Using children's literature to introduce computing principles and concepts in primary schools: work in progress

Sarah Twigg, Lynne Blair, Emily Winter
School of Computing and Communications
Lancaster University
Lancaster, UK
s.j.twigg@icloud.com, l.blair@lancaster.ac.uk, e.winter@lancaster.ac.uk

ABSTRACT

With the recent paradigm shift in the teaching of computing and computational thinking skills, schools are engaging pupils as young as five in learning principles and concepts of programming. However, there are still many challenges within primary computing education, including the cost and availability of resources, and teachers' familiarity and/or confidence with these resources. In this paper, we offer an approach that develops a creative story-based pedagogy to address constraints such as these and facilitate the development of lesson plans supporting scaffolding and differentiation. Children's literature is used to introduce concepts such as pattern matching, abstraction and algorithms, along with the three main programming constructs of sequencing, repetition and selection. Through four stages of Read-Act-Model-Program (RAMP), we present a set of unplugged and Scratch-based activities and reflect on the potential impact of this educational opportunity to inspire an early interest in computing.

CCS CONCEPTS

Social and professional topics → Computing education; Computational thinking; Computing literacy; K-12 education

KEYWORDS

Programming; Computational Thinking; Pedagogy; Primary; Children's Literature

ACM Reference format: Sarah Twigg, Lynne Blair and Emily Winter. 2019. Using children's literature to introduce computing principles and concepts in primary schools. In Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE '19), October 23–25, 2019, Glasgow, UK. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/...

1 INTRODUCTION

Since 2014, the reformed national curriculum for England has mandated that computer science be taught in schools for all 5-16 year olds (Key Stages 1-4) (DfE, 2013). The curriculum's overall aim is to teach children to understand the fundamental principles and concepts of computer science and to develop problem-solving skills using computational thinking concepts such as abstraction, logical reasoning, algorithms, pattern recognition and evaluation. This paradigm shift goes beyond the digital literacy skills required to 'use' computer-based technology, moving into computational thinking skills – and creativity – required to "understand and change the world" (Caldwell, 2017).

In primary level classrooms, children's literature is a familiar and regularly used resource. Particularly at key stage 1 (ages 5-7), such literature is often picture heavy and repetitious. The narrative structures of sequencing, repetition and selection are typical features of children's stories – structures that are shared with programming languages. By linking children's literature and the introduction of programming, we offer teachers of key stages 1-2 a creative approach to teaching introductory computing and computational thinking skills to young children.

2 SELECTING CHILDREN'S BOOKS

A working set of children's literature was selected by reviewing the 'The Book People'[1] website (a bookseller promoted in many of our local schools), looking for age-appropriate (up to age 7) classic picture books. This yielded a set of 75 books, which reduced to 50 once book collections and duplicates were removed. Each book was coded based on whether it exhibited sequencing, selection and repetition (the programming constructs explicitly included in the English KS1-2 curriculum). To perform this coding, each book was read and the storyline considered as a whole to identify all relevant programming constructs.

The following guidelines describe the relationship between children's literature and these 3 programming constructs:

Sequencing – a list of events to be followed in a logical order or plot stages.

- All 50 books (100%) illustrated some form of sequencing.

[1] http://thebookpeople.co.uk

Repetition – at least one example of a pattern of repeated dialogue, actions, environment, etc. For example, a story may go through different contexts where the same dialogue and/or action is repeated in each context.

- 46% of books illustrated some form of repetition.

Selection – at least one example of a choice of dialogue, actions, environment, etc. For example, where the dialogue follows a repeated pattern but changes occur dependent on the context, or where the current context of the storyline is examined to test whether to continue or whether the desired goal has been reached (terminating condition).

- 36% of books illustrated some form of selection.

Overall, a total of 32% (n=16) of the chosen type of books (classic picture books for under 7s) illustrated all three constructs. A table containing this full set of books is available online[2]. Given that a typical primary school has many children's books, our experience from local schools and teachers indicates that the guidelines presented above and the illustrative examples below support teachers to identify other books that can be used to introduce computing concepts to their classes.

Illustrative examples from 3 books:

Dear Zoo, R. Campbell, Penguin, 1985.
Sequencing: After a boy writes to a zoo for a pet, the zoo sends multiple pets one at a time. Each time the pet is not what the boy expected, he sends the pet back. The sequence ends when the zoo sends the boy a puppy.
Repetition: As the boy encounters each of the pets sent to him, the following dialogue repeats:
Say – "They sent me a"
Say – <item in a list of pets>
Say – <the boy's reaction>
Say – "So I sent it back"
Selection: The responses vary depending on the pet sent. If the pet is not a puppy, the above dialogue is repeated. If the pet is a puppy, the following dialogue terminates the story:
Say – "So they thought very hard, and sent me a"
Say – "Puppy"
Say – "He's perfect!"

We're Going on a Bear Hunt, M. Rosen, McElderry, 1989.
Sequencing: The characters set out on a bear hunt and go through six different environments followed by an encounter with a bear! After this they make their way back home through each of the different environments.
Repetition: A dialogue is repeated in each environment:
Say – "We're going on a bear hunt"
Say – "We're going to catch a big one"
Say – "What a beautiful day"
Selection: The above dialogue is extended with additional comments that vary depending on the environment.
If they're in a grass field, say – "Grass! Long wavy grass!"
If they're crossing a river, say – "Dive in! Splash splosh!"

The Very Hungry Caterpillar, E. Carle, Collins, 1979.
Sequencing: A hungry caterpillar is on a hunt for food. After an initial 'set-up' day on Sunday, each subsequent day of the week he finds a new food to eat, and continues the hunt until no longer hungry. The caterpillar then builds himself a cocoon and stays inside it for more than two weeks when it changes into a butterfly.
Repetition: A dialogue is again repeated, but this time the exact dialogue varies depending on the day of the week (that may be represented by a list with items Monday through to Saturday):
Say – "On <day of week>, he ate through <number> <food item(s)>, but was still hungry".
Selection: The dialogue may be generated by selections that use the day of the week to determine the appropriate number and type of food item(s). For example,
if <day of week> = Monday then
  Say - "On Monday, he ate through one apple, ..."
else if <day of week> = Tuesday then
  Say - "On Tuesday, he ate through two pears, ..."

3 RAMP: Read, Act, Model and then Program

Having selected appropriate children's books, the RAMP approach aims for a gradual build-up of subject knowledge and skills, initially through unplugged activities (Bell, 2009; Caldwell, 2017) before moving on to programming tasks.

Read: Read through a story, asking questions about what is happening and introducing the learning objective(s), for example identifying key computing terminology to be introduced.

Act – Act out a story, including watching for interesting patterns of behaviour. Pupils may identify with different roles within a story. For example, for Dear Zoo, roles might include a zookeeper, a postal worker, the boy, and different animals. Watch out for repeated patterns, and what changes during each repetition - including what causes or triggers these changes. Repeat/ affirm key terminology to ensure the link to computing is explicit.

Model – Start to model (or design) the code using unplugged activities. Starting with a pack of laminated printouts of lines of code (Scratch vector blocks), the class may initially be asked to construct a long sequence of events that model the narrative of the story. Alternatively, for a differentiated activity for pupils with low reading ages, pupils can arrange images from the book into a sequence.
The class can then be asked to identify patterns in the sequence and start working to complete a design that identifies blocks of repeated code and choice points in the story. A large template sheet[3] may be used to assist with the algorithm for this design. Depending on the individual classroom environment, differing level of detail may be offered on this template to support differentiated activities for differing abilities. Throughout this

[2] https://community.computingatschool.org.uk/resources/5681/single

[3]https://sites.google.com/view/aprogrammerstale

stage, a pupil can act as a 'computer', reading through and following the design, predicting behaviour, and debugging as necessary, i.e. determining if any deviations from the story are intentional or are bugs!

As can be seen from the text above, this stage offers a natural opportunity to start to introduce elements of computational thinking terminology[4], such as creating *algorithms*, making judgements about the level of detail to be included (*abstraction*), identifying *patterns* of behaviour, *predicting* behaviour, *debugging*, etc.

Program – The Read, Act and Model stages described above are intended to aid the transition to programming through unplugged and design-based activities. To offer additional scaffolding and differentiation, we propose that pupils are further supported in this programming step through examples of the code blocks and structures that they are expected to use/ find. To offer examples of this, sample teaching resources[3] were developed for a selection of the children's books. These have been used and evaluated in a small number of focus groups, interviews, classroom lessons and code clubs.

4 Experiences and Evaluation

14 primary school teachers were recruited through a Computing At School[5] regional centre, to take part in a range of activities including initial focus groups and semi-structured interviews, classroom trials and code clubs, and follow-up semi-structured interviews.[6] The data from these activities was thematically analysed using a mixture of inductive/ deductive analysis. Themes included suggested adaptations, considerations of applicability and appropriateness in different contexts, implementation in the classroom, scaffolding and differentiation techniques, levels of teacher intervention required, and issues/concerns regarding transitioning to Scratch.

Overall feedback indicated that the teachers were happy with and excited about the use of children's books to teach computing concepts, and thought it would be appropriate at different ages across key stages 1-2 (especially years 3-6), with teacher T6 commenting: *'I did the activities with year five and six and said I know these books are a bit old for you but you're going to be coding it to show my year ones and two'.* The teacher explained that the children were all engaged in the stories and loved the nostalgia of revisiting the stories, despite comments from other teachers in the school stay '*you're reading them a story, you do know they're in year 6*'! She further explained that '*we had six weeks of a lot of fun'*.

Teachers were enthusiastic about the opportunity for a whole class activities and children acting out roles. For example, teacher T3 enthused that: '*I'd have children being these things [roles]. And then you've got one who's being the computer… 'Is that the right place for you to be? What do they need to do? They've got to go back now, move on to the next one. Is that right? No, that's not the one either. You've got to go back'. And then sort of have the class sort of directing so they're all involved'*. This was echoed with the code club experience: *'using the example of Dear Zoo, the computer role-playing activity led to several moments where the 'computer' got stuck and the other children participated to describe the problem, then debug and fix the algorithm'.*

T3 commented on the ideal opportunity to introduce computational thinking terminology: '*They love it, they love it. I've got another big computing word for you kids. Are you listening? Make sure, sit up, 'cause this is important... Loads of people won't know what this means but you know what you're smart enough, you can hear this'*. T6 echoed this regarding the early introduction of terminology: '*We do here, we do algorithms and the children can tell me it's a set of instructions for a purpose. We use the real deal because ... it's like teaching them another language. There's no point teaching them one version and then going actually, we're going to change all the names now just to confuse you'.*

Differentiation was raised as a significant component of classroom teaching by the majority of teachers. T3 explained: *'I gave them all the Scratch. We ... self-differentiate, so we set challenges and the children choose the one that they think works for them. ... So we have fix it, revisit it, and push it. So fix it is for if you're not very sure. Revisit just to reinforce, and push-it if you think 'yeah, I'm up for a challenge'. And then I just took bits out for the challenges that are harder. So for fix it I just took out the order of the different things'* [this meant leaving the structure of the Scratch program, but taking out the contents of the repeat/ conditional blocks].

Two teachers commented on potential difficulties, especially for non-specialist teachers, with the transition from the modelling/ design stage to programming in Scratch. T4 commented that the perceived level of difficulty of the programming stage would depend on the confidence and experiences of the teacher, and suggested focusing on the first three stages (Read, Act, Model): *'I think you'd be meeting all the objectives - it would be a brilliant lesson, and I love it all, but <u>not</u> have to put it into Scratch.'* T2 reflected similarly: *I don't have a massive computer science, programming, coding background at all. But it's just something that I'm happy to tinker about with. Looking at the kind of the materials and things… I think it would scare some people who are not specialists. I still think they would struggle with some*

[4] For example, Barefoot Computational Thinking poster: https://www.barefootcomputing.org/resources/computational-thinking-poster, and CAS Computational Thinking - A Guide for Teachers: https://community.computingatschool.org.uk/resources/2324/single

[5] http://www.computingatschool.org.uk/

[6] Participant breakdown: Teachers: 11F, 3M; 2 KS1, 8 KS2, 4 mixed KS1/2 (including 1 special needs specialist and 2 cross-school computing specialists). Classroom trials: all KS2 classes. Code clubs: mixed KS1/2 groups.

elements of the language' [where language applied to both Scratch and computational thinking terminology].

However, two teachers commented on particularly positive experiences with the Scratch activities. One teacher T9 paired children of mixed ability and explained, '*We had one boy who's very dyslexic, who can't read or write, so he made his very graphical. He had the caterpillar moving along the screen and the apple would slowly disappear as he ate it.*' Further to this, T6 commented: '*We've got a severely autistic boy, who loved this. He thought it was amazing: 'look, look, look, look, look, it can do this' and normally when you get him to sit still it's like 'I don't want to, I don't want to, I don't want to' … [but] he was completely the opposite. I think because you can give those different levels and because I put them in mixed ability groups, he could be like 'I want to do it in pictures' and their partner saying 'well can I put this bit on top, with a bit of writing.''* The context here was that both T9 and T6 had higher confidence levels with Scratch and were teaching classes that had previously been introduced to Scratch.

5 Related work: other story-based work

There are also a number of story-based approaches that are now available that teach computer science concepts. For example, in Hello Ruby (Liukas, 2015), the central character (Ruby) goes on an adventure and, as she meets new friends along the way, encounters puzzles and problems that help to develop programming skills as they are creatively explored and solutions developed.

Moving closer to our own children's literature-based approach, Once Upon an Algorithm (Erwig, 2017) illustrates computational layering in daily life activities and in familiar stories. Through such activities and stories, computing concepts such as algorithms, recursion, abstraction, data types/ data representation and complexity are highlighted and explained. This interesting resource differs from our own research in the level of computing concepts targeted, with the work of Erwig targeting concepts that are introduced later on in the education process, for example Key Stages 3 and 4.

Research from an Italian middle school (Di Vano, 2011) explored using nursery rhymes to identify repeated patterns of behaviour in the structure of the rhyme, possibly also identifying a prologue and epilogue if appropriate. Initially nursery rhymes are gathered from pupils' collective experiences and their structure analysed. A set of activities including a 'ladybug' application and the Logo programming language, lead to pupils developing programs to automate the generation of simple (typically cyclical) nursery rhymes.

6 Conclusions

It is increasingly important to develop a clear pedagogy and associated set of resources that support the teaching of computing in primary schools, particularly resources that are low (or zero) cost and are familiar or intuitive for teachers and pupils alike. This paper has discussed the potential of a 4-stage approach for key stages 1-2, that makes use of creative story-based pedagogy to introduce the core constructs of sequencing, repetition and selection, plus computational thinking concepts including pattern matching, abstraction and algorithms.

The first 3 stages of this approach - Read, Act and Model - received very positive feedback from a set of 14 teachers and their classes, with appreciation for the scaffolding and differentiation opportunities. However, it was interesting that the 4th stage - Program - received mixed reactions. Whilst no problem existed for the study participants themselves, two teachers raised concerns when considering how other non-specialist colleagues might approach the transition to Scratch. This needs to be the subject of further study to identify whether our approach can be further developed to support the transition from 'RAM' to 'P', or whether programming (or fear of the unknown) is a more fundamental underlying problem that needs addressing separately.

Finally, we hypothesise that this creative approach has further benefits related to diversity, and our studies have already seen positive signs regarding students with learning difficulties. Future work will investigate these benefits in more detail.

ACKNOWLEDGMENTS

This work was partially supported by the Computing At School Regional Centre – NW (Lancaster). We are immensely grateful to all our local CAS colleagues for their encouragement and time spent helping to develop this research, and their contagious enthusiasm throughout – thank you.

REFERENCES

T. Bell, J. Alexander, I. Freeman and M. Grimley. 2009. Computer science unplugged: school students doing real computing without computers. New Zealand Journal of Applied Computing and Information Technology. 13(1), 20–299; see also the CS Unplugged website: https://csunplugged.org

H. Caldwell and N. Smith [eds]. 2017. Teaching Computing Unplugged in Primary Schools: exploring primary computing through practical activities away from the computer. Sage Publications Ltd.

DfE (Department for Education). 2013. Statutory guidance. National curriculum in England: computing programmes of study. Retrieved 15/6/18 from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

M. Erwig. 2017. Once Upon an Algorithm: How Stories Explain Computing. MIT Press.

L. Liukas. 2015. Hello Ruby: Adventures in Coding. Macmillan.

D. Di Vano and C. Mirolo. 2011. Computer Science and Nursery Rhymes - A Learning Path for the Middle School. ITiCSE'11 (Innovation and technology in computer science education), pp 238-242.