Gameful Direct Instruction (Parson’s Puzzle)

Sudhanshu Mohan

COMS E 6901, Section 28

Fall 2017

Table of Contents

1. Introduction

Computational thinking has received considerable attention over the past several years. We are particularly interested in the ways design-based learning activities like programming interactive media would support the development of computational thinking in students. Computational thinking involves three major dimensions: (1) computational concepts, (2) computational practices, and (3) computational perspectives (Computational Thinking with Scratch, Harvard GSE). It is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science.

As students design interactive media with Scratch, they engage with a set of computational concepts such as sequence, loops, conditionals, operators etc. that are common in many programming languages (Barr & Stephenson, 2011). Such concepts foster critical thinking in students at an early age.

The next step in developing computational thinking framework is to describe the processes of construction, the design practices that students engage in while creating their projects. While working with Scratch, student's understanding of themselves, their relationship to others and the technological world around them gets evolved and leads to a shift in perspective (Computational Thinking with Scratch, Harvard GSE).

Parson's puzzles empowers teachers/authors to impart interactive learning to students. 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. Parsons 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.

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. It is proposed that such extrinsic motivational factors help the students to actively 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 thinking abilities in the students (Mohan & Gupta, 2016). Further, sprites and gameful moves can be included to engage the students.

2. Related Work

A number of applications were developed in past to foster computational thinking such as Hot Potatoes, ViLLE, CORT and Dr.Scratch (Mohan & Gupta, 2016). 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. ViLLE (Rajala, Laakso, Kaila, & Salakoski, 2007) is a Java application/applet which allow context to be created around the editable code. The student can also step through a visualization of the execution of his/her solution. 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.

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. 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.

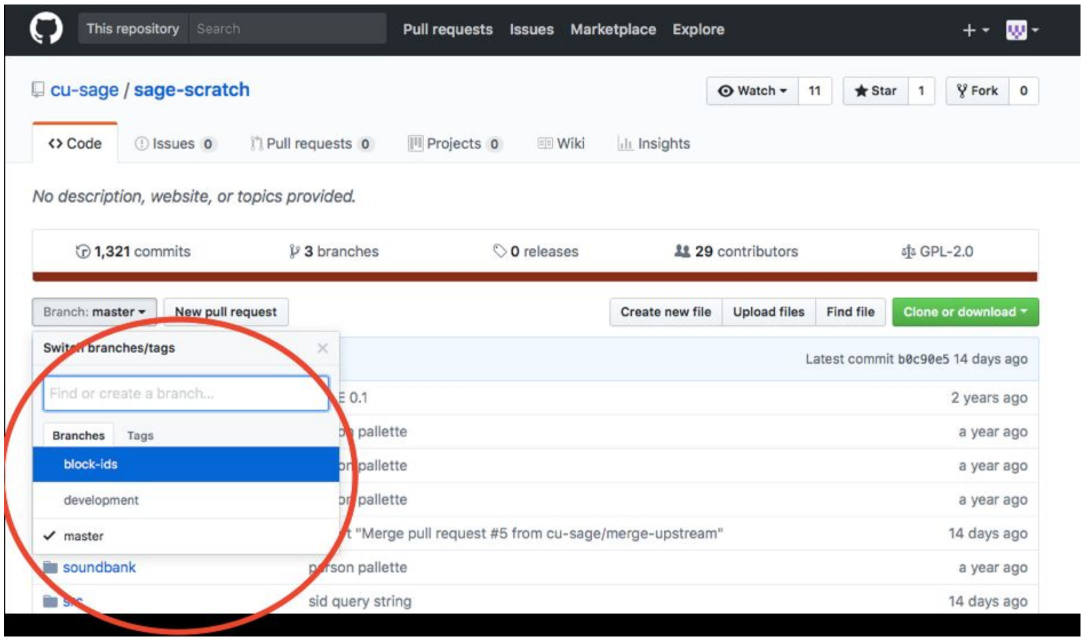
Recent years have witnessed growing interest in teaching computer science principles to students, with a goal of fostering computational thinking and encouraging participation in computer science. Intelligent game-based learning environments hold great potential as a tutoring framework for middle grade computer science education. Since these learning environments leverage intelligent tutoring systems' adaptive functionalities, it is possible to create dynamically tailored learning experiences that are simultaneously effective and engaging. One of the strategies adopted is adaptive task selection based on students' problem-solving performance, and other is adaptive hint generation by exploring the problem's solution space (Min, W., Mott, B., & Lester, J., 2014). They promote students' engagement through the rich narrative and situated challenges of games.

3. Sprints

This section will discuss the functionalities added during the sprints for the teacher/tutor to design and author Parson's programming puzzle for students.

3.1 Sprint 0

This sprint was dedicated to the environment setup and massive cleanup of the Github branches. Over the years, many branches were created by students to work during the semester and were not merged with master leading to outdated master branch. There were seven branches present at the start of Fall 2017. So, the first initiative was to carefully merge the branches and have just two branches - master and development. The merging was quite risky and had to be done carefully. After successful merging, three branches were created - master, development and block-ids. During semester, changes are first committed to development branch. It is merged with master branch after sprint meeting. The branch block-ids is kept for backward compatibility and can be safely deleted later.



E 0.1 — 2 years ago
pallette — a year ago
pallette — a year ago
pallette — a year ago
t "Merge pull request #5 from cu-sage/merge-upstream" — 14 days ago
parson pallette — a year ago
sid query string — 14 days ago

Figure 1: sage-scratch github

3.2 Sprint 1

3.2.1 Student, Assessment, & Video IDs via Query String

Presently, when Scratch launches it opens up a modal section asking the 'Student ID' and 'Assignment ID'. A new functionality is proposed where Scratch should identify the user as well as any assessment and video associated with the user. The student should not be asked for his ID. It could be implemented by passing the student, assessment and video ID information via Query String. Once implemented, the modal window asking the ID will be eliminated.

3.3 Sprint 2

3.3.1 Integrated Parson's Puzzle

The teacher will be facilitated to create Parson's puzzle and Constructionist video games so that students can take guidance from the direct instruction and learn the concept. Initially when an instructor would load a project in Design mode, the puzzle didn't appear in the script pane for the teacher to make modifications. As you can see in the below snapshot, the script is missing from the script pane.

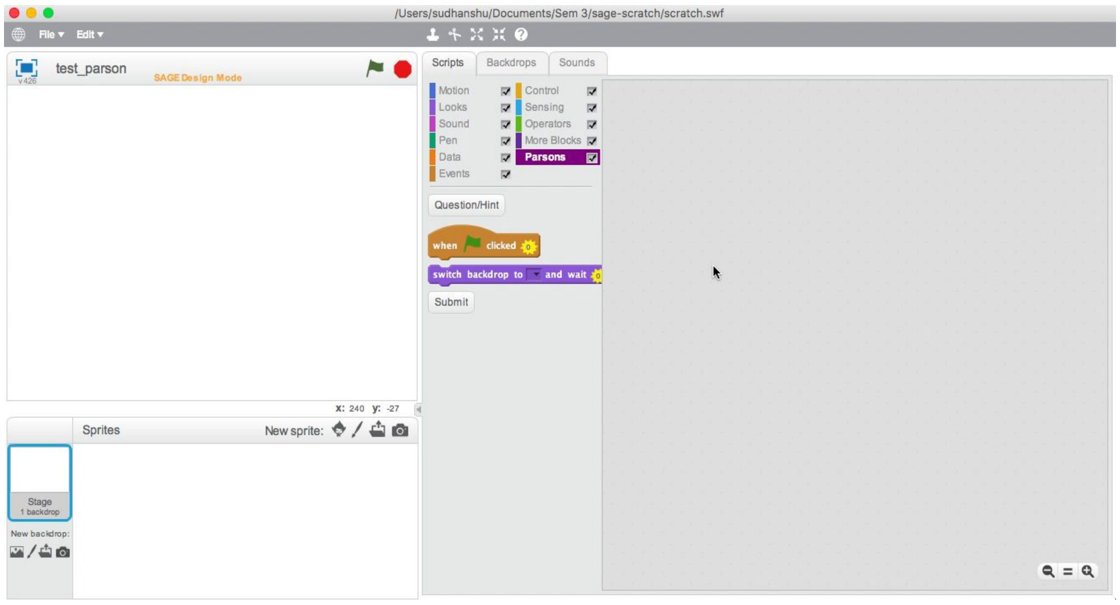


Figure 2: Design mode (earlier) with puzzle missing in script pane

This issue was fixed and now when instructor would load a project in Design mode, the puzzle will appear in the script pane to make further modifications.

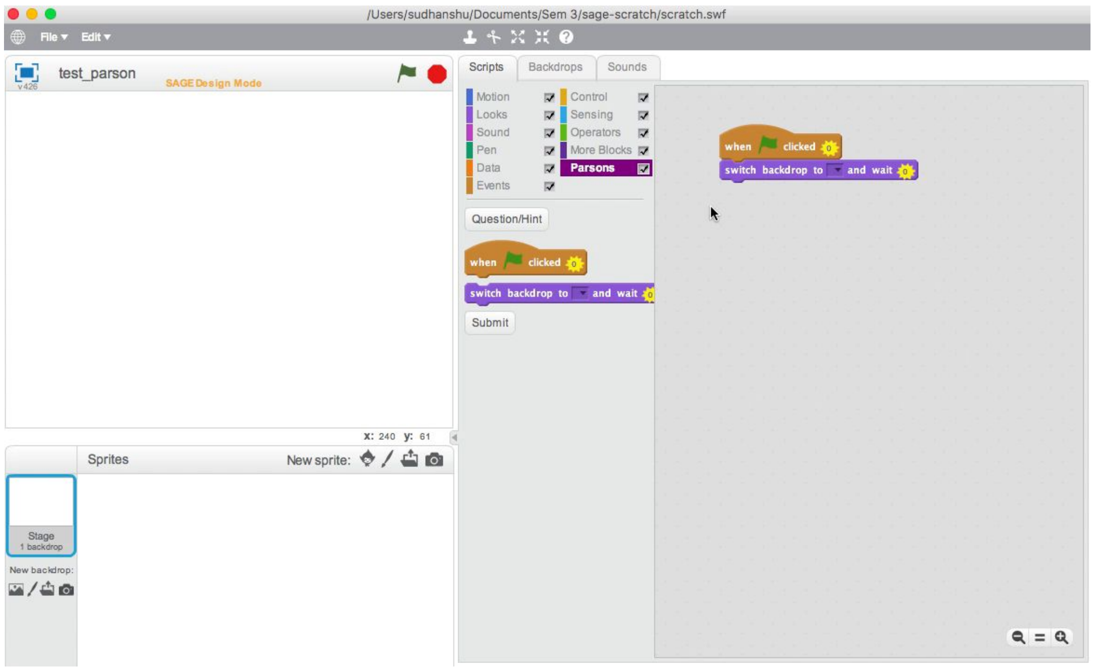


Figure 3: Scripts appearing in Design mode

Similarly, in Play mode, the puzzle was appearing in scripts pane when project was loaded by a student which showed the results to the students.

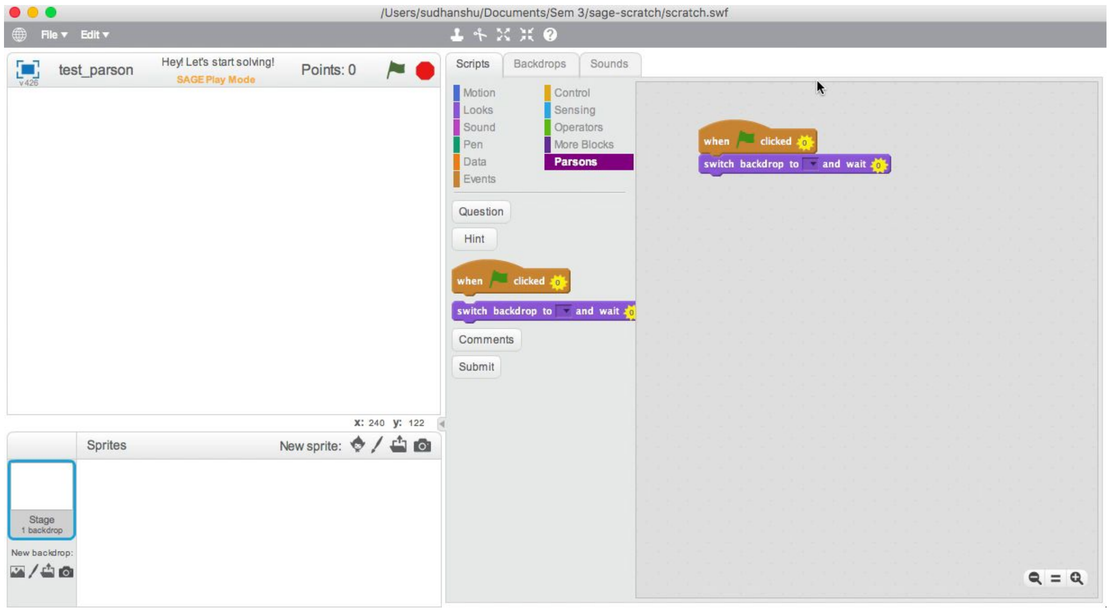


Figure 4: Scripts appearing in Play Mode (earlier)

This was also fixed and now the puzzle does not appear in the script pane when loaded in Play mode.

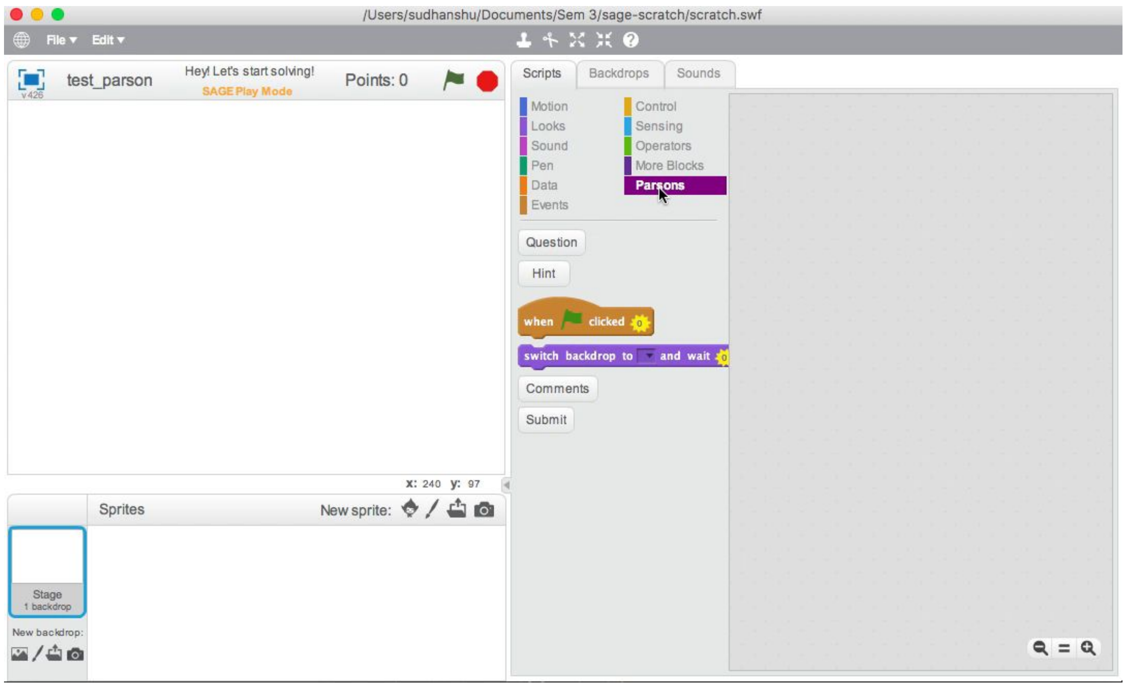


Figure 5: Play mode fixed

The stage content and Constructionist video game will appear in any mode.

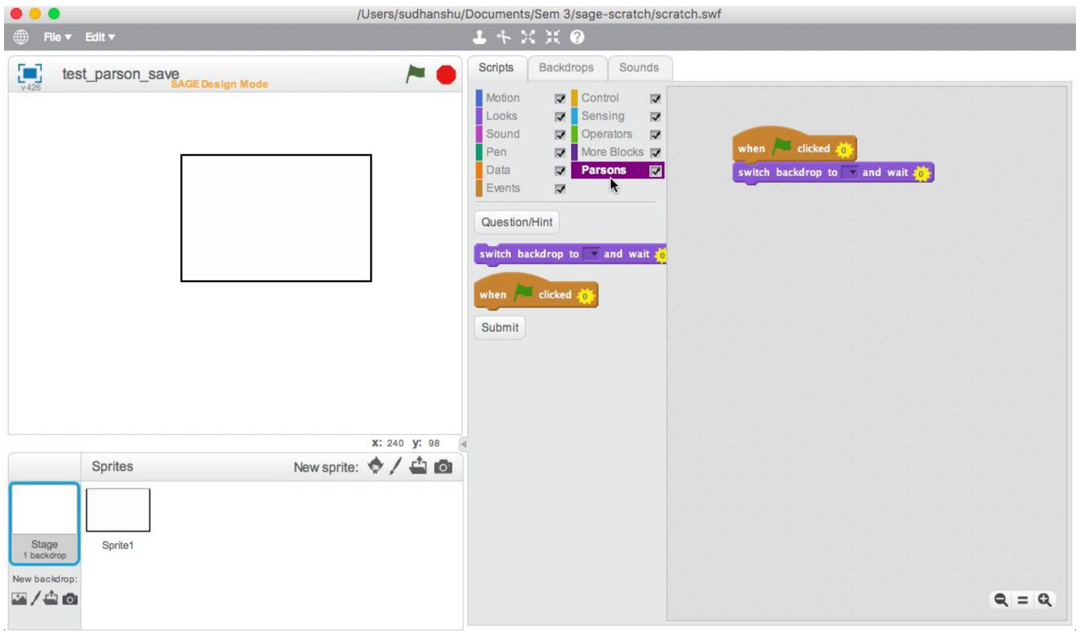
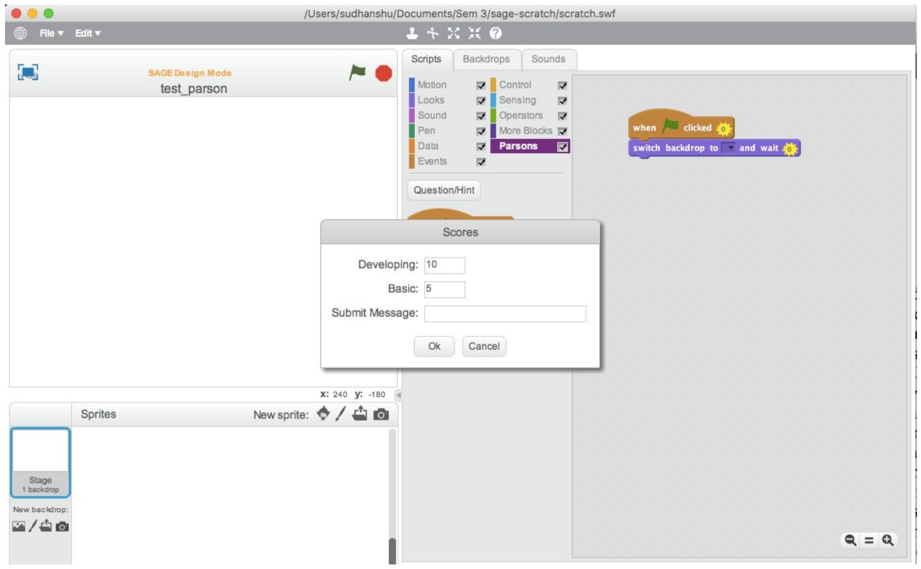


Figure 6: Stage content visible in any mode

3.4 Sprint 3

3.4.1 Parson's Assessment Scores

Once the Parson's puzzle is created or edited, SAGE should be able to calculate the maximum score for the puzzle and also prompt the teacher to enter two lower scores labelled as 'Developing' and 'Basic'. These scores will be then used by the teacher to differentiate performances between students by comparing them to the maximum scores. This information will help the system assign appropriately difficult in-Quest Constructionist Games.



Scores

Developing: 10

Basic: 5

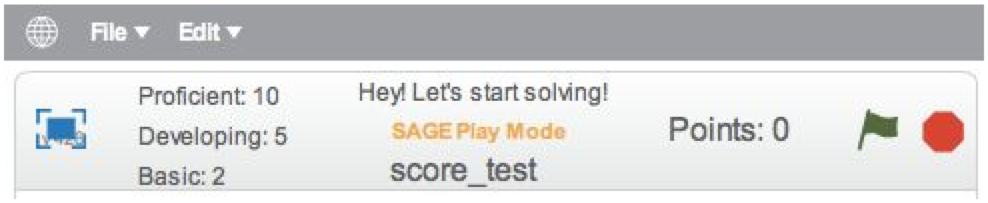
Submit Message:

Ok Cancel

Figure 7: Cut-off scores to be entered by instructor

3.4.2 Parson's HUD

A new Parson's Heads-Up Display (HUD) will be designed which will display the student's score in relation to the two cut-offs score (mentioned in section 3.4.1) and the maximum (or Proficient) score. This will help student to gauge his progress during the play.



Proficient: 10
Developing: 5
Basic: 2

Hey! Let's start solving!
SAGE Play Mode
score_test

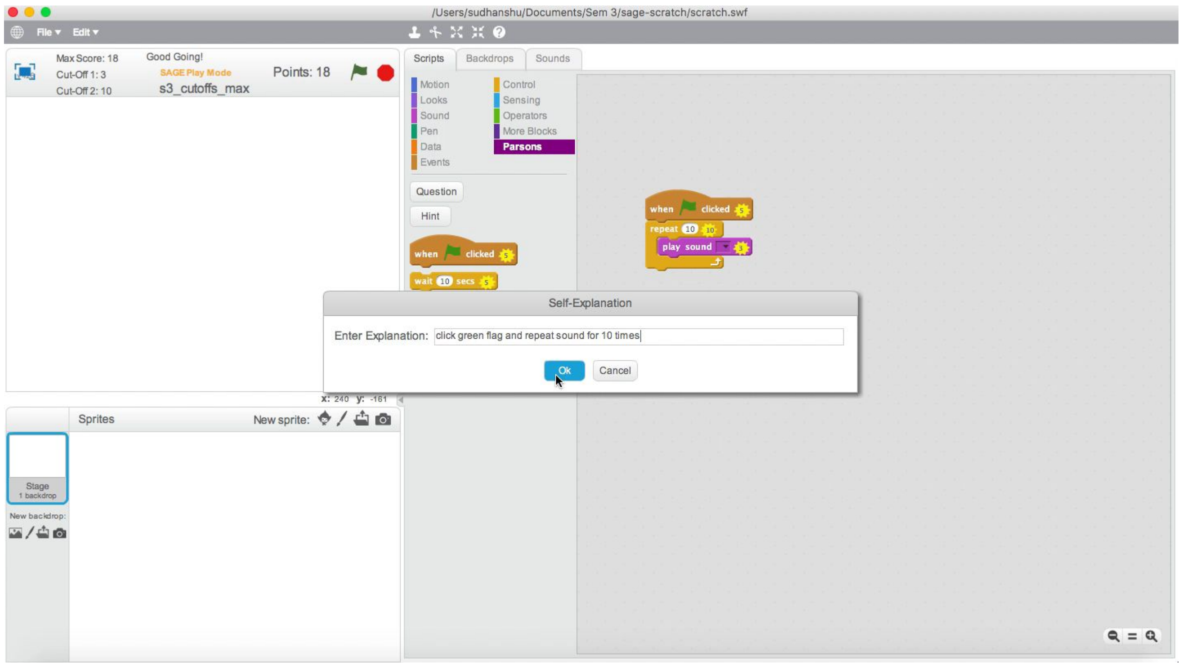
Points: 0

Figure 8: HUD displaying max score along with cut-off scores

3.5 Sprint 4

3.5.1 Parson's Self-Explanations

The students should be asked to explain their approach to solve a particular puzzle prior to the submission of the assignment. This will give an opportunity to students to reinforce learning by self-explanation. In the previous implementation, a comment box appears before the submission of the assignment. This box was repurposed to implement the self-explanation feature.



Self-Explanation

Enter Explanation: click green flag and repeat sound for 10 times

Figure 9: Self-explanation box

3.5.2 Parson's Submissions

Upon submission of the assignment, a bunch of submission messages are displayed. The teacher should be given the ability to author these messages to provide an improved student experience so that they are motivated to play more and thus stress on the ‘Addictive’ component of SAGE.

3.5.3 Parson's Palette

While working on Sprint 1, I observed that Parson's Palette was not getting refreshed when new projects were loaded in SAGE play mode. The blocks were getting appended to the existing blocks which was breaking the functionality. In the below screenshot, the palette contains blocks of three projects which were loaded one after the other.

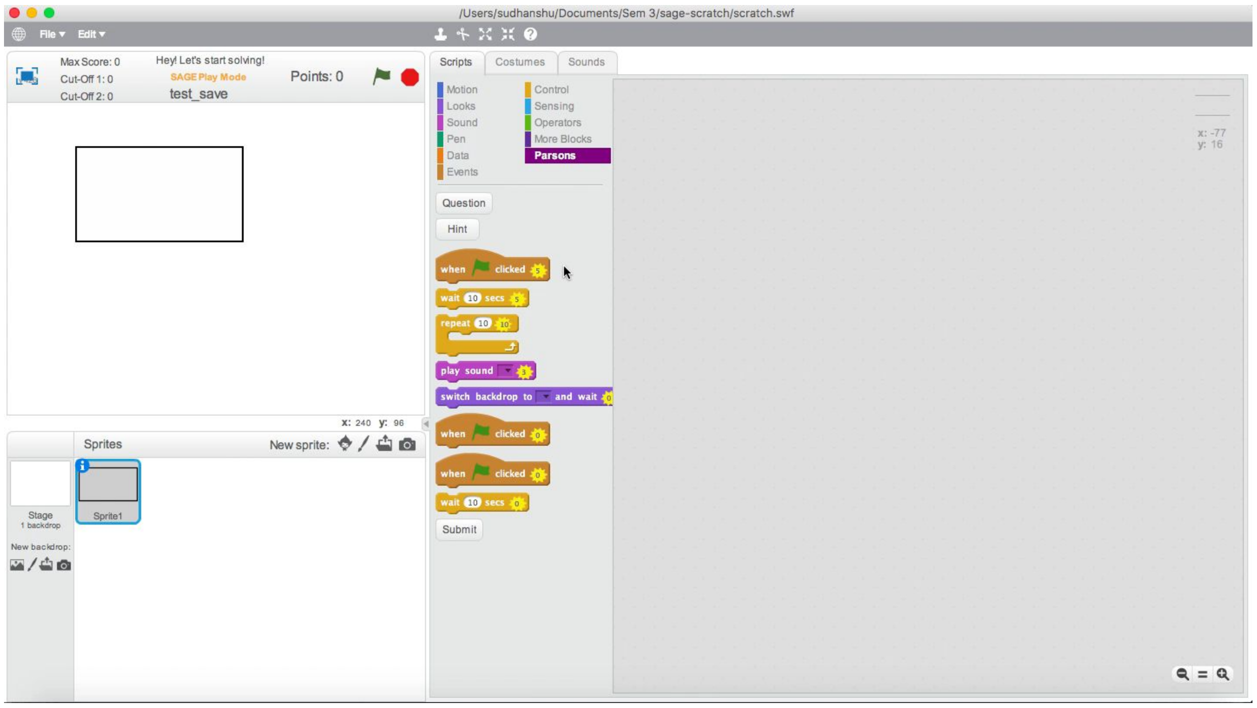


Figure 10: Parson's Palette not getting refreshed when new project is loaded

Thus, I added a new story to fix this bug and planned to fix it in Sprint 4 depending on the availability of the time. So, I debugged and found the place where the Parson's Palette should be initialised whenever a new project is loaded. The issue is now fixed and works seamlessly in play as well as design mode as could be seen in the below screenshot where 2-3 projects were loaded one after the another but blocks of only the latest project are shown.

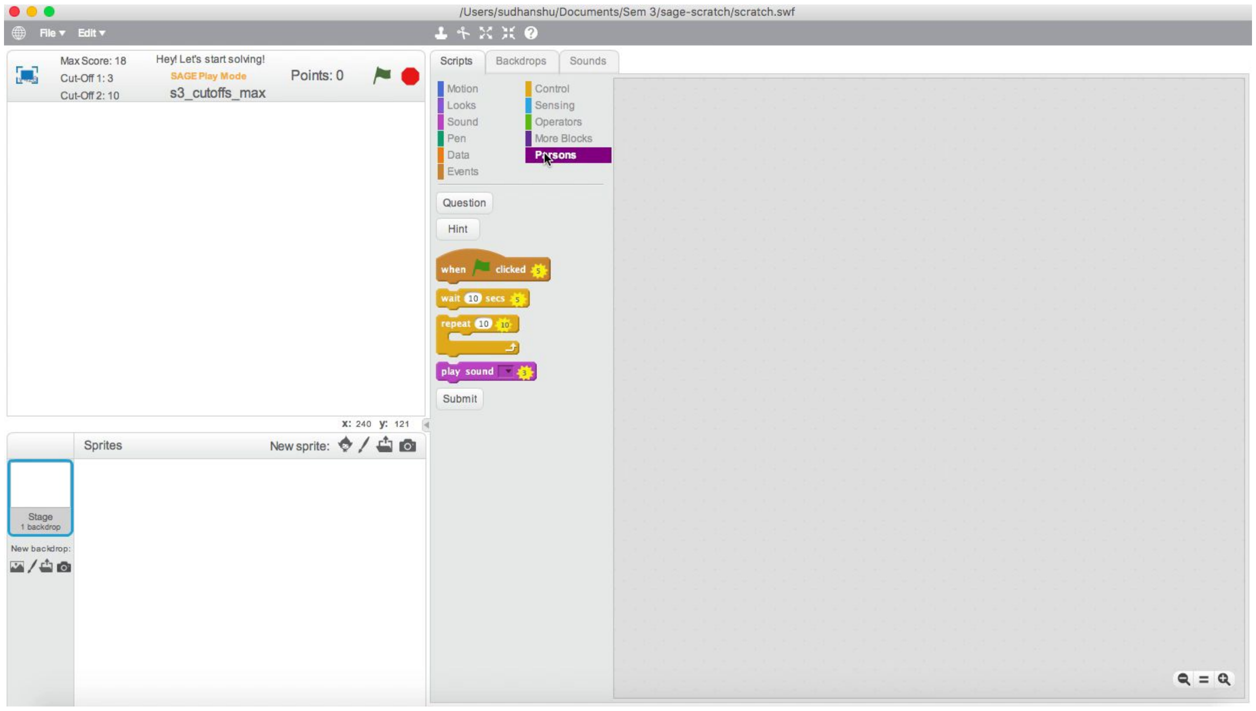


Figure 11: Parson's Palette gets refreshed whenever a new project is loaded

3.5.4 Assessment Server Connectivity

The game should post the game state every second to the assessment server and retrieve results so that the students can receive formative feedback as they play. The student ID, assignment ID and objective ID are being passed to Scratch via query strings. These parameters along with the game state is posted as a json to the assessment server. Similarly, Scratch fetches the assessment results from the assessment server every second. Scratch process the results (exclude/include blocks) and the process continues. The framework for the entire process is in place now and integration testing can be done once the infrastructure on assessment server is ready.

4. Timeline

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	October 1, 2017
2	SAGE Scratch Setup (Sprint 0)	October 12, 2017
3.	Student, Assessment, & Video IDs via Query String (Sprint 1)	October 26, 2017
4.	Integrated Parson's Puzzles (Sprint 2)	November 9, 2017
5.	Parson's Assessment Scores (Sprint 3)	November 23, 2017
6.	Parson's HUD (Sprint 3)	November 23, 2017
7.	Parson's Self-Explanations (Sprint 4)	December 7, 2017
8.	Parson’s Palette (Sprint 4)	December 7, 2017
9.	Parson's Submissions (Sprint 4)	December 7, 2017
10.	Assessment Server Connectivity (Sprint 4)	December 7, 2017

Table 1: Milestones completion dates

5. Future Work

5.1 Mode based Server Interaction

The information that a student or instructor has signed in is available within the affinity space. Scratch window should be rendered in Design or play mode accordingly. If a student has signed in then only play mode should be rendered and the game state should get posted to the assessment server every second. Similarly, if a instructor has logged in then design mode should render by default and the game state posting should be suspended for that session.

5.2 Parson's Scoring

The scoring scheme presently implemented in Parson's puzzle need to be revisited. The incorrect move presently penalizes one point. In future, the penalty of the incorrect move should depend on the block point value. The scoring system should support multiple solutions to the puzzle and should be able to give more points to the elegant solution. Also, there should be a way to differentiate use of distractors from the use of solution blocks placed in incorrect positions.

5.3 Parson's Coaching System

The information and metrics stored during the gameplay should be used to design an efficient Coaching system. The teacher should be able to author messages that would be presented to students during the puzzle. SAGE should provide preset configurations of coaching messages and an option to choose a default from them. Instructor should also have the freedom to modify the messages loaded via a preset configuration. In case a student is struggling a lot while solving a puzzle, a message to watch the video should pop up.

6. References

Computational Thinking with Scratch - http://scratched.gse.harvard.edu/ct/index.html

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.

Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). VILLE - A language-independent program visualization tool. Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88, Australian Computer Society. Raymond Lister and Simon, Eds.

Mohan S., Gupta A. (2016).*Formative SAGE Assessments: Parson's Programming*

Garner, S. (2007). An exploration of how a technology-facilitated part-complete solution method supports the learning of computer programming. Journal of Issues in Informing Science and Information Tech- nology, 4, 491–501. Retrieved from http://proceedings.informingscience.org/InSITE2007/IISITv4p491-501Garn260.pdf

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*.

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

Min, W., Mott, B., & Lester, J. (2014). Adaptive scaffolding in an intelligent game-based learning environment for computer science. In *Proceedings of the Workshop on AI-supported Education for Computer Science (AIEDCS) at the 12th International Conference on Intelligent Tutoring Systems*