SAGE: Affinity Space Design and Game Routes

Research Proposal: Spring 2018

Gavi Rawson & Christine Hsu

Introduction

SAGE introduces a game-based learning approach for elementary school students to learn programming. The SAGE interface allows teachers to create unique games for their students, and track student progress on an individual level. Students learn core programming concepts through these games, and track personal progress through a reward system.

The current implementation of SAGE can be improved in two major areas:

1. A redesigned Affinity Space focused on instruction, consistency, and simplicity.
2. Game Routes (previously Game Mechanic Blocks) that allow teachers to create focused games for students.

Adding these two elements will allow SAGE to be more intuitive, engaging, and focused on specific students. Ultimately, these changes will increase the level of student learning.

Redesign: Instruction

In order to deploy SAGE to schools, the interface should be intuitive and easy to navigate as a first-time user. Currently, the SAGE interface is difficult to understand for both students and teachers. While students can learn to use the interface from their teachers, we need to ensure teachers are familiar with the interface.

We can make the interface more intuitive by adding a help button to each page. The help button will explain core concepts specific to a page, such as the purpose of each tab in the dashboard, or the meaning of and relationship between terms such as “missions” and “quests.” In situations like designing an objective, the help button can provide sample content to offer direction for teachers using the interface for the first time.

Additionally, it may not be clear to a teacher how actions in the teacher side of the interface affect the student’s interface. We can increase transparency by implementing a preview button. Pressing the button will pop up a preview of the content from the student’s perspective.

Redesign: Consistency

The SAGE interface will benefit from a consistent design painted across each page. Currently, many of the pages contain unorganized lists of clickable content. By organizing the content on each page similar to the manner of organization in the “Mission Management” page, the interface will become familiar and easier to use.

Further, we can paint the student and teacher interfaces with different consistent designs. The current design works well for teachers, as they are older and appreciate a more professional and clean user interface. On the other hand, students are much younger, ranging from grades 6-8. To engage this younger crowd, we can implement a more colorful and playful design.

Redesign: Simplicity

By focusing on simplicity when redesigning the interface, we can construct a cleaner and more intuitive interface. On the teacher side, the process of creating and editing items such as missions is cluttered. We can simplify this process by rearranging UI elements and adding features like “edit” buttons that consolidate functionality.

We can also begin to restructure the naming of the items in the menu, as well as the content for each menu item. For example, let us examine the “Home” page on the teacher side of the system. The content of the page displays scores for each student per mission, and thus a more intuitive name for the menu item is “Scores.” Further, the content of the page is organized such that a teacher must click a specific quest to see student scores for that quest. Issues will arise if a teacher wishes to see which quests a specific student has completed. This would currently involve clicking into all the quests and searching for the student. We can simplify this flow by displaying a list of all students in a grid which contains columns of information related to a student. One of the columns can be the quests the student has completed. We can add filters to the top of the page that allow a teacher to single out specific fields. Further, clicking a student’s name can display detailed statistics. While this is a single and specific example, we can apply this simplified restructuring logic to the rest of the pages in the interface.

Redesign: Additional Features

We have an opportunity to add additional features to the student side of the interface that will foster learning and self-evaluation. We can add progress meters for missions and quests, as well as completion indicators for any objectives that are met. We can also add

a feedback system that allows students to send personal notes to a teacher (Katrien Verbert). This will give quieter students an opportunity to seek help.

There is also an opportunity to add additional features to the teacher side of the interface. Teachers can benefit from real-time tracking of student activity (Katrien Verbert), through which they can ensure students are engaged in specific quests. We can also add comprehensive statistics for each student which can include the specific blocks used, and the time a student spends on a game. Research has shown that the latter is a useful metric for teachers (Katrien Verbert).

Game Routes: Design

The current interface provides a freeform mode of instruction. Teachers design games by creating sprites and an environment that form an overall challenge. They may also specify the blocks a student can utilize to interact with the sprites and environment to ultimately solve the challenge. This approach does not allow teachers to create targeted challenges for their students.

We can introduce Game Routes to focus challenges by offering students direction for solving puzzles. A Game Route is comprised of one or more containers that are linked together. Each container offers a place for students to place Scratch blocks. Each container has a set of constraints that are defined by a teacher when the game is designed. Teachers may constrain the type and number of blocks that may be added to a container, as well as the maximum or minimum amount of points in a container.

To offer further guidance for students, teachers can place permanent blocks within containers. Students see the permanent blocks when they open the game, and must make use of these blocks to solve the puzzle. Students can place blocks around the permanent blocks, but the permanent blocks may not be removed. This concept allows teachers to introduce topics to students that they must comprehend if they wish to ultimately solve the puzzle. It also provides a mode of direction for students, suggesting that the best method of solving the problem before them is to use the highlighted concept (Cagin Kazimoglu).

Though the core idea of a Game Route is to offer direct instruction to students, we do not want to conflict with the underlying idea of Scratch. Scratch is built with a constructionist design, offering students the freedom and creativity to explore multiple paths to solving a puzzle. This idea of free constructionism is important for learning (David Weintrop). To retain this element, we can allow teachers to design multiple Game Routes for a game. Students can choose any game route available to solve the puzzle.

Game Routes: Implementation

To implement Game Routes in a clean manner, we must plug into the current Scratch interface. To explain and visualize a possible implementation, we have included a mockup concept below (Figure 1). We will refer to the mockup to explain the proposed features.

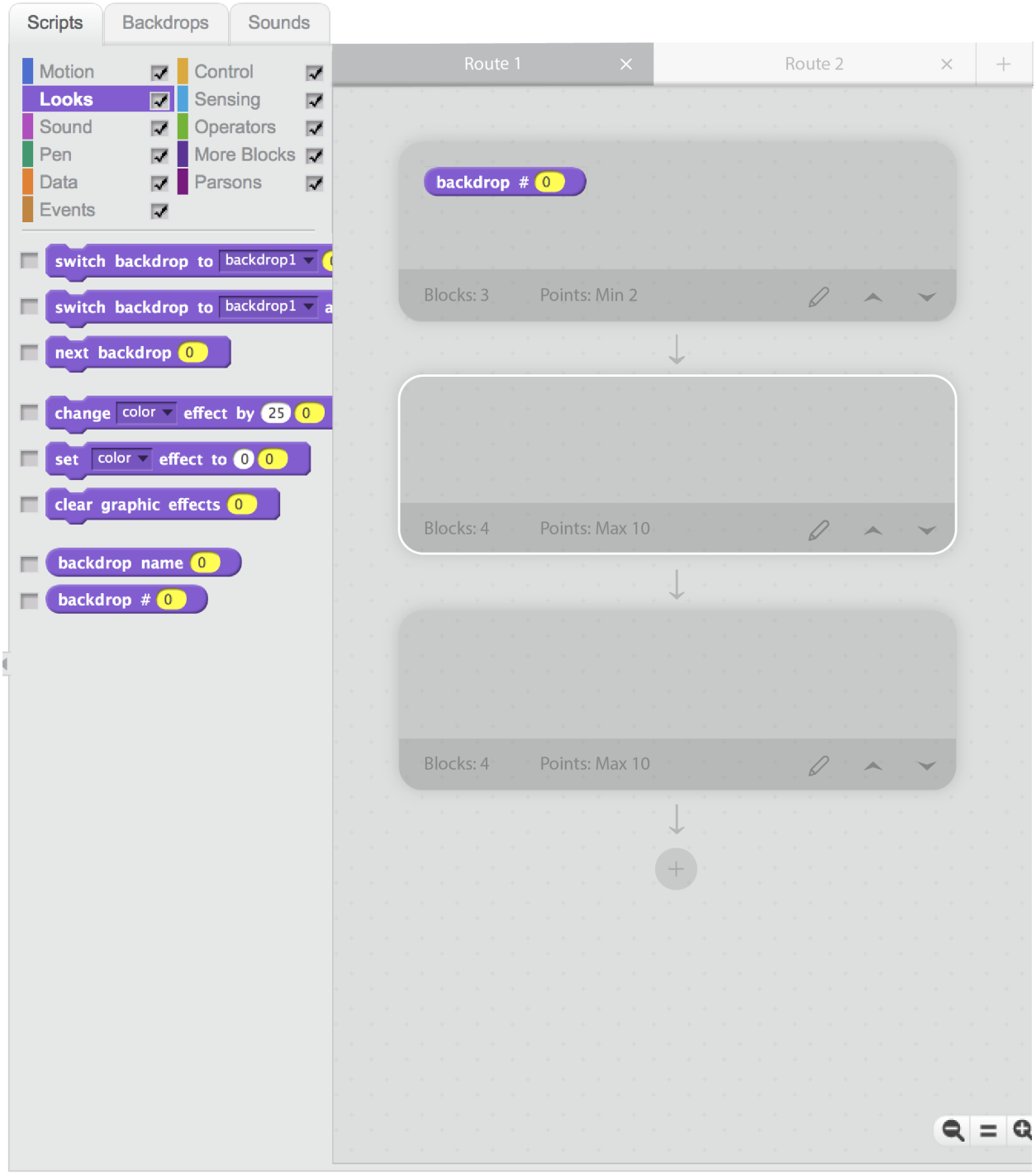

Figure 1: Game Routes Concept Design

This mockup is in the design mode context, through which a teacher creates a specific game for students. When initially opened, there will be no Game Routes defined, just an open-ended scripts space in accordance with the current implementation. To create a

Game Route, a teacher can press the “+” button in the tab bar on top of the right pane. This will create a new instance of a Game Route, and will automatically create one container. Additional Game Routes can be created by clicking the “+” button in the tab bar. A teacher can add additional containers to a Game Route by pressing the “+” button inside the circle under the existing containers.

Each container has a unique set of constraints defined by a teacher. When a container is selected, denoted by the white outline around the container, the scripts pane will change to be specific to that container. A teacher can allow specific blocks to be inserted into a container by clicking the check boxes. Note that all containers are isolated, so allowing blocks for one container will not allow them for others.

Further constraints can be controlled through the arrows and edit button on the bottom of each container. The arrow buttons allow for changing the order of the containers. The edit button will pop-up a window in which a teacher can set the max/min amount of points and blocks for a container. A teacher can also delete the container through the pop-up window.

Once a container exists, Scratch blocks may only be added to containers, not the usual background area. When students add blocks to the containers and execute their program, the containers will operate in succession. When a teacher adds a block to a container, it becomes a permanent block that must be used by the student to complete the puzzle. To explain the programming concept behind a permanent block, we can add a help button specific to each container that can be clicked by a student.

The design for the student’s end of the interface will look similar, but with less edit control and more instruction. The edit button and arrows will disappear, and each container will start with an overlaid title stating “Place blocks here.” We must also make sure students are aware that they may complete any of the available routes to solve the puzzle.

Conclusion

The SAGE interface will benefit greatly from a redesign of the Affinity Space and the addition of Game Routes. The design changes will make the interface more intuitive and accessible for both teachers and students. Game Routes will allow teachers to focus games on specific core programming principles.

Our proposal splits the project into two parts; design and implementation. We have allotted 3-4 weeks for design and 6-7 weeks for Game Routes, as described below in the milestones section. However, we have understood that the driving force of this project

should be implementation and not design. If it is believed that we should spend less time on design, or not work on design at all, we will happily adjust the milestones. We are looking forward to diving into the project!

Milestones

Milestone	**Date**
Design: Teacher interface wireframes.	2/13
Design: Student interface wireframes.	2/23
Game Routes: Implement instances of Game Routes (tab bar).	3/12
Game Routes: Construct containers associated with a Game Route (general UI).	3/23
Game Routes: Allow Scratch blocks to be inserted in and associated with specific containers. Allow teachers to specify which blocks may be added to a container.	4/2
Game Routes: Add constraints to containers.	4/13
Final Report	4/27

References

Cagin Kazimoglu, Mary Kiernan, Liz Bacon, Lachlan MacKinnon. "Understanding Computational Thinking Before Programming: Developing Guidelines for the Design of Games to Learn Introductory Programming Through Game-Play." International Journal of Game-Based Learning (2011).

David Weintrop, Nathan Holbert, Uri Wilensky, Michael Horn. "Redefining Constructionist Video Games: Marrying Constructionism and Video Game Design." Constructionism 2012 (2012).

Katrien Verbert, Sten Govaerts, Erik Duval, Jose Luis Santos, Frans Van Assche, Gonzalo Parra, Joris Klerkx. "Learning dashboards: an overview and future research opportunities." Personal and Ubiquitous Computing (2013).