MILES BERRY PRINCIPAL LECTURER

IN PRAISE OF BLOCKS

Why we should resist the urge to move on to teaching a text-based language too soon

Scratch may not have been the first block-based programming language, nor is it the most flexible, but it's undoubtedly the world's most popular. I don't think it's an exaggeration to suggest that the global movement to teach children to program wouldn't have got very far if it hadn't been for Scratch or something very much like it. I think there's a danger that CS educators working with upper primary or lower secondary students often want to move on to Python or another text-based language sooner rather than later, in part because they can, but also perhaps because pupils moan that they've 'done Scratch'. I think it's worth resisting this urge, staying a little bit longer with Scratch, and making sure the foundations of coding and computational thinking are really firm, for all our pupils, before we subject them to the additional cognitive load of text-based programming. Let me give you some of the reasons why.

Programming is *hard*, but text-based programming is particularly difficult because the programmer has to think about two things at the same time: the semantics of the program, the algorithms and data structures that the programmer is trying to code, and the syntax of the program, the particular vocabulary, grammar, and punctuation that's needed to express it in Python or whatever. In Scratch, the syntax is taken care of by the blocks provided in the language, and thus the programmer gets to focus much more of their thinking on the semantics. If what we're interested in is developing pupils' 'computational thinking', then Scratch makes it much easier to emphasise this, rather than having to master the syntax of Python.

One consequence of the syntax being built into the blocks is that it's all but impossible to make syntax errors when using a visual language. Many of the frustrations felt by those learning to program in text-based languages are due to the frequency with which they'll make syntax errors in their code, thus visual language programs are far less likely to have errors than their text-based equivalents, and the errors that programmers do make are likely to be much more *interesting*, providing much more opportunity for learning from mistakes than simple typos.

In developing Scratch, Resnick and his team were deeply inspired by construction toys, from Froebel's gifts through to modern LEGO. While LEGO sets typically come with step-by-step instructions, younger children will just *play* with LEGO, making whatever they have in mind through clicking blocks together and seeing what works, and more experienced LEGO builders will range far beyond any step-by-step guides, again letting their imagination take the lead in creating novel, original artefacts. Because it's so easy to playfully experiment in Scratch, pupils can quickly try ideas out to see what difference they make, and to figure out for themselves what works, or what improves their program, and what doesn't.

By making it easy to learn the programming language, Scratch makes it possible for young programmers to acquire a sense of mastery of the tool, of fluency in the language. After an introduction to the language, and plenty of opportunity for play and experiment, many children pass beyond learning to code to being able to express their creative ideas fluently through the medium of code: instead of spending time learning the language, they can spend their time building, making, experimenting, and sharing. (HW)

Miles is Principal Lecturer in computing education at the University of Roehampton. He's a member of the Raspberry Pi Foundation and serves on the boards of CAS and CSTA.