FEATURE

COLLABORATION IN PROGRAMMING PROJECTS

Stefan Seegerer, **Tilman Michaeli,** and **Jane Waite** introduce Smerge, a free version-control system for the block-based programming language Snap!

Getting pupils to collaborate effectively on programming projects is tricky. Often we can't see how much work each pupil has done and it's hard to interrupt the flow of work to provide constructive feedback. It's also difficult to bring the reality of the world of work, and how programmers work in industry, into the classroom. Smerge is a free Snap! add-on developed by a research team in Germany that helps tackle these issues.

What is a version-control system?

Version control is crucial in real-world software development. Whether you are working in a small or large team, collaborating face to face or remotely, knowing where you are up to in the creation and modification of your code is essential. Learning about version control and the underpinning concepts as a core computational practice is important, as is learning how to use version control. However, professional tools are extremely complex, and even professional software developers can have problems with them.

Version control for Snap!

Smerge is an easy-to-use, beginner-friendly online version-control system for the block-based language Snap! Smerge enables students to work together collaboratively to develop programs. Teachers and peers can provide feedback on specific versions of a project, giving hints, tips, and suggestions for improvement, and they can do this in class or at home.

The design of Smerge was based on a review of professional version-control systems and how they were used in computer science classrooms. While being tailored to novice programmers, Smerge still includes all core concepts of professional version-control systems, which are:

- Project history: in Smerge, the project and its history are visualised in a project history diagram (a graph).
- Committing: changes are added to the project by committing them to the version control system; this is done within Snap! as changes are committed back to Smerge by simply clicking on an add-on Snap! block ('Post to smerge').
- Branching: branching opens an alternative path so that changes can be made in parallel. A branch might be used for developing new features or fixing bugs, without impacting the current state of the project. This is where Smerge differs from many existing systems. Each person editing the shared program is provided with their branch automatically.
- Merging: a version control system simplifies combining changes and the resolution of conflicts. For merging, users select which nodes they want to merge in the project history diagram (graph) and confirm their selection. When possible, conflicts are resolved automatically by Smerge. If they cannot be automatically resolved, the conflicting code fragments are shown in a merge view in Snap! for the person editing the program to sort out.
- Data backup: having all files backed up to version control allows for returning to every (older) version easily, thus, enabling risk-free trial and exploration of different ideas. In Smerge, old versions can be accessed using the graph.

How to Smerge

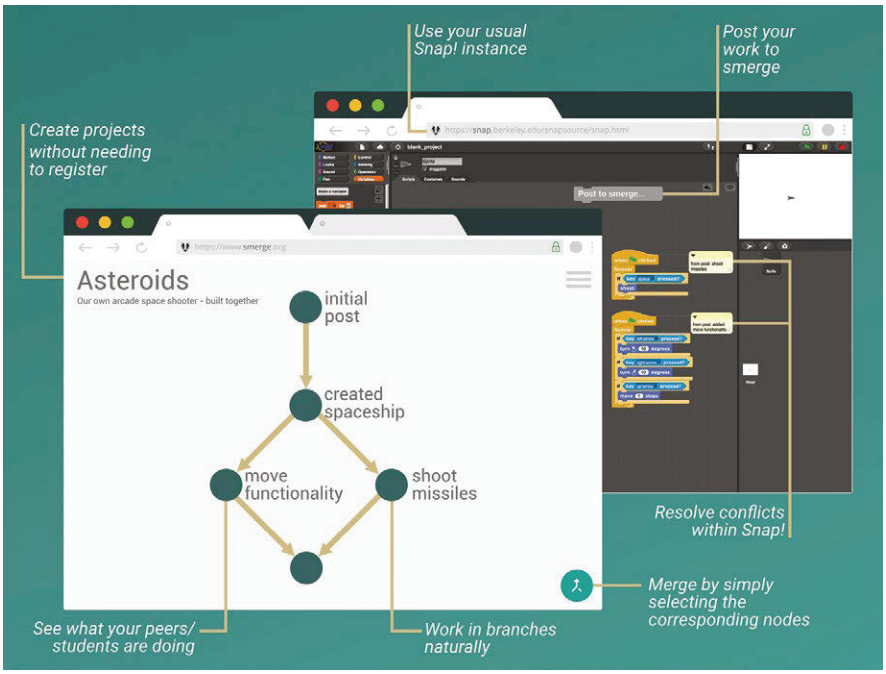
To get started with Smerge, the teacher or students create a project on **smerge.org**. At no point do they need to register on the Smerge system. Students might start from a given template or a plain Snap! project. Imagine that a team wants to create a game such as Asteroids, the classic arcade shooter game. One student can develop the functionality to move the spaceship, and another to program the feature to shoot missiles.

To start implementing this in Snap!, each of the students just double-clicks on the node they would like to edit. Each student is automatically provided with their own branch to work on. Then, they just use their usual Snap! window for programming. When finishing adding their feature, they post their program back to Smerge by using the custom 'Post to smerge' block in Snap! This block is found on the variables palette.

USEFUL SMERGE FEATURES

Smerge was created for classrooms and offers teaching and learning features:

- Enables collaboration on project-based learning with Snap!
- Teachers can supply custom sample code
- Teachers and students can provide feedback or marking and suggest alternative ways forward
- Helps teach industry version control skills
- Helps students think about the development process by making it more transparent
- Students can see the version history, revert changes, and go back to old versions
- Teachers and students can easily share projects with no need to register and no personal information stored



Smerge makes version control simple

One of the students then activates the merge mode in Smerge. This is done by clicking the merge button. The student selects the two nodes to be merged and confirms the selection. The code will be automatically merged, resulting in a new version. If more than one student changed the code for the same script in the same sprite, both versions of that changed script are retained by Smerge and both are shown in the new merged code. This enables the students to discuss the conflict, test the two options, and choose which version to keep.

Project ideas

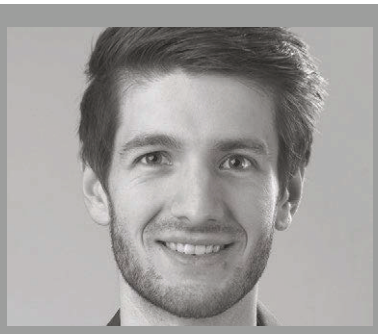
You can use Smerge for any Snap! project. Here are some ideas to foster collaboration:

- Animal dance party: the teacher creates a starter program, with one dancer and music. Every student then adds their own dancer to the party.
- Celebration card: in this project, students collaboratively create a celebration card, such as a birthday card. Every student creates one letter, and the letters are shown one after the other. This involves students working out how to coordinate the appearance of each sprite.
- Quizzes: the teacher provides a starter quiz and students add extra questions. Lots of testing might be needed here to check that shared variables, such as a score, and lives are used in the same way by all contributors.
- Animated story: students storyboard the animation of a story, figure out what work needs to be done, and share the work between group members.
- Games: last but not least, students can collaboratively create games, as each student focuses on its different parts.

Pedagogy ideas

As well as modelling how Smerge works, try a 'use, modify, create' approach. Create an example Smerge project and ask students to explore the project history. Then ask them to take one branch to make their own version. Ask the students to work in pairs to compare their versions and discuss a potential combined version. Encourage using the merge mode, and compare the results with their expectations. You could start more simply, introducing collaboration later on.

Using Smerge in the classroom provides valuable opportunities to address important aspects of collaboration and program design. When using version-control systems, students learn to sensibly decompose functionality, create reusable functions, think about interfaces or test data, and meaningfully name sprites, assets, or descriptions of what has been changed. Smerge is free to access at **smerge.org**. (HW)



STEFAN SEEGERER

Stefan is a researcher at the Computing Education Research Group within the FAU in Germany, exploring ways to make computing accessible to everyone (**@StefanSeegerer**).

TILMAN MICHAELI

Tilman is a researcher at the Computing Education Research Group within the FAU in Germany. He is working on projects such as developing concepts for the classroom supporting debugging skills, demystifying AI, and fostering collaboration for programming projects.

JANE WAITE

Jane is a researcher and teacher trainer at Queen Mary, University of London. Current interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC.