Code Club: Bringing Programming to UK Primary Schools through Scratch

Neil Smith
Centre for Research in Computing
The Open University
Milton Keynes, UK
n.smith@open.ac.uk

Clare Sutcliffe
Code Club
London, UK
clare@codeclub.org.uk

Linda Sandvik
Code Club
London, UK
linda@codeclub.org.uk

ABSTRACT

Code Club is a network of after-school programming clubs for primary (US: elementary) schoolchildren, run by technically-competent volunteers in conjunction with (generally technically-unskilled) teachers. The main motivation of Code Club is to inspire children with a sense of fun and achievement for programming and digital creativity. This paper reports on the first year of Code Club in 1000 UK schools. The results were extremely positive, but some children had difficulty understanding the concepts behind the projects.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education; K.3.1 [**Computers and Education**]: Computer Uses in Education; D.1.7 [**Programming Techniques**]: Visual Programming

General Terms

Active Learning, K-12 Instruction

Keywords

introductory programming, elementary school, primary school, K-12

1. INTRODUCTION

Eric Schmidt's 2011 MacTaggart speech [17] galvanised an overhaul of the UK's K-12 Computing education provision. He pointed out that a generation of UK schoolchildren was not receiving a substantive Computing education. Prior to 2011, ICT (Information and Communication Technologies) education in UK schools consisted mainly of lessons in how to drive office automation software with little, if any, programming or other technical expertise being developed. However, experience with tools such as Scratch [15] and Storytelling Alice [14] show that primary (age 5–11) schoolchildren can engage profoundly with computing concepts.

Seeing that primary schoolchildren were being abandoned by the current ICT education provision, two of the current authors decided to do something about it. In 2012, they founded Code Club, a network of after-school clubs in primary schools. Code Club is intended to provide an easy and *fun* introduction to digital creation that will motivate children to pursue Computing throughout their school careers and beyond. In the process, the children will learn the basics of programming and fundamental CS concepts.

Meanwhile, the proposed new Computing curriculum [9] is following Code Club's direction. The new government-mandated national curriculum includes major changes, such as the introduction of algorithms to children under 7 years old. However, it will not be in place before September 2014, and there remain significant challenges with its deployment, including the lack of expert teachers [10].

2. CODE CLUB'S APPROACH

The idea of Code Club formed after a hack day in early 2012. Hack days are events where designers and developers come together to produce a new technology product. One key characteristic of these events is that they are *fun* for the participants. Anecdotal questioning of various industrial IT practitioners indicates that *fun* was a fundamental reason for them progressing in the industry: the fun of meeting a challenge, of inventing a new product, of understanding a difficult concept. For many current IT practitioners, their passion was ignited at school, often during after-school clubs. These impressions are borne out by Schulte & Knobelsdorf [18], who indicate that prior experiences of design activities (including programming) encourage people to engage more fully in Computing at university, as well as fostering a more positive view of CS. Many practitioners are keen to contribute to bringing their passion to a new generation, but lack the pedagogic skills, child-protection accreditation, and access to children to do so effectively.

This desire must be seen in the changing context of Computing in UK schools. Since Schmidt's speech, the UK government and other organisations have started a process of major changes in how Computing is taught in school. This involves both updating the curriculum and training the teachers; many years of neglect of Computing in schools had left little Computing expertise among the teaching staff with many non-specialists now teaching the subject [10]. Much of this effort is directed at secondary (age 11–18) schools. However, primary schoolchildren are capable of productive engagement with Computing concepts [16, 15, 14, 2, 6] but the problem of teacher expertise is even more acute: few primary school staff have a technical background and therefore lack both the skills and confidence to bring programming to their pupils.

Therefore, the obvious solution was to provide a vehicle by which IT professionals could productively engage primary schoolchildren in fun digital design and implementation activities. If we could excite children's interest in Computing, they would absorb the basics of CS by engaging with the material. However, there was no way that a pair of outsiders to education could usefully influence the ongoing governmental plans for the revision of CS education, especially in the short term. Code Club was the solution to this quandary.

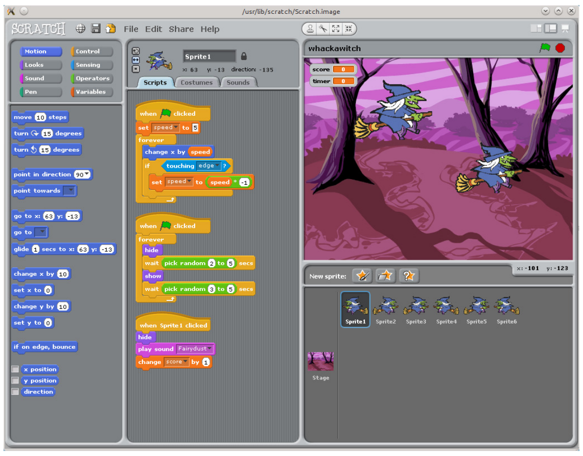
The Code Club model is an after-school club, run by an technically-adept volunteer in tandem with a teacher in the school. The volunteer brings the necessary technical expertise to the club, while the teacher provides additional pedagogic support and "crowd control" expertise. Code Club provides a set of materials for the children to work through, guiding them to create a rich multimedia project, generally a game (games are recognised as strongly motivating children to engage with programming concepts [8, 22, 1]). Code Club also provides volunteers with an easy route to child-protection vetting via the UK government's existing STEM Ambassador programme. Code Club does not technically vet the volunteers or provide any technical training. This does not seem to cause any issues, especially as the programming is straightforward for anyone with programming experience.

Code Club's approach has limitations, mainly due to Code Club's self-selecting nature. Initially only the more forward-thinking and technically-aware schools signed up for Code Club, though this is becoming less of an issue as more clubs sign up. Code Club can only run if the school already has sufficient PCs available and if there is a committed volunteer nearby. As an after-school club, attendance by children is both voluntary and limited, meaning that only a few of the most interested children will attend. However, we consider that Code Club is the best we can do in the short term.

In April 2012, Code Club was announced and the news quickly spread. Within weeks, over a thousand volunteers and two hundred schools had expressed an interest in setting up a Code Club. A steering committee of nine (including all of this paper's authors) was assembled to produce and oversee the Code Club materials. Twenty-two schools, with enthusiastic and technically-skilled teachers, were selected to pilot Code Clubs at the end of the 2012 summer term. Code Club opened to all schools in September 2012. 250 clubs were active by the end of 2012; by October 2013, over 1300 clubs were registered. The international branch of Code Club, Code Club World, launched in June 2013 with 60 clubs registered outside the UK by October 2013. News of Code Club continues to spread mostly by word of mouth, with additional exposure from articles in widely-read publications (e.g. *Wired UK* [12, 21], national newpapers *The Guardian* [7] and *The Indepenent* [11]) and highly public awards [20].

3. SCRATCH

We chose Scratch [15] as the first programming environment. This was due to its known ease of use by primary schoolchildren, the range of media resources easily available, and support for saving and sharing projects via the Scratch website. Scratch can be used for a variety of projects, often animations, simple games, and interactive stories.



```
when clicked
set speed to 5
forever
    change x by speed
    if touching edge ?
        set speed to speed * -1

when clicked
forever
    hide
    wait pick random 2 to 5 secs
    show
    wait pick random 3 to 5 secs

when Sprite1 clicked
hide
play sound Fairydust
change score by 1
```

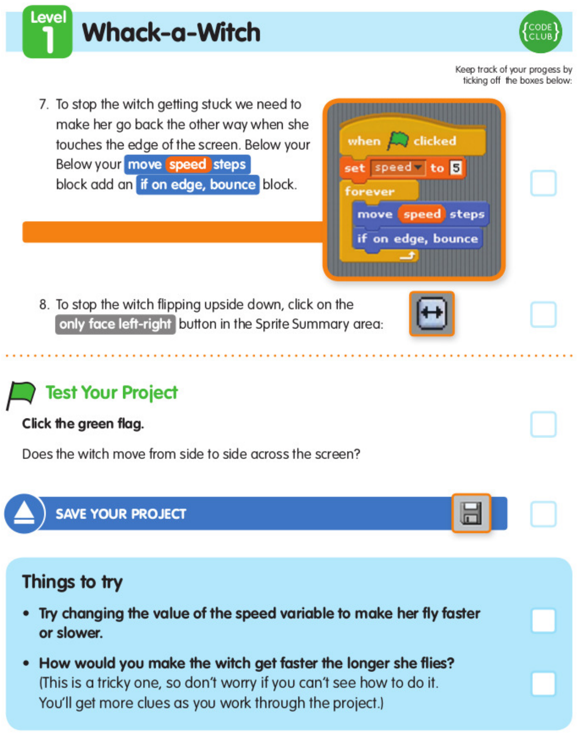
Figure 1: The Scratch environment.

Scratch is a visual programming environment (see Figure 1). It features a set of *sprites* that move around a world (called the *stage*). Sprites have one or more costumes and the stage has one or more backgrounds, which define their appearance. Both sprites and the stage can have their own behaviours, defined by *scripts*. Sprites and the stage can have multiple scripts which run concurrently and asynchronously. Scripts are formed by clicking together pre-defined code *blocks*, selected from a number of themed palettes. Some blocks perform operations such as moving sprites, changing their appearance, changing costumes, playing sounds, detecting collisions, and so on. Other blocks expose much information about sprites, such as position and direction. More blocks implement standard control structures, including a variety of loops, if-then-else selections, and pauses. Scripts are triggered by either user input (mouse clicks and keyboard presses), broadcast messages sent from other scripts, or the "green flag" button on the Scratch UI; the latter is often used to start a project. Programmers can create variables with with scope either global or local to a sprite. Sprites can communicate either through global variables or broadcast messages.

We have created further projects using HTML5 and Python, though these projects have not yet received sufficient use to form reliable conclusions on their reception.

4. PEDAGOGY

There are several existing programmes of study for teaching children the basics of computing and programming (e.g. [6, 2]) but none was suitable for Code Club. The published curricula generally assume that the teaching will be delivered during a formal, class-based, teacher-led programme of study, lasting from a term to a year or more. Code Club runs as a ten-week after-school club with one hour sessions (including allocating time for children to arrive and pack up). Existing programmes of study assume that children will attend virtually all the sessions. Attendance at after-schools clubs is more erratic. The emphasis in the existing programmes is on the skills and behaviours that the children should acquire. The intent of Code Club was to create a sense of fun and excitement in its child participants and inculcate a desire to engage with Computing, with skill de-



Level 1 Whack-a-Witch

Keep track of your progess by ticking off the boxes below:

7. To stop the witch getting stuck we need to make her go back the other way when she touches the edge of the screen. Below your Below your move speed steps block add an if on edge, bounce block.

```
when [flag] clicked
set speed to 5
forever
    move speed steps
    if on edge, bounce
```

8. To stop the witch flipping upside down, click on the only face left-right button in the Sprite Summary area:

Test Your Project

Click the green flag.

Does the witch move from side to side across the screen?

SAVE YOUR PROJECT

Things to try

- Try changing the value of the speed variable to make her fly faster or slower.
- How would you make the witch get faster the longer she flies? (This is a tricky one, so don't worry if you can't see how to do it. You'll get more clues as you work through the project.)

Figure 2: A sample project step.

velopment a secondary concern.

We adopted a project-based pedagogy [5]. Each project creates a specific Scratch application, such as a game or a drawing package. The projects are broadly similar to others designed for a similar age group (e.g. [13]), though earlier projects are smaller to allow them to be completed in a single club session.

The first term's activity consists of nine projects, with each project expected to take one or two sessions. Each project is divided into a series of steps (see Figure 2 for an example). The first few steps are directed with a restricted scope and aim to give the child a basic, working application within around twenty minutes. Subsequent steps extend the basic project. Some steps are directed and add specific pieces of additional functionality. Some steps describe an extension for the project but leave its implementation to the pupil. Others are more open ended, inviting pupils to explore how they can customise or extend the application. As the projects progress throughout the term, the level of direction decreases and pupils are increasingly asked to design and implement their own ideas and extensions. The first term concludes with a "capstone" project where pupils design and build their own game, drawing together what they have learnt from previous projects.

All project steps include specific prompts for pupils to test their projects and to save their work. The projects make extensive use of Scratch Cards [19], quick summaries of often-used functionality, to emphasise skill and knowledge transfer across projects.

The second term of Scratch projects is organised as three larger projects (an animated monster, a suite of musical instruments with recording and playback, and a platform game), each intended to be completed over three to four weeks. These projects contain a substantial design component, with at least one session dedicated to the students designing their projects. These projects contain little sample code for the children to copy. Instead, they are expected to create their own scripts, prompted by what they have achieved already or suggested samples in the projects.

The project-based approach has several advantages for Code Club. First, it emphasises work on authentic problems. This motivates the children to engage with the material and hence learn the programming skills they need, almost as a by-product. Second, pupils produce a discrete artefact, which gives the children a sense of achievement as well as the ability to show off the products of their labours to friends and family (assisted by Scratch's ability easily to share projects via its website). Third, it allows for much more flexibility in attendance; the material allows for children to miss sessions at different times and still be a full participant on their return. Fourth, it gracefully allows for differentiation across pupils of different ability, with pupils able to progress through projects at the speed most appropriate to them.

5. THE FIRST YEAR

We evaluated the adoption and progress of Code Club with two surveys, performed by club leaders completing online questionnaires. We received 150 responses (of approximately 280 clubs) to the first survey in December 2012, and 130 responses (of approximately 800 clubs) to the second survey in June 2013. We also examined a sample of 22 completed capstone projects, where clubs made them available to us.

5.1 Survey Results

The first survey concentrated on the demographics of the club members; the second on the results of up to a year of Code Club.

The first survey revealed 2233 children enrolled in 150 clubs, giving an average of 14.9 children per club (Figure 3). 40% of club members were girls. Club sizes ranged from one to eighty children; we assume the latter were where Code Club was used in lessons in schools across year groups. Extrapolating this across all clubs meant that in December 2012 there were over 5800 children in Code Clubs and over twelve thousand children in Code Clubs by July 2013. 77% of club members had no prior programming experience; 20% had used Scratch before.

By the end of the year, children had completed an average of six projects each (Figure 4), though some clubs have completed ten or more projects. This low number is expected as many clubs started at various points throughout the year.

We also asked the club organisers to rate the children's confidence with different programming concepts (Figure 5). "Confidence" was reported on a ten-point Likert scale and respondents were asked to give a rating for the club overall. Several respondents commented that children in their club had a range of confidence in these concepts. Despite this, the ratings are encouraging, with most clubs showing that most people are at least reasonably confident using a variety of programming concepts. Children are reported as being less confident with debugging than with any of the programming concepts. These confidence levels are borne out in the analysis of the projects, below.

We asked respondents to rate how many of their club

Figure 3: Code Club sizes.

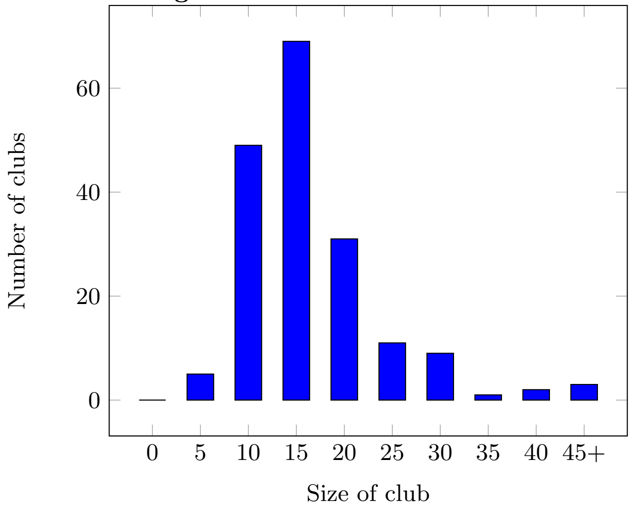


Figure 4: Number of completed projects.

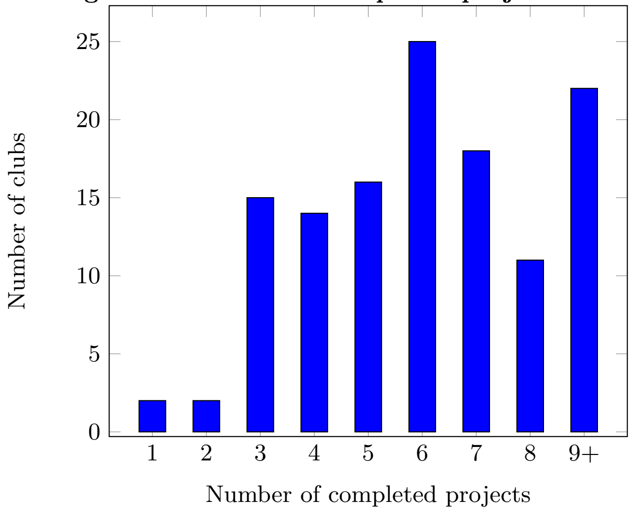
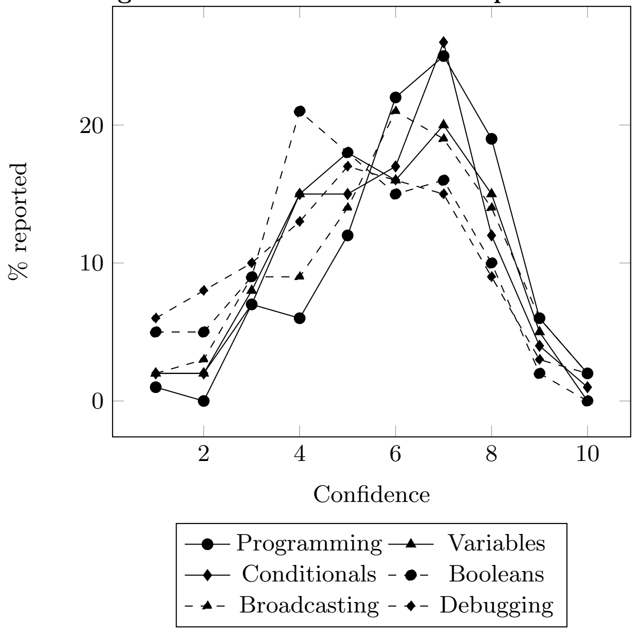


Figure 5: Confidence with concepts.



members engaged in programming or some other form of digital making outside Code Club. The results were surprisingly evenly split, with one third saying that less than one third of their club members engaged in such activities, one third said about half did, and one third saying that more than two thirds did some form of digital making outside Code Club.

Finally, we asked respondents an open-ended question for further comments. Most were positive, with many mentioning minor errors in the project sheets. The most common substantive comment was about the transition from following instructions to open-ended tasks. Many respondents commented that children were able to follow the early, guided instructions but were unable to apply any knowledge to unguided challenges. This will require a revision of our pedagogy to give more structure to these challenges, such as Bagge's Scratch projects [3] which show the blocks to be used in a challenge step without showing how they should be assembled.

5.2 Project Analysis

We analysed a sample of 22 of the final 'build your own game' projects to see what programming concepts were contained within them. These projects were designed and built by the children themselves, rather than following any instructions from a Code Club sheet. We analysed the programming concepts used in these projects (see Table 1), noting whether the concepts were properly used or just attempted. We made no judgements about the presence of bugs or the quality of the gameplay! Many of the final projects were based on earlier Code Club projects, but all contained their own variations in program code. Some projects were completely novel, including a two-player game of tennis and a wizardly duel. As can be seen in this table, most of the projects appropriately used user input (often as a game control), control statements, parallel execution (both multiple sprites and multiple, simultaneous execution of scripts

Table 1: Concepts used in 'build your own' projects.

Concept	Correctly used	Incorrectly used
Variables	13	0
User input	21	1
Broadcast within a sprite	1	0
Broadcasting between sprites	13	2
Control statements	20	1
Boolean connectives	7	0
Parallel execution	22	0
Detecting state	20	0

within the same sprite), and detecting and acting on states in the project (normally collision detection).

The more tricky concepts of variables and message passing were used in over half the projects, mostly appropriately. Where variables were used, they were always used appropriately, normally as score counters or game timers. Where variables were not used in the project, they were generally not needed. Message passing was often used to allow sprites to communicate game events, such as losing a life.

The distributed nature of Code Club, and the flexibility of timetabling individual projects, made it impractical for us to observe a representative sample of children developing their own projects in a club. Observation of children in our own clubs and at other events indicates that children are generally comfortable with most of the concepts listed in Table 1. Children seemed to start the design of their projects from considering the end result and bringing in programming concepts as appropriate to that end. We rarely saw children wanting to achieve some functionality and not knowing how to approach it. We did not perform more in-depth exercises to gauge children's design processes, such as "think aloud" design sessions.

6. CONCLUSIONS AND FUTURE WORK

Code Club is a new network of after-school clubs aimed at introducing primary schoolchildren to the joy of programming. Surveys of Code Clubs show that Code Club is achieving its objectives. The children have fun. They successfully engaged with a range of digital design and implementation tasks, creating several artefacts along the way. The children coped remarkably easily with some difficult programming concepts. Children were able to show off their creativity within the projects and many children were eager to continue developing projects outside Code Club. Code Clubs continue to spring up across the UK and internationally.

Code Club continues the work of other projects that bring programming experience to schoolchildren [14, 15, 8]. Code Club is distinct because it is addressed exclusively at primary schoolchildren, and it has a national, and increasinly international, scope. It is an initiative that answers the call to arms of how to update the Computing curriculum [4].

Code Club continues to expand. We are still aiming to have Code Club in 25% of UK primary schools by the end of 2014. Additional project types are being developed, using HTML5 and Python. After a period of rapid expansion and development, we intend to consolidate our progress with the project materials, revising them as necessary to improve their ease of use and to smooth the transition from following instructions to designing your own code. We will continue to monitor and evaluate Code Club across these changes.

7. REFERENCES

[1] J. C. Adams and A. R. Webster. What do students learn about programming from game, music video, and storytelling projects? In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 643–648, 2012.

[2] C. at School Working Group. *Computer Science: A Curriculum for Schools.* Computing at School, 2012.

[3] P. Bagge. Junior Computer Science Scratch Projects, 2013. Available from http://code-it.co.uk/year4/scratchprojects.html

[4] V. Barr and C. Stephenson. Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads*, 2:48–54, 2011.

[5] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist*, 26(3/4):369–398, 1991.

[6] K. Brennan, M. Chung, and J. Hawson. Scratch curriculum guide draft, 2011.

[7] J. Butterworth. Primary Code. In *The Guardian*, 15 July 2013. Available from http://www.theguardian.com/science/life-and-physics/2013/jul/15/programming-schools-children

[8] M. Carbonaroa, D. Szafronb, M. Cutumisub, and J. Schaefferb. Computer-game construction: A gender-neutral attractor to computing science. *Computers & Education*, 55(3):1098–1111, 2010.

[9] Department for Education. National curriculum review: new programmes of study and attainment targets from September 2014, 2013. Available from https://www.gov.uk/government/consultations/national-curriculum-review-new-programmes-of-study-and-attainment-targets-from-september-2014

[10] S. Furber. *Shut Down or Restart? The Way Forward for Computing in UK Schools.* The Royal Society, 2012.

[11] R. Garner. Welcome to Code Club: UK programme that teaches children computer coding goes global. In *The Indepenent*, 24 July 2013. Available from http://www.independent.co.uk/news/education/schools/welcome-to-code-club-uk-programme-that-teaches-children-computer-coding-goes-global-8730806.html

[12] D. Geere. Afterschool 'Code Clubs' planned to teach kids programming. In *Wired.co.uk*, 17 April 2012. Available from http://www.wired.co.uk/news/archive/2012-04/17/code-club

[13] LEAD Project. *Super Scratch Programming Adventure!* No Starch Press

[14] C. Kelleher, R. Pausch, and S. Keisler. Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Concerence on Human Factors in Computing Systems*, pages 1455–1464. ACM, 2007.

[15] J. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk. Programming by choice: Urban youth

learning programming with scratch. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 367–371. ACM, 2008.

[16] M. Petre and B. Price. Using robotics to motivate 'back door' learning. *Education and Information Technologies*, 9:147–158, 2004. (Draft available from http://mcs.open.ac.uk/bp5/papers/2004-eit/2004-EIT-Robotics-Backdoor.pdf)

[17] E. Schmidt. *MacTaggart Lecture*. Edinburgh Television Festival, 2011.

[18] C. Schulte and M. Knobelsdorf. Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. In *Proceedings of the third international workshop on Computing education research*, pages 27–38, 2007.

[19] Scratch. Scratch cards, 2012. Available from http://info.scratch.mit.edu/Support/Scratch_Cards.

[20] Talk Talk. Digital Heroes Award, 2012. Information available from http://www.talktalk.co.uk/digitalheroes/winners.php

[21] N. Tufnell. Code Club Robo-Boogie competition asks kids to invent a robot dance. In *Wired.co.uk*, 15 November 2013. Available from http://www.wired.co.uk/news/archive/2013-11/15/kids-get-coding-with-robo-boogie

[22] L. Werner, S. Campe, and J. Denner. Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 427–432, 2012.