Formative SAGE Assessments with Parson's Programming

Proposal By: Akanksha Gupta and Sudhanshu Mohan

COMS E 6901, Section 14

Fall 2016

1. 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.

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.

In the past few years, a lot of interest has gone towards understanding how people think. Researchers have focused on design based learning activities using interactive media to keep students engaged. People have focused on honing the computer programming skills of students for introductory Computer Science courses as well as advanced courses.This has risen interest in Parson's Programming puzzles. Parson's programming puzzles are a fun and interactive way that allows teachers/tutors to touch upon important topics and syntactic and logical errors in a program. Such puzzles expose students to short pieces of code that help students construct the logical order of statements for any Parson coding puzzle. *S*ince each puzzle solution is a complete sample of well-written code, use of the tool exposes students to good programming practices.

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. At the same time, the intent of the teacher is to help the student and guide him/her through the solution and reach the final answer. This 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.

We propose to include the concept of Parson's programming puzzles for SAGE. SAGE (Bender, 2015) extends Scratch, which is a programming language and community that allows teachers and tutors to author gameful puzzles for students. The language is drag and drop based that provides functionalities of media, stories, animation and much more for authoring puzzles (shown in Figure 1).The Scratch community is very active and has millions of users across the globe. It's a powerful learning tool that has wide audience using it. SAGE extends the idea further by proving game based-learning techniques and instills motivation for computational thinking at an early age. SAGE takes advantage of the collaborative nature of the online Scratch community and traditional classroom environments to encourage students to

compete with each other while playing the games and, most importantly, learning important concepts. Using, SAGE the teacher drives the learning for the student.

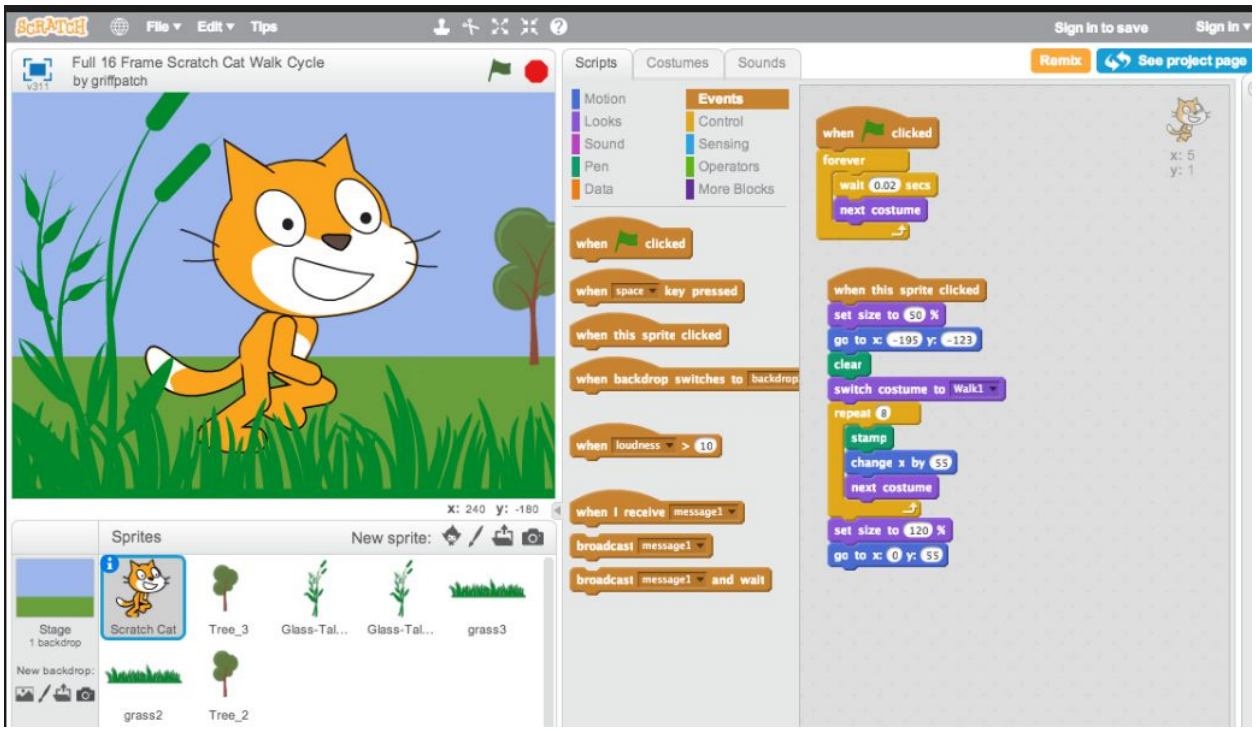

Figure 1

SAGE extends Scratch by empowering teachers to play the role of game designers. This enables them to create games for students to play as they learn computational thinking concepts. Since games are fun and engaging for young students, SAGE leverages game-based learning techniques to instill intrinsic motivation for computational thinking. The teacher designs games and students play the game by building Scratch programs to solve objectives. Students compete to perform better and actively learn concepts. The difficulty level of the puzzles can be increased by combining two or more concepts together. It can be made interactive and interesting by having sprites. Sprites, either user-created, uploaded, or found in the sprites library, are the objects that perform actions in a project. A snapshot of the present SAGE Editor is shown in Figure 3.

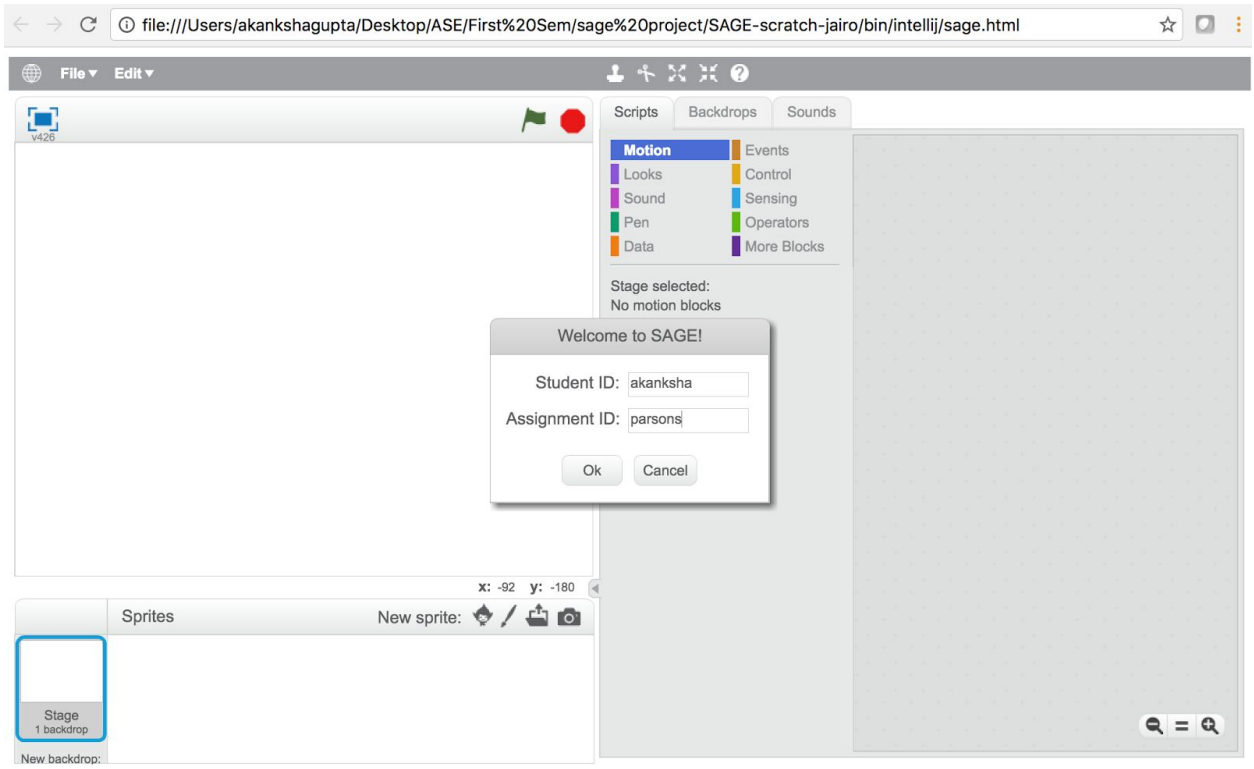

Figure 3

The proposed project will therefore provide the option for teachers to author Parson's puzzles which can then be solved by students. The students will have a timer to time bound the time taken to solve Parson's puzzles. The timer will help the tutor to assess the child. Further the teacher will be given an option to embed video tutorial related to the puzzle to help the student. The assessment provided will be real time, with actions tracked as sprites move. Teachers can increase the difficulty level by using the block and template restriction functionality. Also, we will provide an option for students to leave comments and feedback and share their solving experience with the teacher. This will provide a platform for students to communicate their thought process with the teacher who assess them.

2. Related Work

2.1 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 4). 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.

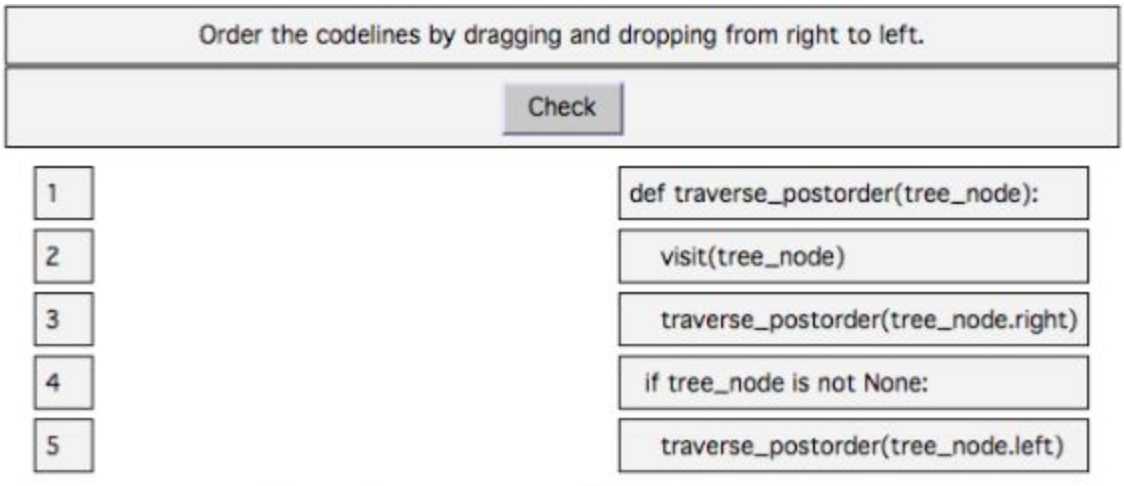

Figure 4: A Parson's problem in Hot Potatoes

2.2 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 5.

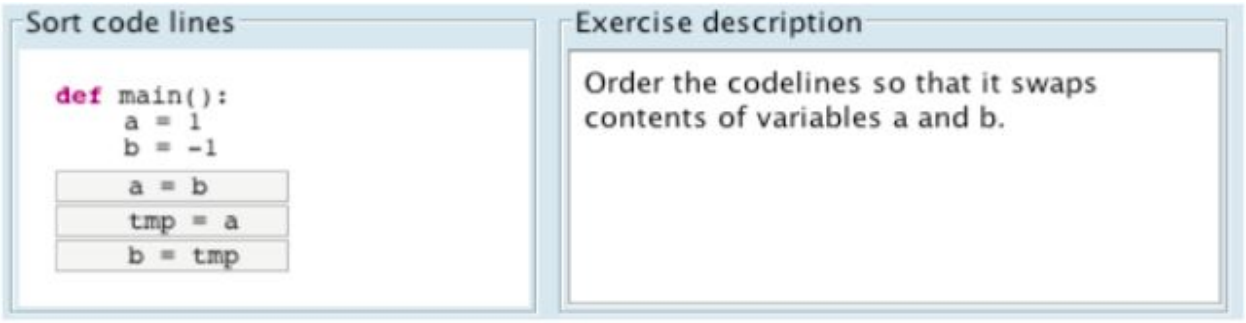

Figure 5: ViLLE

2.3 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.

2.4 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.

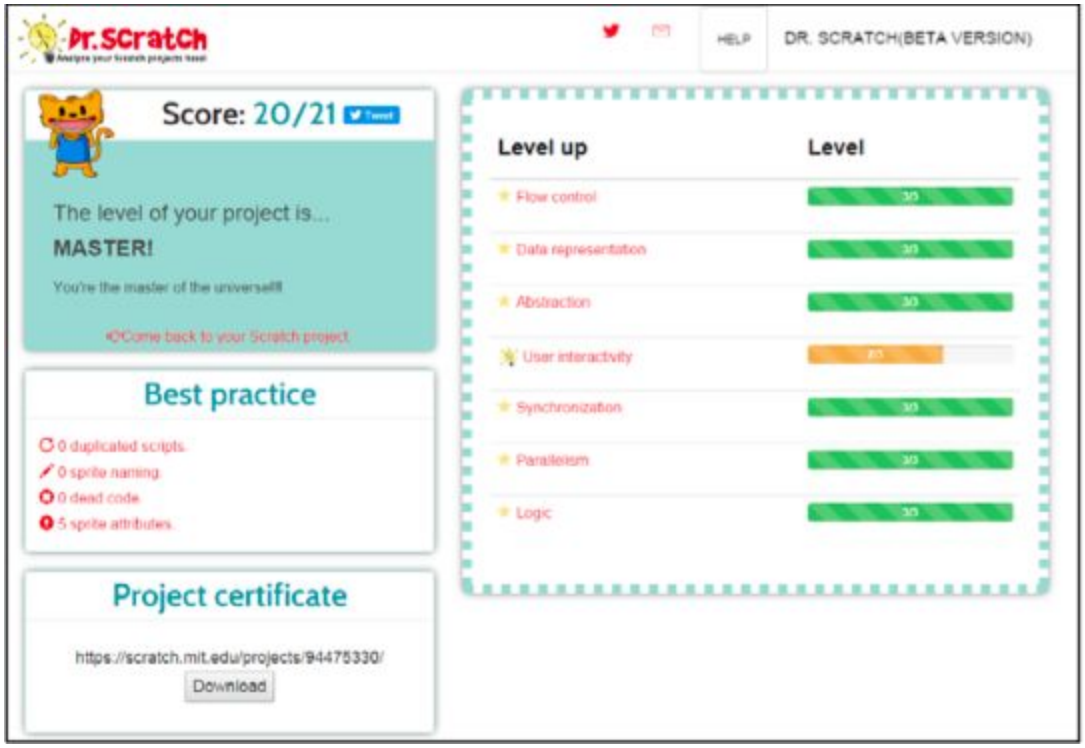

Figure 6: Dr.Scratch scorecard

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.

3. Proposal

The proposed solution will add the functionality for the teacher/tutor to design and author Parson's programming puzzle for students. There will be a separate palette for Parson's programming which the teacher can select and start authoring the Parson's puzzles. All other palettes will become inactive on selection of Parson's palette that can guide the student through the block pieces to use to get the solution. Also, when the student authors the solution, our system would randomize the lines and present the randomized lines of code to the student as a distractor. There will be a hint option which the student can leverage in case he/she doesn't understand the way to solve the problem. Illustration of the proposed interface can be seen below in Figure 7. Also, we will have a marking scheme based on the number of lines of code placed in correct placeholder. As an example, the student will get +2 for every line placed at the correct position and -1 for every line placed at incorrect position. The cumulative result will be displayed to the teacher along with the fact whether the student used the hint or not. The timer provided will help the student to track his/her progress and submit solution on time without getting distracted.

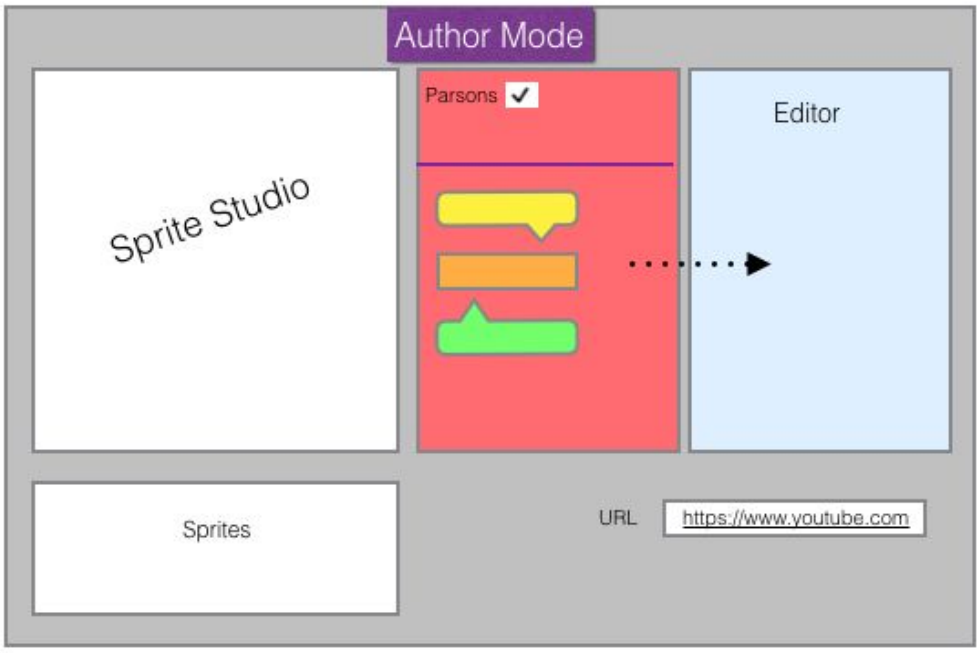

Figure 7: Proposed Author Mode in SAGE

Author Mode is for the teachers to design puzzles for the students. It'll have a Parson's palette which will contain programming related blocks. Teacher can drag and drop blocks in the editor. There will be a URL section to embed a link of the video tutorial. Teacher can also enable/disable templates as well as blocks in this view.

In addition, as soon as student starts ordering the lines for the solution, he/she can see corresponding sprites move in the left pane. This will be interactive for the student and will give a gameful experience to the student. Also, we plan to provide a feedback/comments section for the students where the students can write a note to the teacher, share their experience while solving a problem and also share their thought process. Illustration of the proposed interface is shown in Figure 8.

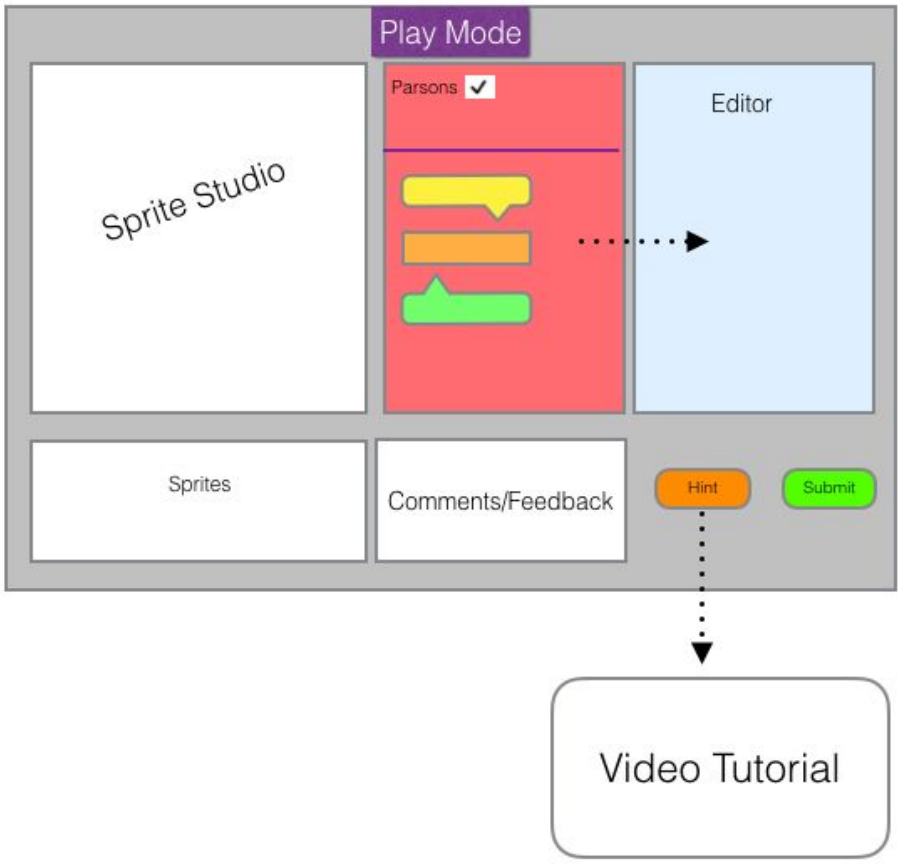

Figure 8: Proposed Play Mode in SAGE

This will help the teacher better assess the student and even take into consideration if a student has any concerns. The final reports will have marking based on the number of lines placed in correct order by the student, time taken to solve the puzzle and also if the student used the hint or not. Final report will be graphical , showing the students focus and how many puzzles were solved correctly with a final score. A database will be used to persist the results of an assignment's evaluation over time to enable the ability to mine the data for further insight as part of future work.

3.1 Parson's Palette

There will be a separate palette for Parson's programming. On selecting the Parson's palette, the teacher will get the option to author Parson's puzzle. The teacher can drag and drop the blocks that will have code to generate small puzzles. We need to create new logical blocks(assignments, loops, conditional) to facilitate the problem solving techniques. A sample palette is shown in Figure 9.

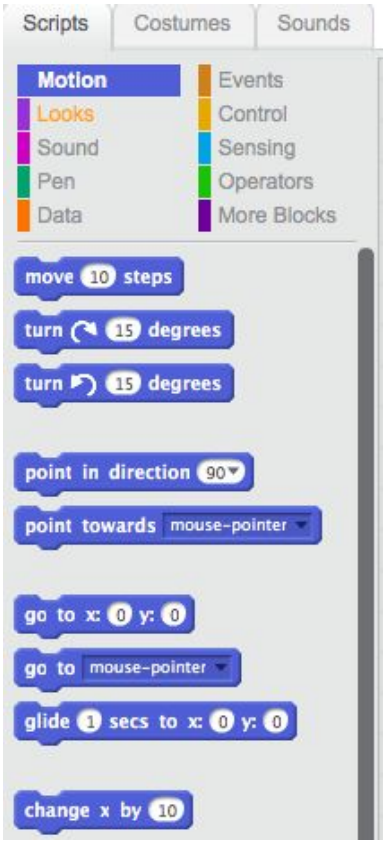

Figure 9

3.2 Palette restriction

On selecting the Parson's palette, the teacher will get the option to author Parson's puzzle. All other palettes will become inactive on selection of Parson's palette. This will be a palette restriction that will be implemented so that student focuses only on those blocks of code which the teacher wants the student to focus. There will be an option to selectively enable/disable palettes which will increase the difficulty level for the students (shown in Figure 10). Also, there will be option given to the teachers in author mode to selectively disable block in a particular palette. Once excluded, the block will be greyed out for the students in play mode.

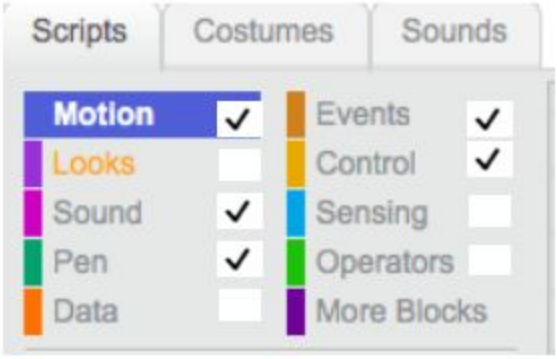

Figure 10

3.3 Interactive Sprites

As the student orders the lines of code, sprites will move correspondingly in the left pane and show the progress of student. Such interactive sprites will give hints to the student and also be gameful for the student. This will ensure the student is not distracted and has active learning , enjoying the puzzle.

Figure 13

3.4 Timer

Further there will be a timer in the play mode. This will help the teacher track the time taken by student to solve the problem. This result will also help assess the student and will have a marking scheme taking time taken by student to solve the puzzle into consideration. This will also help the student as well to time himself and focus more on problem and not get distracted.

Figure 11

3.5 Report Card

The results on the dashboard will indicate to the student which evaluations have passed successfully and which evaluations are not passing. For those evaluations that are not passing, the student will need to continue to work. The dashboard will, in cases where it is appropriate, display suggestions that will help the student work towards successful evaluations.

4. Milestones

Table 1 outlines the various milestones and their estimated completion dates. Dates are also defined for midterm and final project deliverables.

S.No	Milestone	Date
1.	Project Proposal	September 30, 2016
2.	Parson's Palette	October 26, 2016
3.	Palette Restriction	November 4, 2016
4.	Interactive Sprites	November 11, 2016
5.	Timer	November 18, 2016
6.	Report Card	November 25, 2016
7.	Testing and Stabilization	December 9, 2016
8.	Final Presentation	December 16, 2016 (Tentative)

Table 1: Proposed Milestones completion dates

5. Future Work

The future work would be to add two difficulty levels for solving the Parson's puzzle. One will be easy and the other will be advanced level. The advanced level will have the option for teacher to add distractors within similar piece of code. As an example, there will be two statements that look similar but might differ as one will have a logical/syntax error.

Int a=0,b=1; // correct version

Int a=0,,,b=1; // incorrect version-distractor

So the student chooses between similar statements and then solve the puzzles. Further the user interface can be made more user interactive. The option for students to see talking avatars will be interactive as they log into the system.

The design of the proposed system will provide the flexibility to collect and analyze data for future study and apply analytics on them. For example, the test server will maintain a history of assignment evaluations. This data may provide useful insights that can be mined to provide students with personalized recommendations. For example, consider a student that is currently struggling on an assignment. This student may exhibit a pattern of passed and failed automated evaluations that is similar to other students in the past who have worked on the same assignment. A recommendation can then be made to the student based on the path to success that past students have taken. The mining strategies, automated evaluations, and kinds of recommendations may be the subject of study for future work.

6. References

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

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

(2016, September 29). Retrived from Dr.Scratch - Analyse your Scratch projects here: http://drscratch.programamos.es

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.

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.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint- inspired Static Analysis of Scratch Projects. *SIGCSE.* Denver: ACM.

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*.

Murphy, C., Kaiser, G., Loveland, K., & Hasan, S. (2009). Retina: Helping Students and Instructors Based on Observed Programming Activities. *SIGCSE* (pp. 178-182). Chattanooga: ACM.