Formative SAGE Assessments: Parson's Programming

Final Report

Akanksha Gupta (ag3749@columbia.edu)
Sudhanshu Mohan (sm4241@columbia.edu)
COMS E6901, Section 14
Fall 2016

TABLE OF CONTENTS

1.0 Introduction

The motivations behind this project are to include 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) concepts via personalized feedback on how they perform on Parson's puzzles.

1.1 Computational Thinking

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.

1.2 Concept of Parson's

Parson's puzzles empowers teachers/authors to impart gameful interactive learning to students. Not just designing, we ensure students get real time feedback with every move they make. Students are guided towards the solution, rather than rote-learning. Each puzzle is a complete solution in itself and teaches the student a concept. Such puzzles expose students to short pieces of code that help students construct the logical order of statements for any Parson coding puzzle. 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 and leads to active learning rather than a passive learning.

In recent times, there has been a debate on how much importance should the teacher give on extrinsic factors like giving bonus points and badges to students who perform better versus just showing the final marks straight. We propose that such extrinsic motivational factors help the students to actively learn the concept and not rote-learn the concept. Further, if students have intrinsic motivation for coding puzzles as a hobby, then the amalgamation of intrinsic and extrinsic factors will lead to computational gameful thinking abilities in the students. Further, sprites and gameful moves can be included to engage the students.

1.3 Objectives

This document will describe the work completed for the SAGE Assessment: Parsons' Programming project. The objective of this project are to:

1.) Provide capabilities to author/teachers to create Parson's puzzle using the Design mode
2.) Provision to enter problem statement and hints for students to solve the Parson's puzzle
3.) The teacher should also be able to configure points for each block which help in assessing the final score for Parson's puzzles
4.) For interactive and guided learning, scores should be dynamic and guide students towards the right solution.
5.) Students should be able to solve Parson's puzzles and try achieving high scores.
6.) Further to aid students learning, they can use the hint provided by teachers. Also, to enable communication, the student can record their feedback about the experience of solving the puzzle.

2.0 Earlier SAGE

The earlier SAGE allowed the teachers to play the role of game designers and create assignments for students and involve students in a computational learning experience. By computational thinking we mean the approach to solve problems using concepts of abstraction, recursion, and iteration. Computational thinking is critical and is crucial for overall learning of the student. Teaching computational thinking, however, has its own challenges. Many teachers do not have the training to lead classroom discussions or prepare assignments to build this skill in their students. And even when teachers are motivated, it is a lot harder to keep students engaged because these skills can be difficult and frustrating to grasp at first.

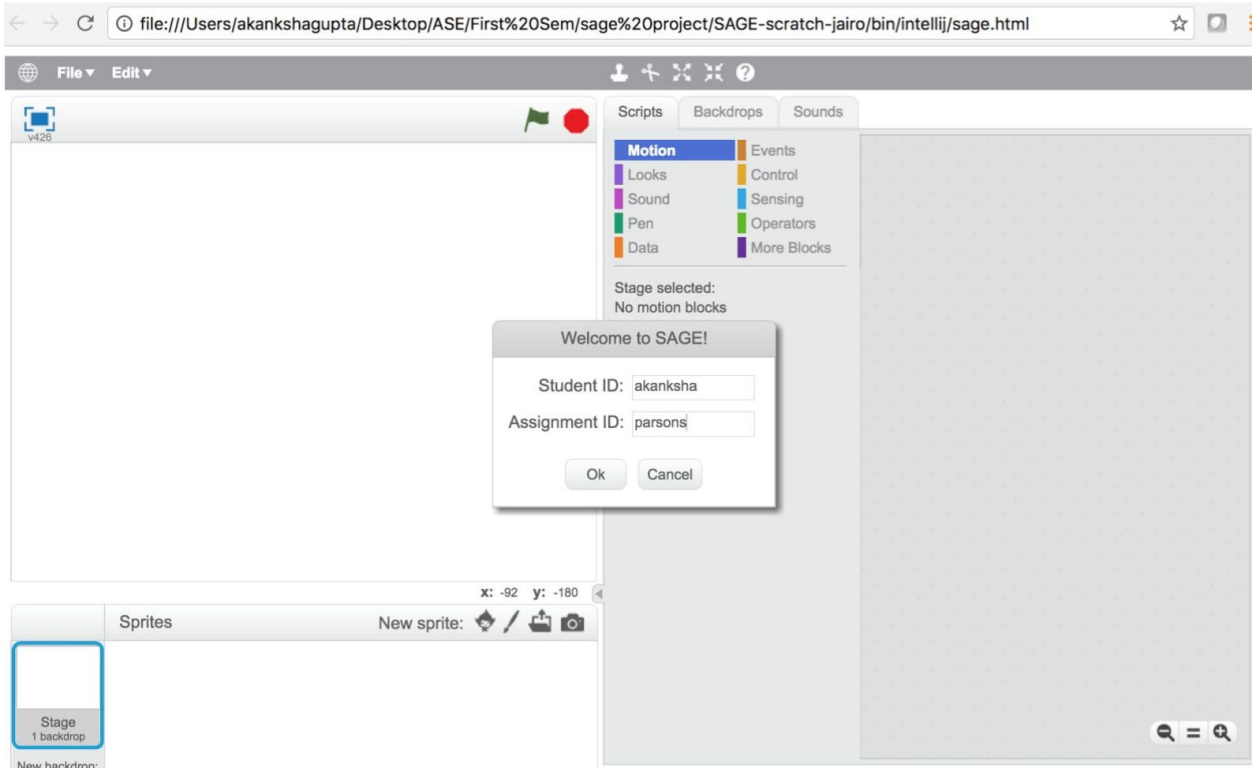


Figure 1: Earlier SAGE screen

Learning syntactic and logical constructs and differentiating between the correct and incorrect solution is essential to learn how to code. Writing full code from scratch is tedious, especially for students who are new to coding skills. A student new to coding should first be taught the basics focusing on logical errors and syntax. Without a good grasp of syntax and logic, it is difficult for students to write complete programs as syntax and logic form the base of learning to code.

3.0 Related Work

3.1 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 2). 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.

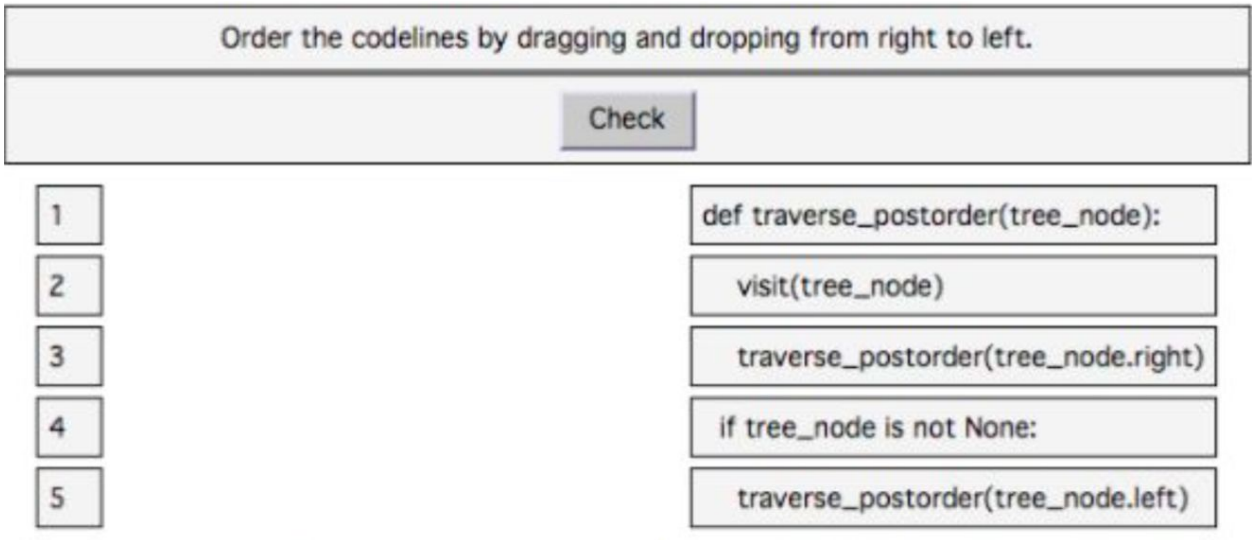


Figure 2: A Parson's problem in Hot Potatoes

3.2 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 3.

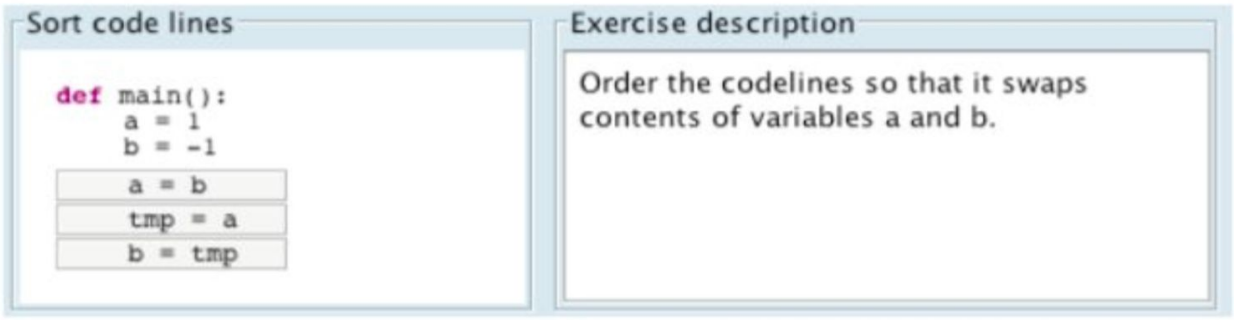


Figure 3: ViLLE

3.3 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.

3.4 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.

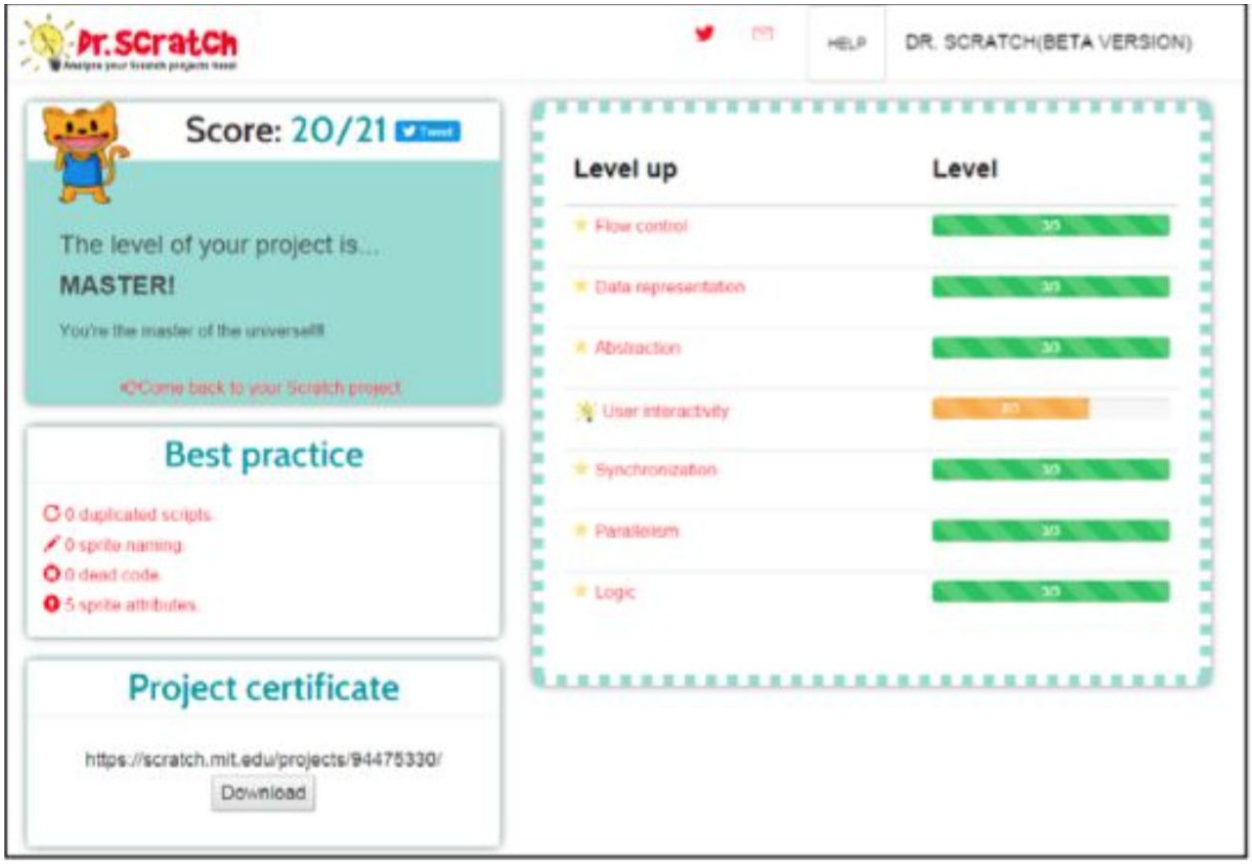


Figure 4: Dr. Scratch scoreboard

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.

4.0 Vision

4.1 Traditional Parson's

The traditional parson included a set of statements in random order. The student will just drag and drop them in order. On submission of the puzzle, the student will see the result. This has been more of a static testing where the teacher's solution is compared with the students solution statically. This does not help the teacher learn factors such as was the student able to solve the puzzle instantly or whether it took several moves to figure out the correct solution. Further, what if a student used the trial and error method and just guessed blocks and tried submitting solution with trials on several incorrect moves. To bring a solution that caters to all these things, we bring different score metrics to Parsons.

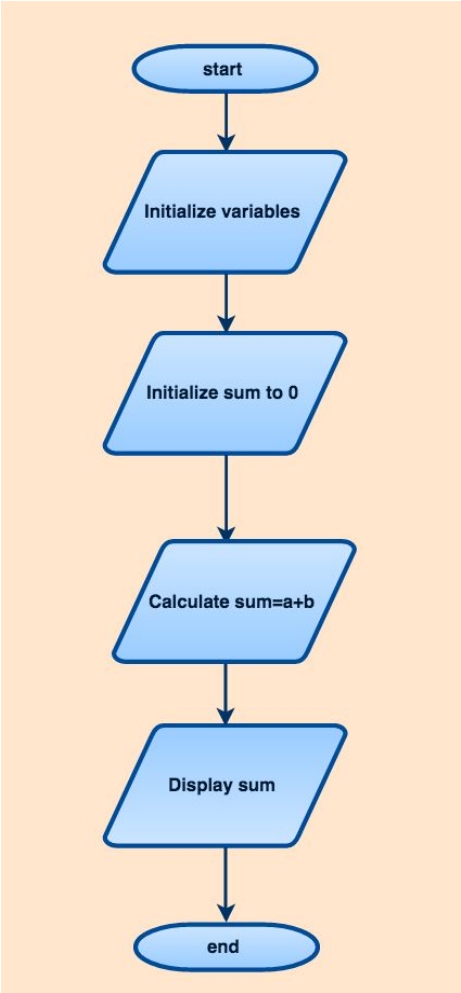


Figure 5: Traditional approach

4.2 Parson's in SAGE

Our implementation of Parson's uses the traditional Parsons and builds upon it to provide a gameful learning experience. The student uses the directed blocks from the Parson's palette as decided by the author by selecting blocks from the available palettes. The score against each block plays significant role in deciding the final score of the students. Also, we ensure every move of student is captured and with every move, we provide insightful feedback and guide student towards the solution. Details of this approach is provided in the later sections.

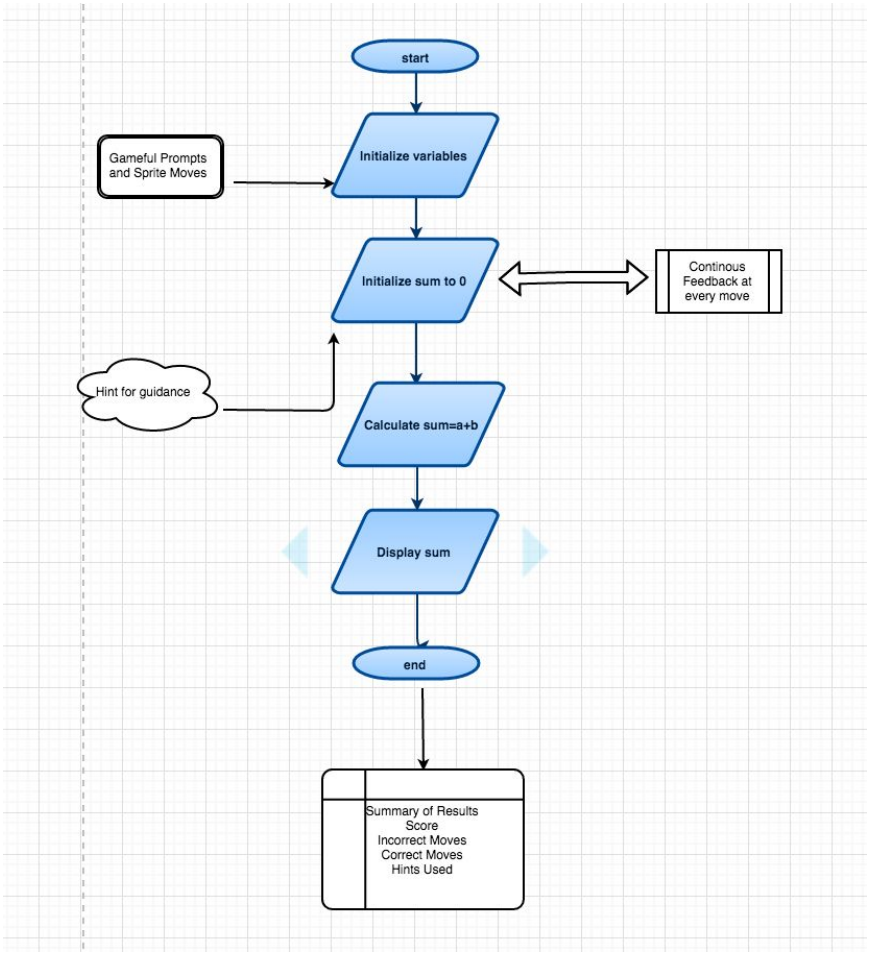


Figure 6: Parson's in SAGE

5.0 Implementation

The Scratch 2.0 editor is implemented using ActionScript and Flex. The code for the editor was written by the Lifelong Kindergarten Group at the MIT Media Lab and is publically available on GitHub under the GPL v2 license (MIT Lifelong Kindergarten Group, 2016). This Formative SAGE Assessment project extends the fork created by Jeff Bender (Bender, 2015).

5.1 Design Mode

5.1.1 Question/Hint

As soon as the SAGE platform is loaded, it is ensured that the Design mode opens if the assignment Id and Student Id is not entered. This provides the platform for the author/teacher to start designing the Parson's puzzle. This mode has an additional Parson's palette which will include blocks selected by the teacher to be available to the students to solve the puzzle.

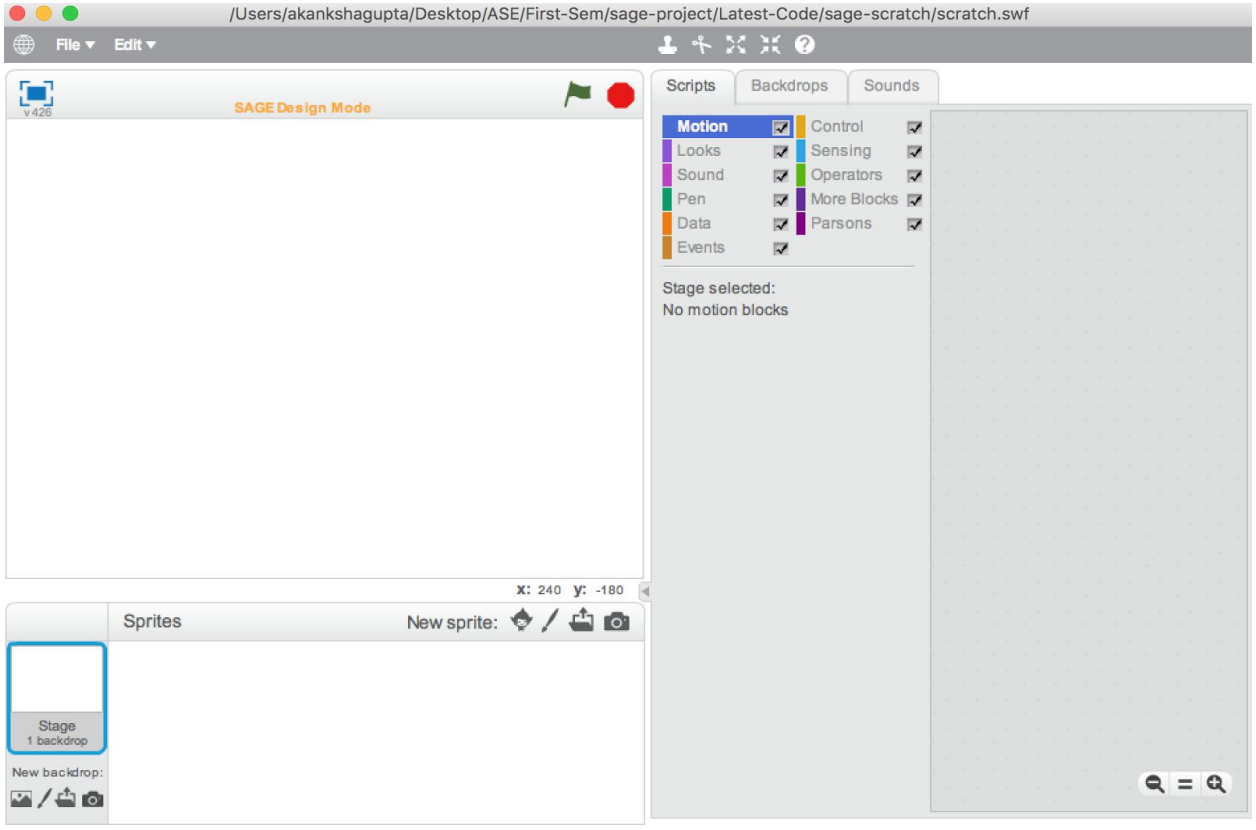


Figure 7: Design Mode

The blocks which are code snippets to solve a particular programming puzzle, are initially available in various categorized palettes and the Parson's Palette implemented is blank initially with only option to enter the "Question/Hint". This provides a way for teacher to share the problem statement with the student.

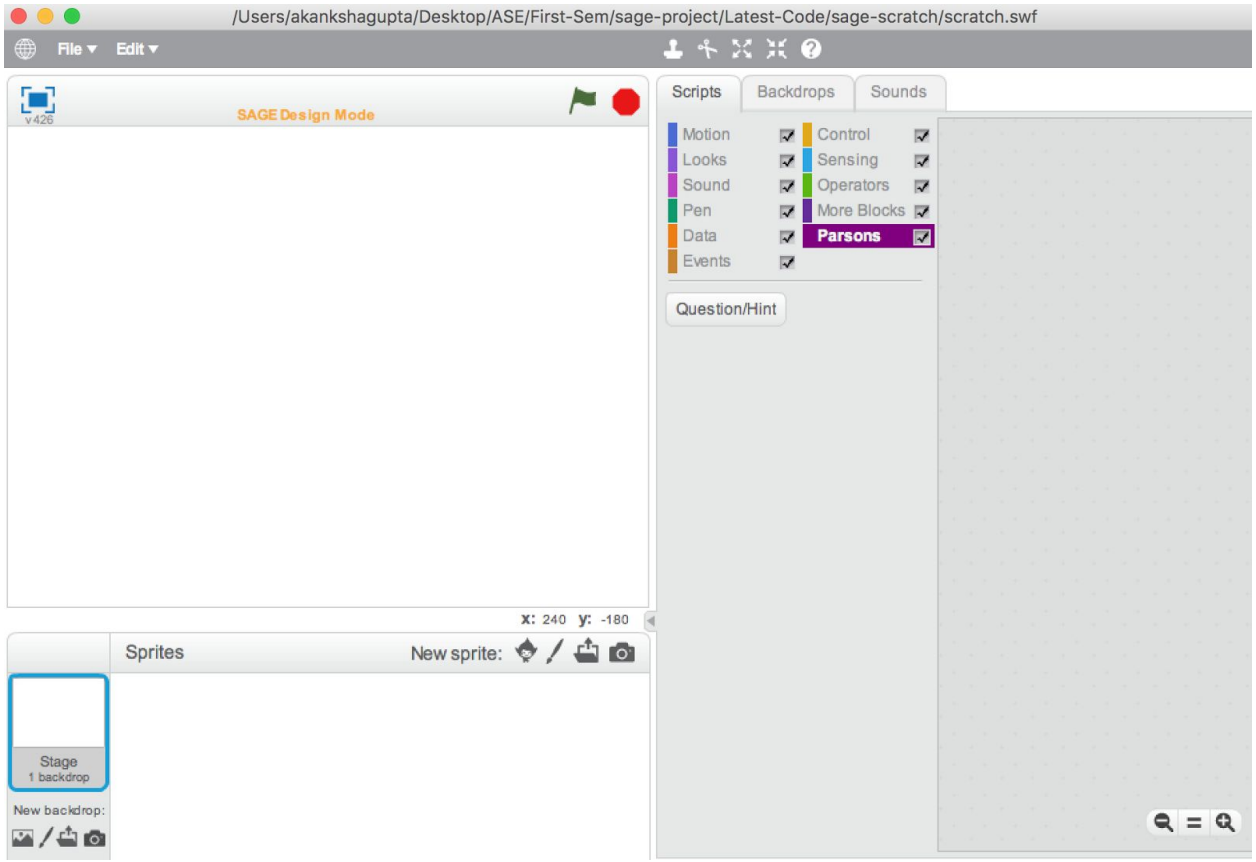


Figure 8: Parson's Palette (initially blank)

The teacher enters the question and the hint for the student. Also, the hint provided here is provided to the student only if the student demands for it. Every time the student uses a hint, the score of the student gets affected and is decremented. Also, we monitor the number of times the student uses the hint.The hint is used only when the student is struggling to get to the solution.

In our demo, the teacher designs a parsons puzzle to get the sum of first 'n' even numbers.So the author enters the relevant question and the hint for the student.

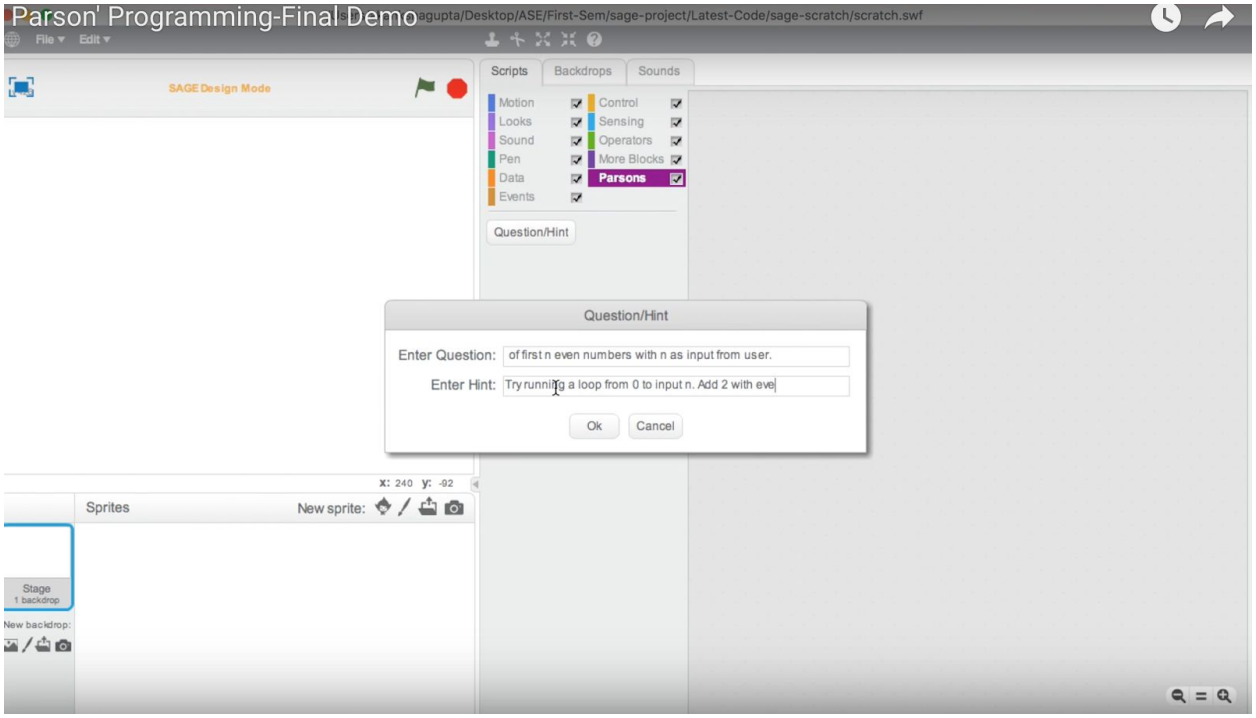
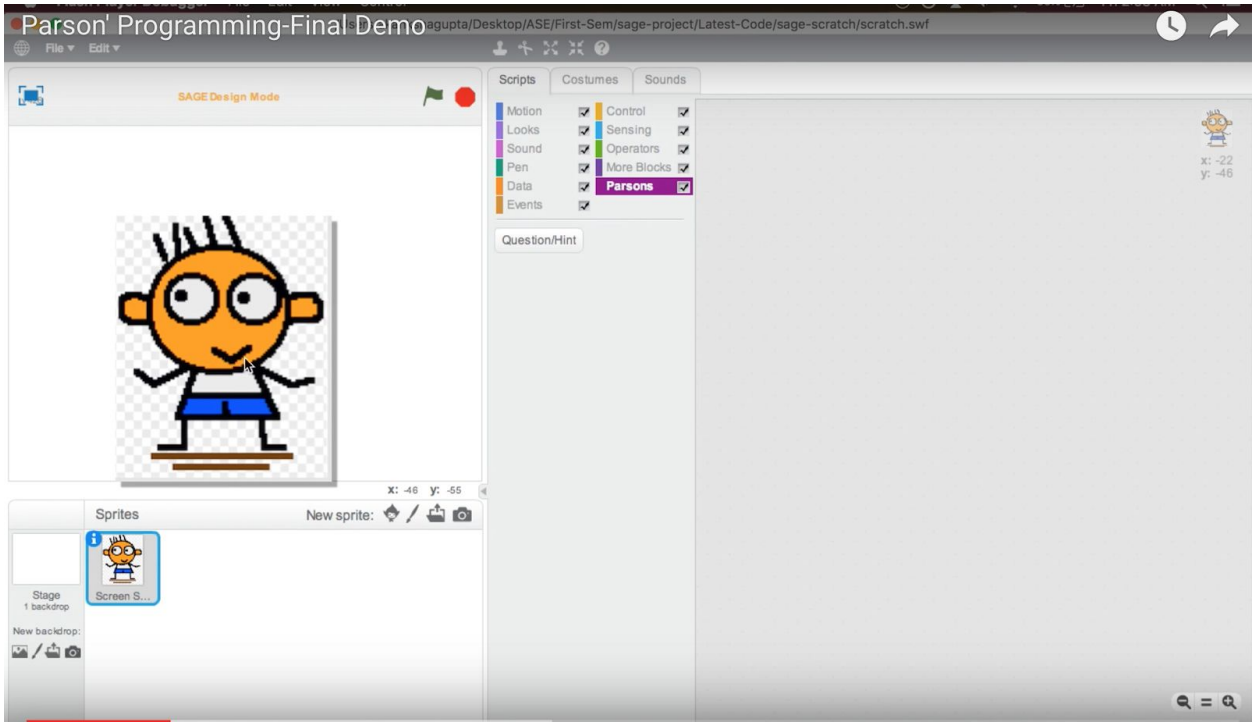


Figure 8: Question/Hint entered by the author

Next the author/teacher can attach a sprite to the custom block scripts that will be designed. This will provide a visual representation to the student and will be exciting and not mundane.



5.1.2 Designing Parson's Palette

Teacher/author now need to select the blocks which will become part of Parson's Palette. This can be achieved by selecting the check boxes against each block. Selecting them dynamically adds the selected blocks to the Parson's palette and creates the toolkit that will be provided to the student during puzzle solving. We ensure the students uses constraint logic to solve a puzzle which ensures that the student does not digress from the problem statement and is kept on track. This is achieved by using Blocks provided in Parson's palette. In the end, the student has to discern between correct logic and sound syntax to solve the puzzle. As we see in the below screenshot the teacher checks one of the block 'When Flag clicked' from Events palette.

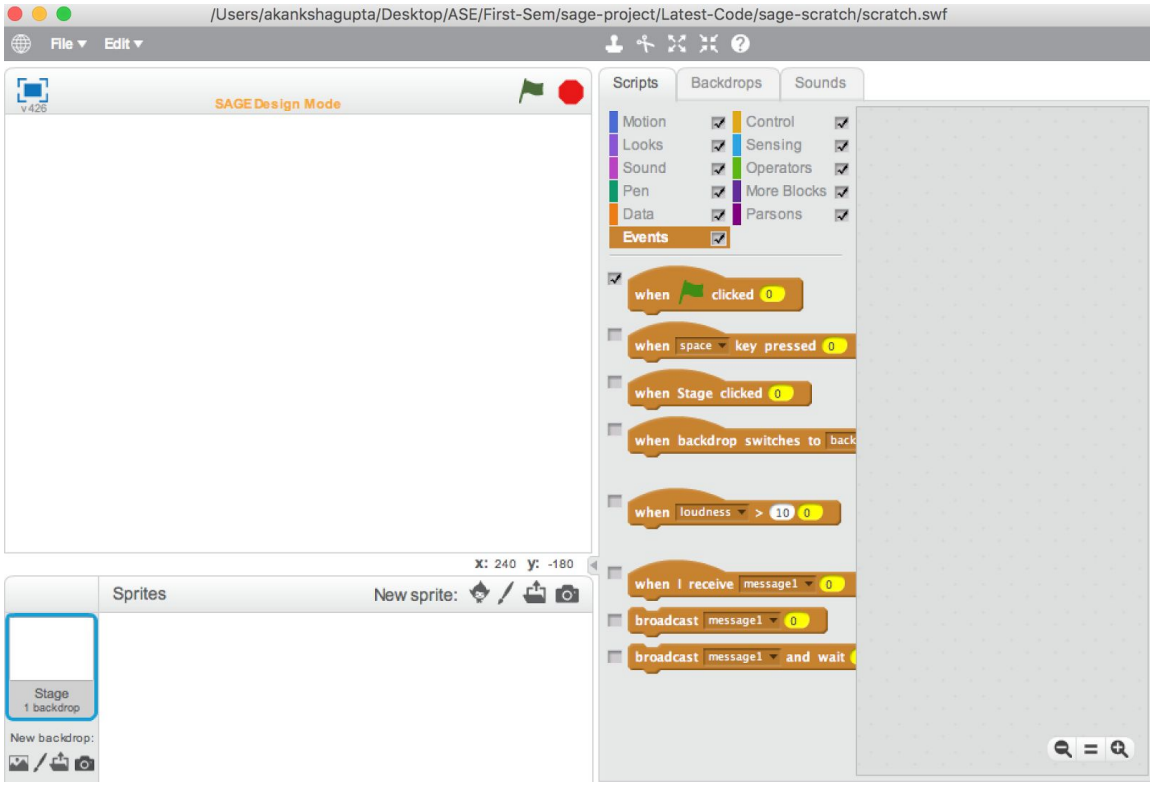


Figure 10: Selecting blocks from different palettes

The author/teacher can continue adding as many blocks as needed for the Parson's. Each block is allocated a set of score that signifies the importance of placing the block in correct order. It is these points that are used to calculate the score for the student

with every move. A correct move leads to incrementing points and a wrong move penalizes the student and decrements from the main score.

5.1.3 Distractors

We include the concept of distractors for the students. Distractors are unwanted blocks that are not part of the solution. Including distractors ensure the student discerns between the correct and incorrect blocks and then drags the desired block on the script pane. Distractors can be logical or syntactical error blocks or unwanted blocks that are not related to the problem. This ensures the student focuses on minute details. As in the below example, we see that the main script does not include the block ‘answer’(username) but it has been included in the Parson’s palette to be a distraction for the student.

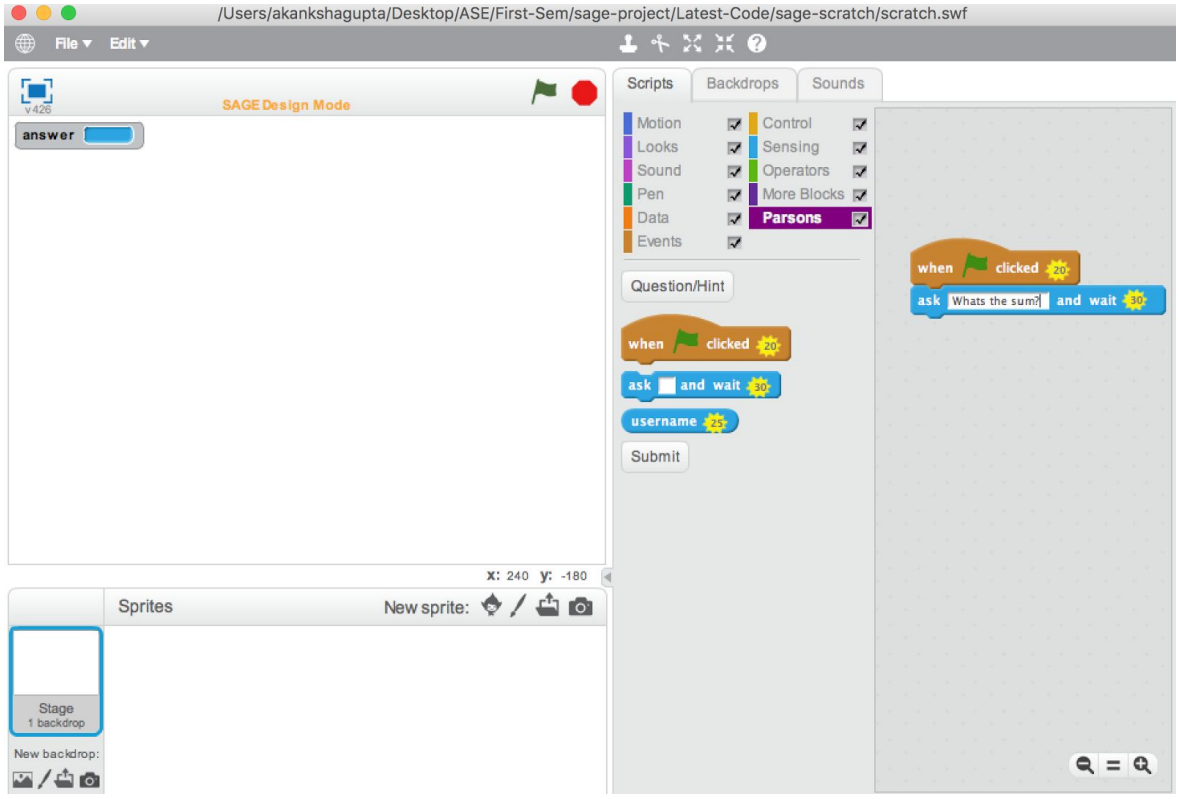


Figure 12: Distractors

5.1.4 Removing unused palettes

Further, before the submission of the assignment, the teacher/author excludes the palettes that are not to be considered at all for solving a particular Parson's puzzle. This is in accordance to guided learning and ensuring student uses the blocks from the palette that is supposed to be used. Teacher need to uncheck the checkbox against the palette to disable all the blocks contained in it. There's also a functionality to exclude blocks within the palette. Once excluded those blocks won't appear in Parson's Palette. Blocks appear in Parson's palette only when they are checked and the corresponding palette is also checked. In the below screenshot, 'Pen' palette is unchecked by the teacher to ensure that students don't use blocks from this palette to solve the puzzle.

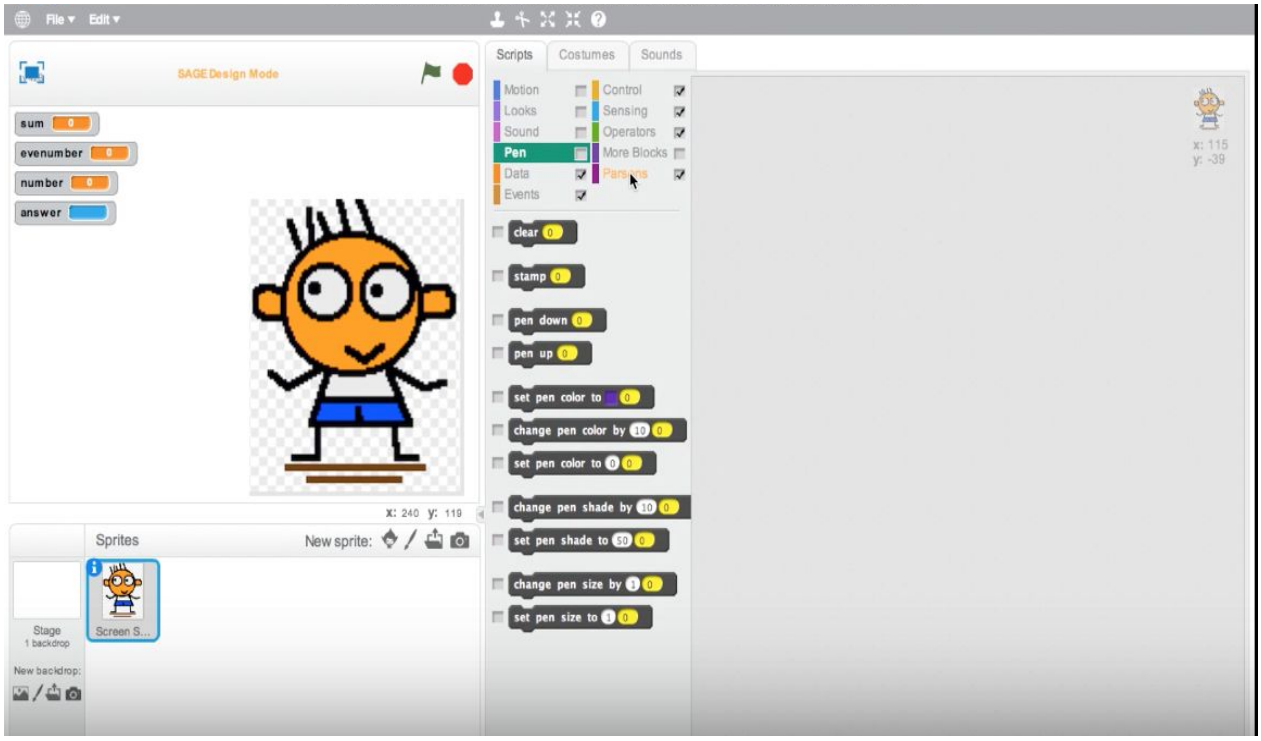


Figure 13: Uncheck unused palettes

5.1.5 Designing Parson's Puzzle

Once the Parson's palette is finalised, teacher now designs the puzzle using the blocks available in parson's palette. Below screenshot shows the completed parson's puzzle for sum upto 'N' even numbers as authored by teacher.

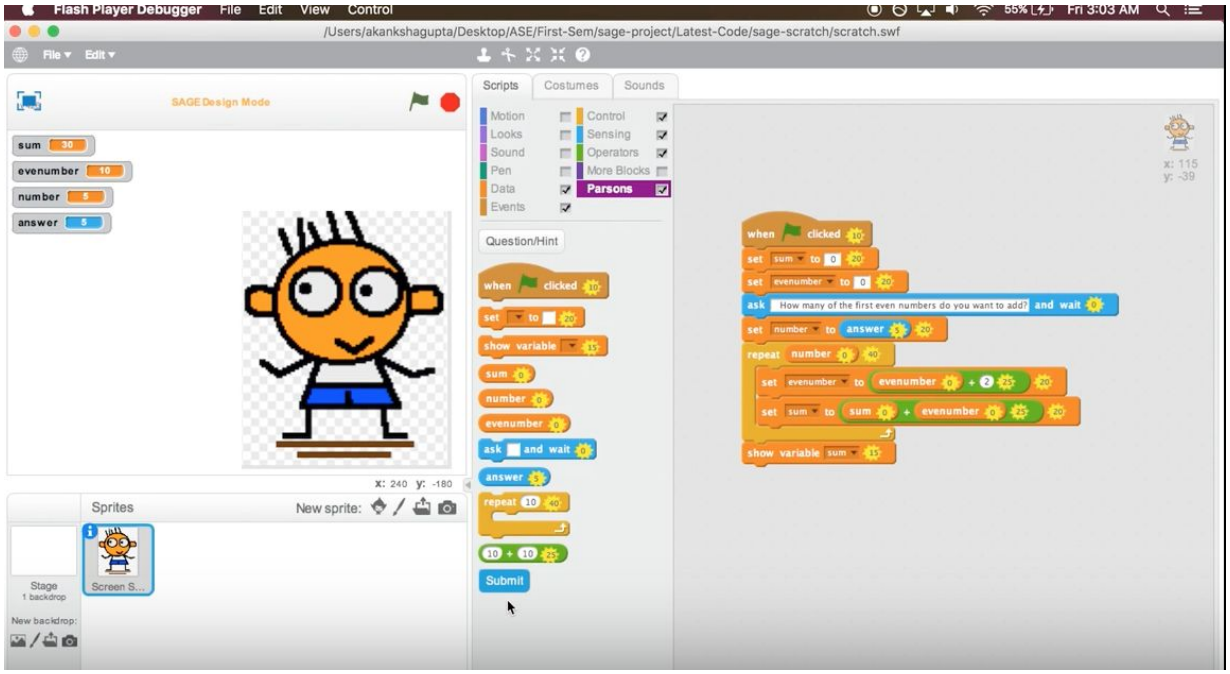


Figure 14: Parson's puzzle

5.1.6 Submit Project

Once the problem is designed, the teacher/author saves the project and the assignment is created and saved successfully. This is done by clicking on the submit button as shown in screen below. On click of Submit, one can choose the name of project and save it.

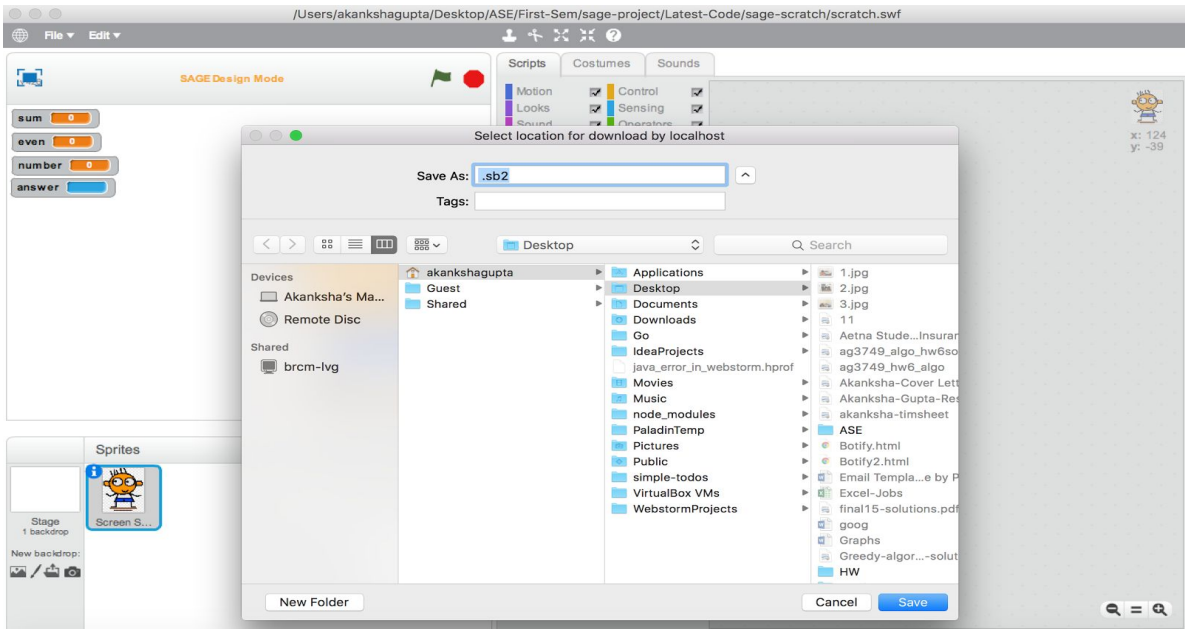


Figure 15: Submit Project

5.2 Play Mode

5.2.1 Student Login

When the student logins to SAGE, he/she enters the Student Id and the Assignment Id. This opens the SAGE in the play mode to be used by student to solve the Parson's puzzle.

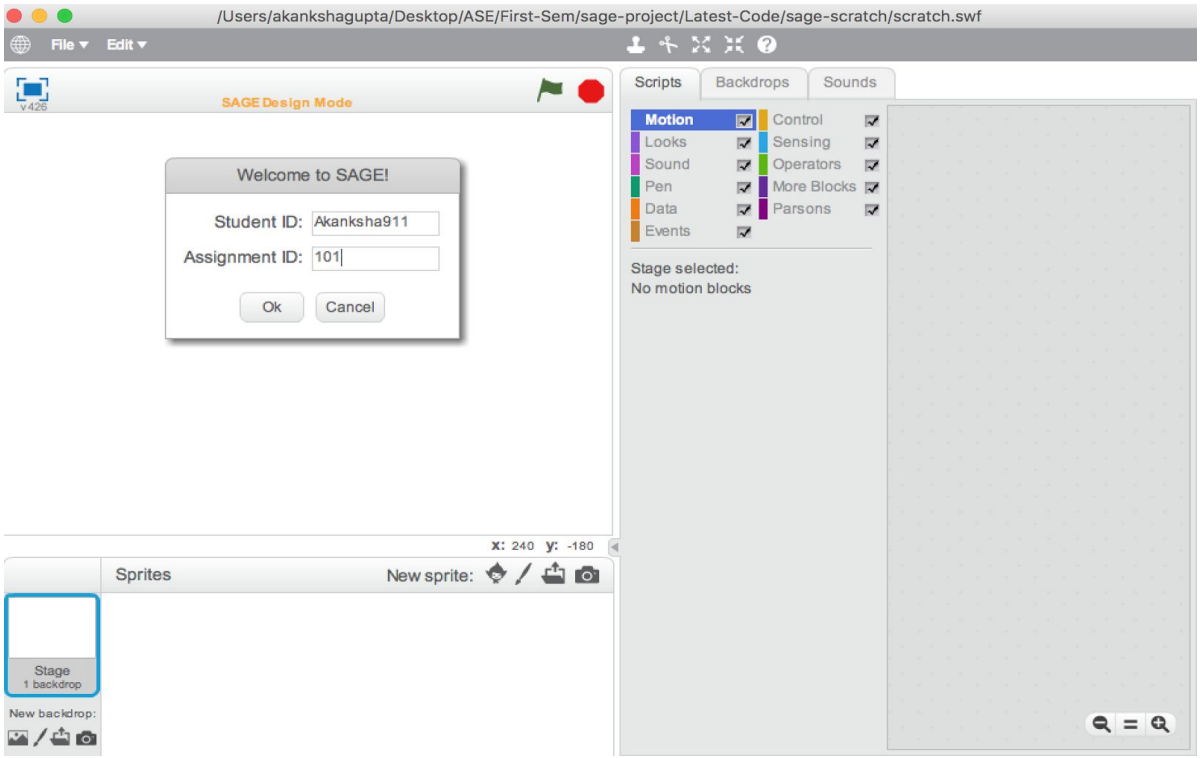


Figure 16: Student Login Page

5.2.2 Load Project

The landing page has initial score as 0 and a message 'Hey, Let's start solving' in the Play Mode. Next the student loads the Parsons project shared by their teacher/professor. As soon the project gets loaded and the student clicks on the Parson's palette, he/she can see the blocks to be used to solve the problem. This reduces the cognitive load for the student by not writing the code lines, but rather understanding and focussing on usage, logic and syntax. Below screenshot displays the Parson's palette as shown to the student in play mode.

Figure 17: Parson's Palette in Play mode

5.2.3 Puzzle Solving

The student begins solving the puzzle by first clicking the 'Question' to read the problem statement.

Figure 18: Question seen by student in Play mode

Next the student starts to solve the Parson's puzzle by dragging and dropping the blocks. As soon as the first correct block is dropped in the script pane, we see a corresponding change in the score as well as the message (‘Good Going’) as to how we are proceeding with the puzzle. Here we see 10 points get added to the score as the dragged block had 10 points allocated to it.

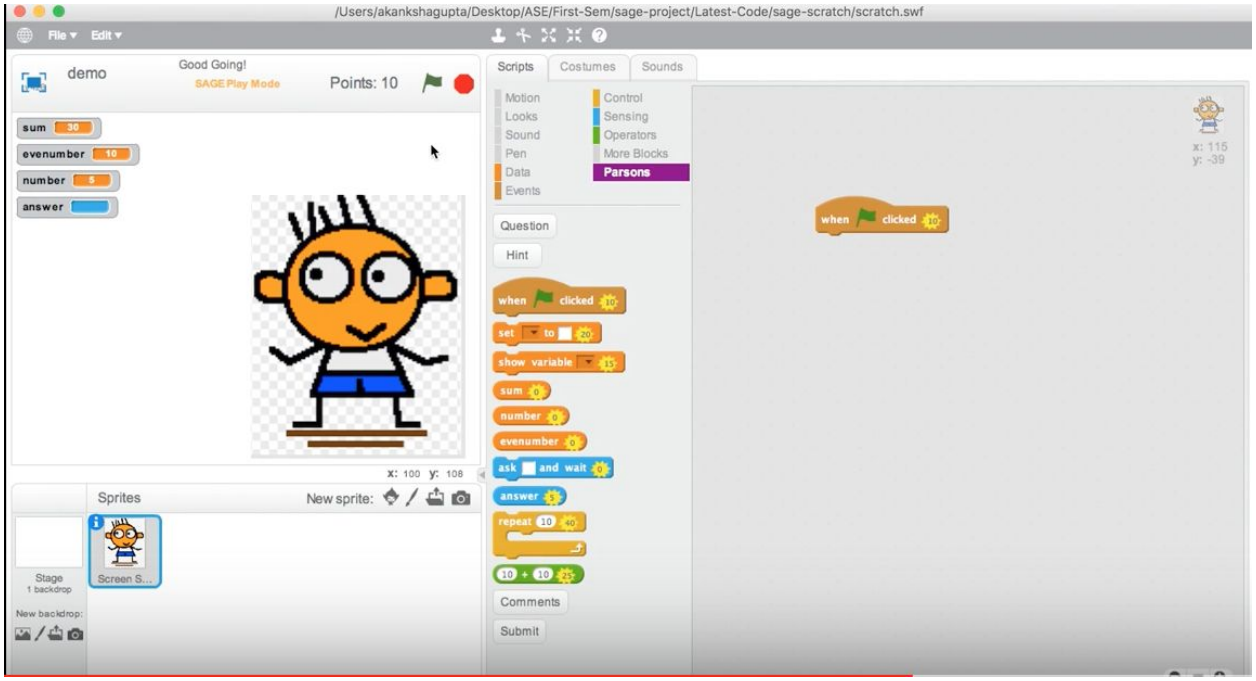


Figure 19: Points/Message when correct block is dragged to scripts pane

Similarly, If an incorrect block is dragged into the pane, we see a deduction in points. As an example, the repeat block with 40 points attached to it is dropped in script pane and in incorrect order, the total score becomes 10-40=-30. Also the puzzle prompts that *Oops, Wrong choice* that will immediately alert the student about the wrong move. This is in line with the Parson’s concept of penalising student for wrong moves which actually guides students towards the correct solution.

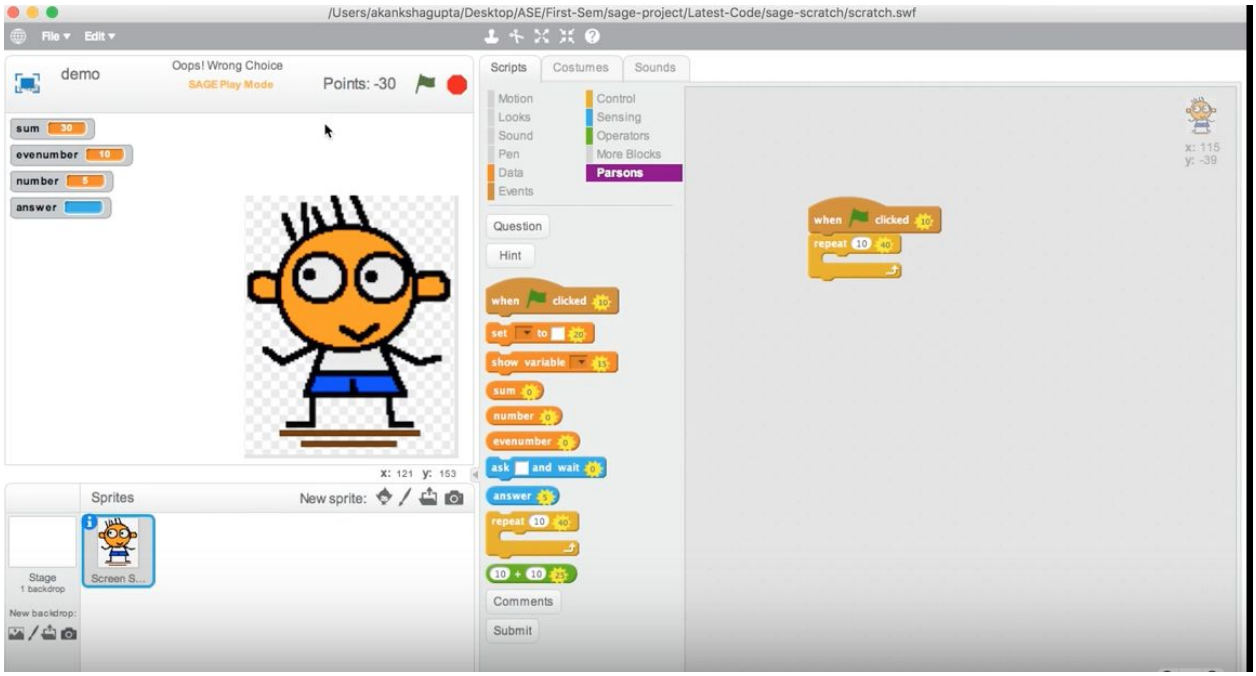


Figure 20: Points/Message when correct block is dragged to scripts pane

Next, the student can use the hint option to get guidance towards the right solution.

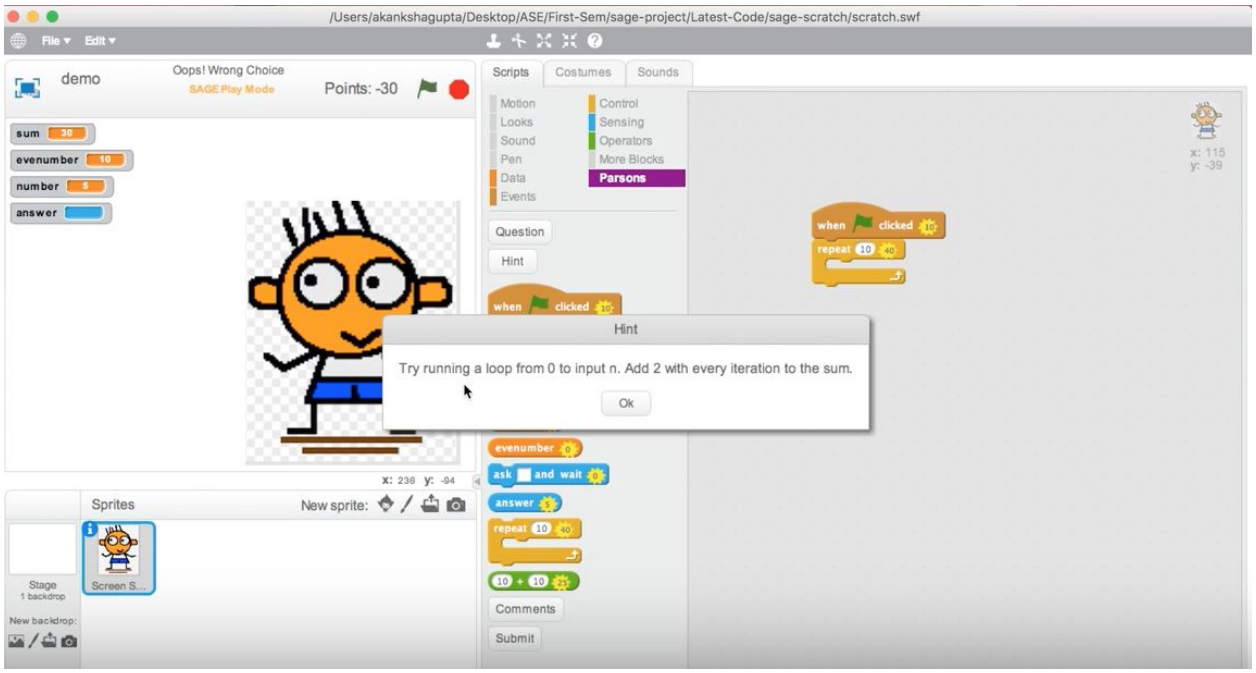


Figure 21: Hint as seen by student in Play mode

Another interesting score metric we have implemented is to keep a count of incorrect moves. Say in the above example, current points are -40. Now, after checking the hint, the student realizes that the block was incorrect and so moves the block back to the palette. This does not grant back the 10 points he/she had earlier, but instead results in score of 7 as shown below. This has been implemented to differentiate between students who got the solution correct in one go without any incorrect move or using the hint option at all to student who made an incorrect move and then used hint to get back to track. Hence, -1 for using the hint and moving repeat block to and fro from palette to script pane and back resulted in -2. Hence total score became +7. Such a scenario will help identify students who solved puzzles in one go to students who did trial or error or made some incorrect moves before reaching the final solution.

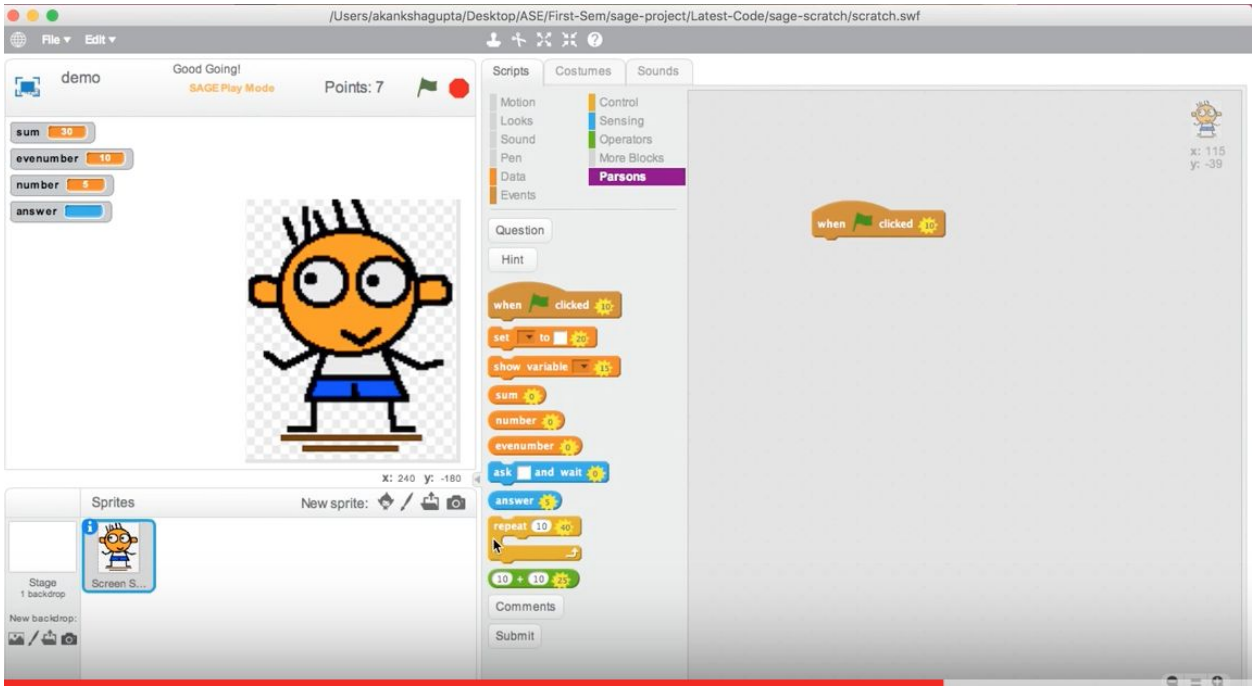


Figure 22: Score taking in account incorrect moves

Also, another scenario that has been taken care of is the scenario where the student randomly places the block from palette to script pane, but doesn't connect them in any order to rest of the blocks, Since the placement of that block makes no sense to solving the solution, marks are deducted and at the same time, the system prompts the

message to try removing some blocks, hence again guided towards the right solution set.

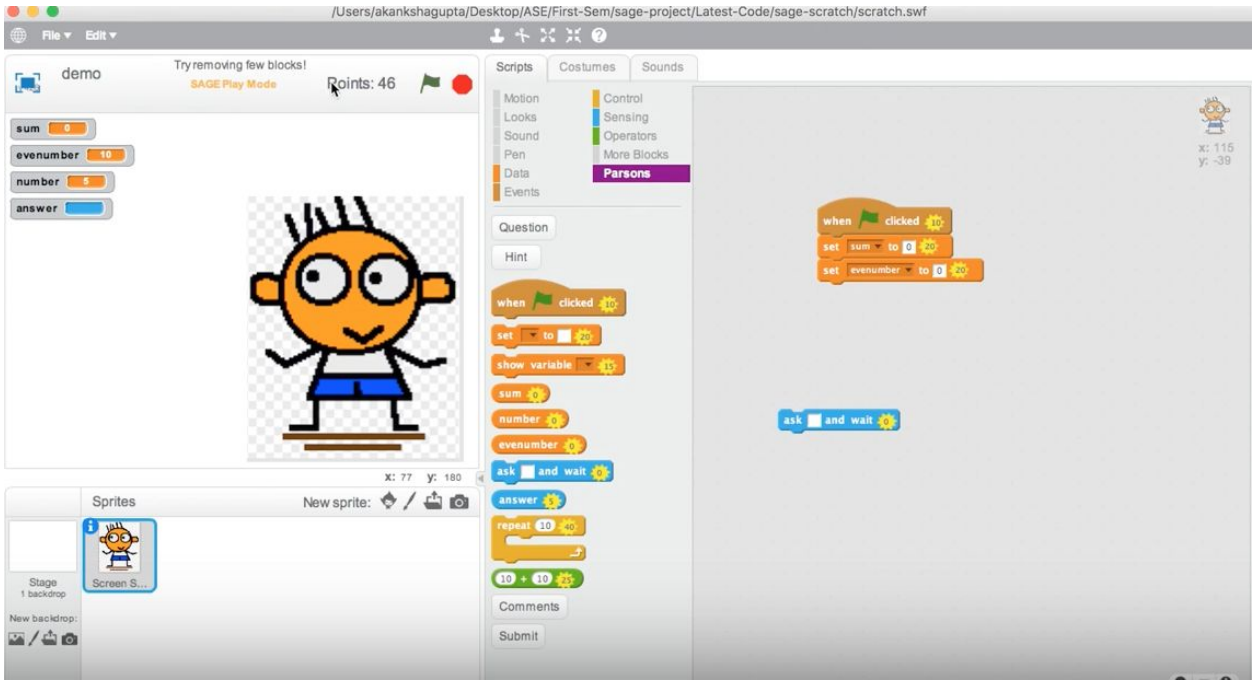


Figure 22: Score taking account of irrelevant moves

The student designs the puzzle and then runs it to confirm the answer.

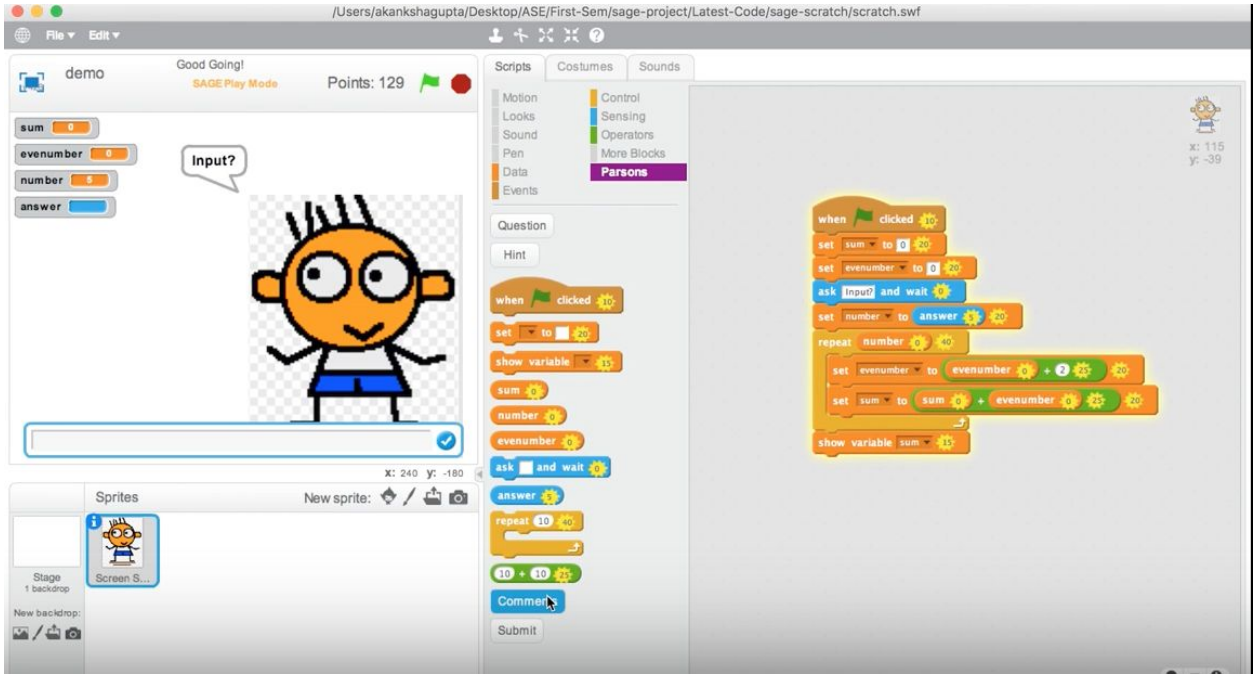


Figure 23: Final Puzzle solution

Also the student is provided with an option to provide Comments and feedback which the teacher can use to learn more about the experience of the student and improve the puzzle.

Figure 24: Final Puzzle solution

On successful completion and submission, the student sees the summary for the result and what went correct and what went wrong.

Figure 25: Summary

5.3 Score Metrics for Parson's

Student is awarded points based on various scenarios.

S.No.	Move	Score	Comments
1.	Block moved in correct order to correct location on script pane	Points incremented based on points associated with the moved block	
2.	Block moved and attached in incorrect location in script pane	Points decremented based on points associated with the moved block	
3.	Hint clicked	Every time hint is used, -1 from the score	
4.	Incorrect moves	-1 for every incorrect move	This is implemented to ensure student does not apply trial and error method to reach the correct solution

5.4 Project JSON

Project json is altered to send Parson's palette information to the Play mode.

6.0 Code Repository

GitHub (Github, 2016) was used as the code repository for the coding for Parson's on Scratch Editor. We created a branch forked from from Sage(Jeff Bender,2015 and Jairo Pava, 2016).

Link to GitHub:

https://github.com/cu-sage/sage-scratch/tree/parsons

Wiki Link 1

https://github.com/cu-sage/sage-scratch/wiki/Parson's-Programming

Wiki Link 2

https://github.com/cu-sage/sage-scratch/wiki/Score-Metrics:-Parsons-Programming

7.0 Future Work

There remains plenty of future work to make the Formative SAGE Assessment with Parsons an excellent platform for computational thinking in the classroom with Parsons. A study should evaluate the responses and experience of teachers as game designers on how they feel the Sage Parson platform to design Parsons puzzles. Ease of creation of puzzles should be evaluated. Students should be asked to solve the puzzles and learn more about their strengths and weaknesses for particular concepts.

The final score summary results and score metrics should be evaluated to learn more about ways students solve the puzzles and differentiate between students solving puzzles in one go flawlessly and those that made mistakes and learnt during puzzle solving. It should also be evaluated for how well it is able to get students to focus on specific computational thinking concepts that the teacher is targeting through the puzzles.

Considering the puzzles, one enhancement that can be worked on is the ability for the teacher to submit multiple solutions to the same problem. This way, the teacher can expose students different ways of solving the same puzzle and also teach differences between naive approaches and optimized approaches. This can be combined with different marking schemes for the different solutions.

Further a review option should be made available to students once they are done with the submission of the puzzle. This will empower them to learn different solutions that the teacher provides and learn from them and try them on the Sage Parson platform. This can further provide personalized recommendations to the students instead of general pre-decided recommendations to the students that teacher configures during the puzzle creation.
Further, the puzzles can be extended by including difficulty levels likes Easy, Medium and Hard while practicing Parson's puzzles and learn at different levels at which they are comfortable playing.

8.0 Conclusion

The SAGE Parsons Project was presented in this report. It empowers teachers as game designers to design Parson's puzzles based on the drag and drop to correct locations for the students using computational game thinking skills. The teacher designs the complete Parson's palette dynamically and exposes the blocks he/she wants to expose to the students to solve a particular puzzle, ensuring guided learning. This enables the students to focus only on particular blocks and removes the load of writing code from the scratch.

Further the students can learn from the puzzles. Distractors are kept in the palette to teach students to discern between logical and syntactical errors. It also enables students to focus on the specific goals of the puzzles created by their teachers. The students get real-time feedback on each move they make in the script pane. Further score changes according to the moves and a defined score metrics. Once the puzzle is submitted, the final summary of results is shown detailing about the score, number of correct moves, incorrect moves and if hint was used or not.

There is much work left to be done for this project. An evaluation of the project should be conducted in a classroom environment where its strengths may be built upon and its weaknesses addressed. Future work should aim towards further analysis of how the puzzles can be designed in this project can be improved to better capture a student's computational thinking progress over time. Also, review of solution and introducing difficulty levels in puzzles would be a good to have feature.

9.0 References

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

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.

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

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.

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*.

Danielle S. Mc.Namara , Ville Karavirta (2011). *Two-Dimensional Parson's Puzzles: The Concept, Tools, and First Observations.* Aalto University, Finland

Danielle S. Mc.Namara , G. Tanner Jackson, Art Graesser. *Intelligent Tutoring and Games (ITag) .* Aalto University, Finland

Adobe. (2016, May). *Adobe*. Retrieved May 2016, from Adobe Flash Player: https://get.adobe.com/flashplayer/

What is the Role of the Computer Science Education Community? ACM InRoads.
Bender, J. (2015). *Developing a Collaborative Game-Based Learning System to Infuse*

Github. (2016, May). Retrieved from Github: https://www.github.com

MIT Lifelong Kindergarten Group. (2016, May). Retrieved from Scratch - Image, Program, Share: https://scratch.mit.edu/