4. Display those metrics in the web page.
   There is a metrics collection in our database currently. It can show the review of a class or a student. This part of data could be displayed in the future for research.
5. Change the whole project to a test-driven mode.
   Add test for each function in the code base is a good way to reduce the time for debug. The developers will benefit a lot from a test-driven mode.

6. References

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

[2] Bender, J. (2015). Developing a collaborative Game-Based Learning System to infuse computational thinking within Grade 6-8 Curricula.

[3] Jeannette M. Wing. 2006. Computational thinking. Commun. ACM 49, 3 (March 2006), 33- 35. DOI: https://doi.org/10.1145/1118178.1118215

[4] Mohan. (2017). Gameful Direct Instruction (Parson's Puzzle)