Blink: a debugger for SCRATCH

Niko Strijbol, Robbe De Proft, Klaas Goethals, Peter Dawyndt, Christophe Scholliers
Ghent University (correspondence: niko.strijbol@ugent.be)

Debugging is an important aspect of programming. Most programming languages have some features and tools to facilitate debugging. As the debugging process is also frustrating, it requires good scaffolding, in which a debugger can be a useful tool. Scratch is a visual block-based programming language that is commonly used to teach programming to children, aged 10–14. It comes with its own integrated development environment (IDE), where children can edit and run their code. This IDE misses some of the tools that are available in traditional IDEs, such as a debugger.

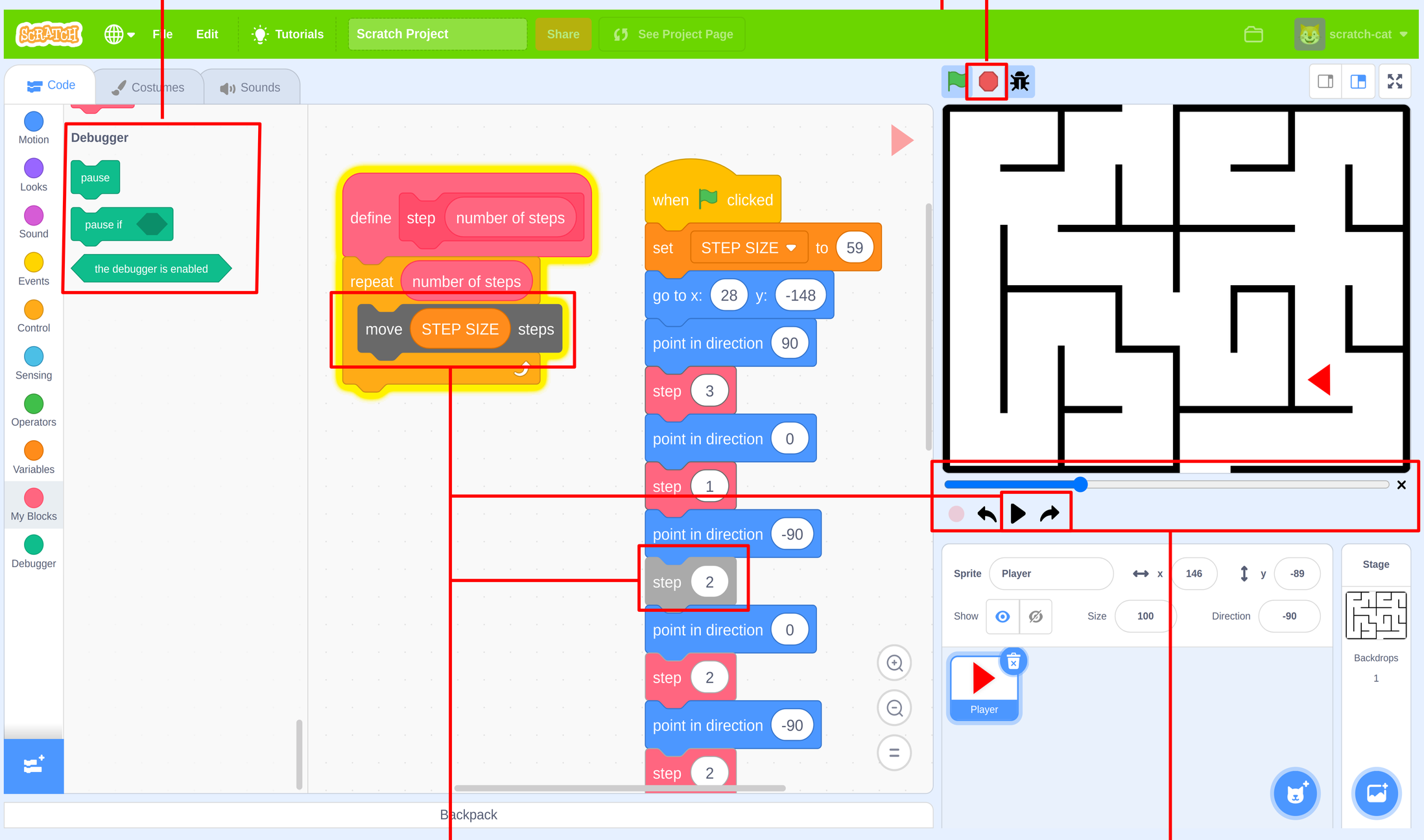
Try it yourself at scratch.ugent.be/iticse2023

(Conditional) breakpoints

We provide two pause blocks, similar to traditional debuggers: one that simply pauses, and one that pauses if a condition holds.

Debug mode

A green top bar indicates that the debug mode is enabled: execution is automatically saved.



Pause and step

The two left-most controls are a play/pause buttong and a step button respectively. This allows pausing the execution at any point, and executing each step one by one.

The next block that will be executed is highlighted in dark grey. Since this is inside a custom block definition, the caller is highlighted in light grey.

We also noticed that the step functionality (with the next block being highlighted is also useful as an instructional tool: it can help explain the execution order, especially with control flow blocks.

Time-travel debugging

Each ‘step’ of the exeuction is saved in a timeline. This allows going back, stepping through the execution backwards and forwards. This feature is especially useful to find where a bug lies, which is a difficult task. For example, breakpoints already require the students to have a general idea of where the bug lies, before the debugging has actually started.

In this screenshot, we are pauzed in the past. Once in the past, the play/pause and step button operate on the timeline: pressing the step button does not execute one step, but replays one step from the timeline.

Advanced time-travel debugging (future work)

We are working on an extension of the timeline feature, where you can go back on the timeline, modify some code, and then resume execution from that point, but with the modified code.

The demand for this feature was observed during user testing: multiple students tried to use this feature, indicating that is useful and intuitive.

GHENT UNIVERSITY
FACULTY OF SCIENCES

Part of this work was supported by Flanders Innovation & Entrepreneurship, Belgium (VLAIO HBC.2020.3300).