Teaching Computational Thinking by Playing Games and Building Robots

Jonathan Francis Roscoe
Computer Science, Aberystwyth University
jjr6@aber.ac.uk

Stephen Fearn
Infinity, Aberystwyth University
sdf@aber.ac.uk

Emma Posey
Technocamps, Aberystwyth University
emma.posey@technocamps.com

***Abstract*—Computing in schools has gained momentum in the last two years resulting in GCSEs in Computing and teachers looking to up skill from Digital Literacy (ICT). For many students the subject of computer science concerns software code but writing code can be challenging, due to specific requirements on syntax and spelling with new ways of thinking required. Not only do many undergraduate students lack these ways of thinking, but there is a general misrepresentation of computing in education. Were computing taught as a more serious subject like science and mathematics, public understanding of the complexities of computer systems would increase, enabling those not directly involved with IT make better informed decisions and avoid incidents such as overbudget and underperforming systems.**

We present our exploration into teaching a variety of computing skills, most significantly "computational thinking", to secondary-school age children through three very different engagements.

First, we discuss Printcraft, in which participants learn about computer-aided design and additive manufacturing by designing and building a miniature world from scratch using the popular open-world game Minecraft and 3D printers. Second, we look at how students can get a new perspective on familiar technology with a workshop using AppInventor, a graphical Android programming environment. Finally, we look at an ongoing after school robotics club where participants face a number of challenges of their own making as they design and create a variety of robots using a number of common tools such as Scratch and Arduino.

I. Introduction

Modern computing ubiquity has led to basic computing skills (or "digital literacy") becoming an innate ability of most young people. Unfortunately, the UK education system has failed to keep up to date and upskill students beyond the essentials. Published in January 2012, the Royal Society's report "Shutdown or Restart?" [1] stated the need for recognition of computer science as a rigorous academic discipline of great importance in schools and described the current delivery of computing education as "highly unsatisfactory".

We use the term "digital literacy" [2] to refer to common and basic computer use, such as typing, word processing and web browsing. These skills are taught under the vague term "ICT" and are fast becoming redundant as a result of ubiquitous exposure to computing. More recently, the term "computing" is being used to replace ICT and attempt to clarify the scope and nature of the subject."computing" has a much broader scope and shifts focus from **using** computers to **understanding** how computers work and behave - a topic of importance to an increasing number of the population, regardless of their career or interests.

For those not necessarily employed in the computing industry digital literacy skills improve communication through email, ease learning through online resources and provide new marketing opportunities. Beyond that understanding of the broader issues of computing could enhance the ability to innovate business operations, develop new tools and understand issues previously considered to be exclusively within the arcane domain of the dreaded IT department.

Thanks to the maker community [3], people are shifting from being users of technology to creators - leading to a new era of innovation by hobbyists and professionals alike. Advanced computing skills open up a world of opportunities for developing new tools, toys and experiments.

Technocamps is one project that aims to improve computer science education for 11-19 year olds through a variety of engagements with partnered universities across Wales. In this paper we look at the importance of going beyond basic digital literacy and some of the engagements designed to teach a core concept in computer science - "computational thinking".

A. Computational Thinking

Computational thinking is a fundamental problem solving technique that has applications beyond computing [4], [5] and is considered by many to be a fundamental life skill [6]. Although the term is open to some interpretation, generally accepted key principles of computational thinking include:

- **Abstraction** - solutions are often not straightforward and require multiple levels of thought and application
- **Logical analysis** - making the most of the information you have to deduce solutions
- **Algorithmic thinking** - a solution to a problem often has several steps, even repetitions and requires strategy and formulation of a set of rules
- **Efficiency** - resources are previous, often this refers to the time required to solve a problem
- **Innovation** - observing the world and noticing where things might be improved can lead to major advancements and push boundaries

Computational thinking is often brought up in the context of learning computer programming, a notoriously difficult task [7] and whilst many struggle others seem to excel.

978-1-4799-6795-7/14 $31.00 © 2014 IEEE
DOI 10.1109/iTAG.2014.15

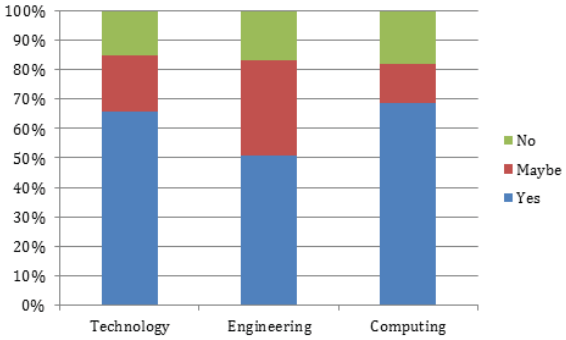

Fig. 1. Student interest in subjects after attending the Printcraft bootcamp [11].

Yet there are no studies suggesting individuals may have a natural aptitude for programming. Programming languages are a formal combination of structures and commands that allow algorithms (any step-by-step set of instructions) to be implemented. Without the ability to effectively dissect a problem and design a solution it can be difficult to translate intentions into code. One argument for the difficulty that many computer science undergraduates face when they start programming is that problem solving with computational thinking is not taught as a skill in the same way arithmetic or writing is.

When building a computer program it is not a simple task of writing code until it is done. Many problems must be broken down into individual tasks each with their own solutions. Across the world, thousands of developers work seemingly in isolation, yet collaboratively on projects [1] and are able to do this thanks to the separation of concerns and abstraction resulting from a computational approach to problem solving.

But it is not just programming challenges that benefit from computational thinking. Our reason for focusing on the skill of computational thinking in this paper is not only its significance to the field of computing but also its application to literally any field [4], [8].

II. Engagements

Technocamps Aberystwyth performs a variety of style of engagements such as bootcamps (half-term, multi-day courses), workshops (in-school, 3 hours) and an after-school club. Here we look an example of each and the resources utilised to enthuse students with a passion for technology whilst teaching transferable skills.

A. Printcraft Bootcamp

Minecraft [2] is a hugely popular, multiplayer sandbox game environment where participants can build with others online. It attracts a massive number of users of all ages and has proven efficacy as an educational tool [9], [10]. Most often, Minecraft is compared to LEGO™ for the style of play.

3D printing is a process, also known as Fused Deposition Modelling, in which 3D structures are made through the extrusion of a molten material at precise locations. As the extruding tool moves and the material cools and hardens, a solid structure is formed. The path of a 3D printers extruder is controlled with a series of low-level machine instructions. The generation of the instructions can be a complex problem as most efficient paths for travel and details of interior support structure are calculated and so they are usually generated automatically from computer-aided design (CAD) software.

We used Printcraft [3] as a means to combine Minecraft with real-world manufacturing for teaching fundamental computational, science, technology and engineering concepts without the need to become familiar with complex CAD software. For example, when working with a 3D printer a number of aspects must be considered. As in Minecraft, it is not possible to place a block in mid-air, necessary support structures are required. Similarly, bridge gaps will often collapse and necessitate a series of steps to form an adequate arch (as seen in real world architecture). Drawing similarities from the virtual world of Minecraft and the physical world enables students to experiment with engineering concepts and design to develop an appreciation of issues going from concepts to implementation.

Students are organised into teams, each tasked with the goal of a specific building or group of buildings (such as a school, or housing). Teams are allowed to self-organise but are encouraged to collaborate with one another.

But it is the concepts of abstract design, combined with numerical control that we emphasise for their similarities with computational thinking.

Fig. 2. A town, designed in Minecraft then built with 3D printers. Students are taken on a journey through design and manufacturing over the course of a 2-day bootcamp.

Particpants of the Printcraft bootcamp were all aged between 11-19 and were challenged to work in teams to design (in Minecraft) and build (with 3D printers) a model city. The bootcamp was assessed in the form of a questionnaire after the event that asked a number of technical questions as well as interest in various subjects (Figure 1).

Printcraft is a good "gateway" tool offering an approach to increasing understanding in computational thinking, computer-aided design and manufacturing in a way that is engaging and stimulating for pupils. The learning outcomes of the bootcamp include understanding of:

[1] Many examples are open-sourced on sites such as http://github.com/
[2] http://minecraft.net/
[3] http://www.printcraft.org/

- 3D printing as an engineering tool
- Software and hardware processes behind 3D printing (numerical control)
- Role of computer-aided design (CAD) in engineering
- Inter- and Intra- team collaboration
- Project planning and team role development

B. AppInventor Workshop

Mobile devices such as smart phones or tablets are quickly replacing desktop computers for basic tasks such as email and web browsing. Many social media services are dedicated to mobile use and recent trends suggest mobile devices will overtake desktop computers for browsing in the next couple of years. As the availability of mobile computing increases there are new opportunities for development.

AppInventor [4] is a graphical android programming environment. Graphical programming replaces conventional text based control structures of programming languages with a drag-and-drop interface allowing users to connect loops, conditionals, etc in a visual environment. Graphical programming environments can be an effective tool for introducing young people to programming and the associated computational thinking without having to worry about syntax or spelling [12].

For many students, their existing familiarity with such devices gives them some expectations as to how apps work and what capabilities to expect. Developing their own app reshapes the way they view the technology.

Over the course of 6 hours, students attending the workshop first created a rudimentary app by following the example of the tutor. The goal of this task was to give students a set of tools and processes that could be applied to use any component of their choice. Students were specifically shown integer/string variables, if..else statements and for loops, as well as some basic components (buttons, accelerometer, images). After this brief introduction, students were encouraged to design then create an app related to their studies. Examples included a "'babygotchi" childcare game, bird call library and historical quote generators. Many students used their own initiative to find audio/visual material on the Internet to use within their applications.

Figure 3 shows the various components of the Android API (application programming interface) the 28 apps developed by students used. We use the API interactions as a rough indicator of participant interest and ability. The students from these sessions had no prior programming experience and were aged 17-19. Most enthusiasm came from applications utilising sound in response to actions as well as dynamic events.

From the workshop, students gained an appreciation of the complexity of apps often taken for granted and an understanding of how software interacts with the hardware of a device. The advantage of graphical programming is that logical and algorithmic skills essential to computational thinking can be taught in a way that does not intimidate students with unnecessary complexity. The learning outcomes of the workshop include:

[4] http://appinventor.mit.edu/explore/

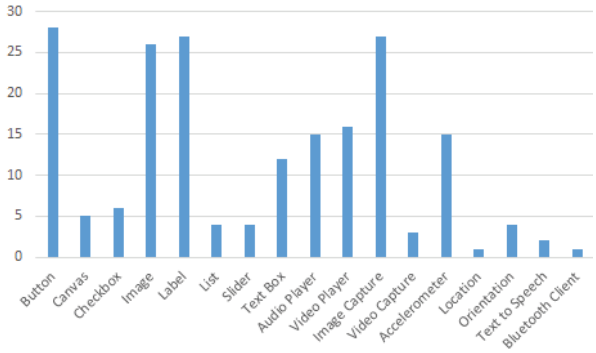

Fig. 3. 28 student pairs designed and created their own mobile apps after minimal tutoring. The graph shows how many used various components provided by AppInventor.

- Understand the purpose of control structures in software code
- Appreciate the hardware devices of typical mobile computing devices (accelerometer/orientation sensors, touch screen, audio, video, etc)
- Understand a generalised process of software engineering with deployment and development through an integrated environment

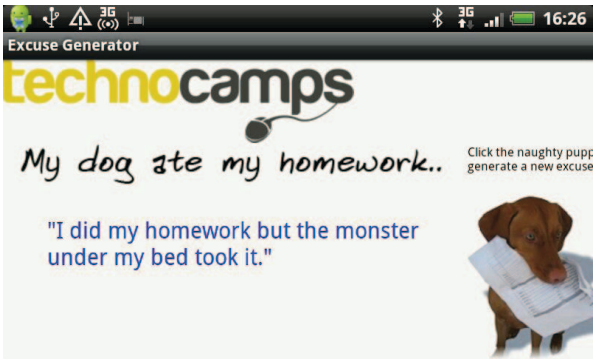

Fig. 4. Example application: "Excuse generator' that utilises arrays of strings, the device touch screen and a random number generator.

C. Robotics Club

Our most diverse engagement is via an after-school robotics club which is an ongoing project with weekly engagements with a general goal to building robotics projects of the participants' own designs. A much broader understanding and range of skills are required to cover both the hardware and software aspects and access to a variety of resources including Arduino microcontrollers and Lego Mindstorms is provided. Figure 5 shows one project - a humanoid robot made from 3D printed parts.

The learning outcomes of the club include:

- Understand basic electronic circuits and specific components (such as light emitting diodes, tranistors and potentiometers)

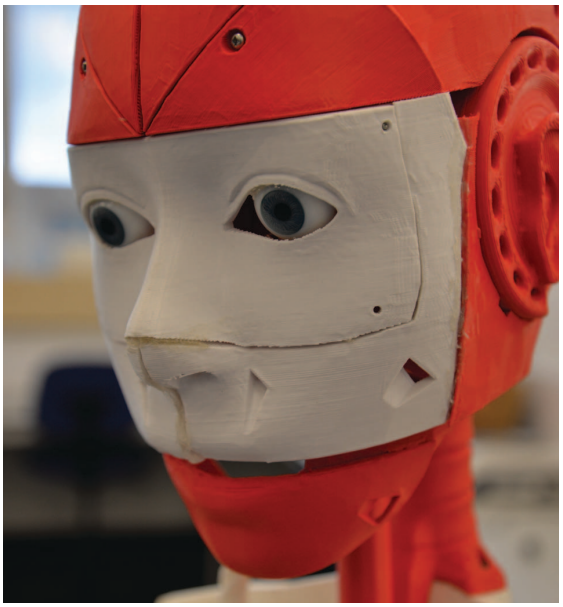

Fig. 5. 3D printed head of an Inmoov - an open-source project for a humanoid robot, selected and built by students at the club.

- Work with popular environments such as S4A (Scratch for Arduino) and NXT-G for microcontroller programming
- Work with CAD software for printed circuit board design

There are 22 long-term particpants aged 11-19; a questionnaire was devised to assess the learning and progress of particpiants that tested general and specific knowledge of technical areas. One item of interest highlighted by the questionnaire was the concept of variables in programming, with 43% unclear on where they should be used.

The robotics club offers opportunities to work with both software and hardware which helps reduce the esoteric nature of some computing aspects and allows participants to work where they have the most interest, without being forced into overly unfamiliar territory.

III. Conclusions

Everyone should have the motivation and means to explore their world. Computing is now ubiquitous, whether you realise it or not - when you wear a digital watch, carry a mobile phone or use an electronic door or use a cash machine - you're using computers. Computing furthers industry and science and improves education. It is only sensible that young people are brought up with intimate knowledge of computing.

The term "computational thinking" is a common one that communicates logical, algorithmic processes that can empower students greater reasoning ability and problem solving. We have presented three diverse engagements that seek to communicate this skill, as well as technical expertise in a way that is reproducible and engaging for participants.

The learning outcomes we have presented teach theoretical concepts to fuel an appreciation of technology in the everyday world as well as highly technical skills that contribute directly to tasks such as programming. Regrettably, we have limited empirical evidence for determining how much students gain from attending these courses. In the future it would be desirable to formulate a comparable and robust framework for assessing knowledge gains and reasoning skills before and after engagement.

To those seeking to teach computational thinking or other technological skills we hope the engagements discussed here are of interest. Activities with which participants have existing interest and experience are easier for many to relate to and the proliferation of technology means that almost any student has a subject or platform of interest to them. There are a number of well-designed educational tools such as AppInventor and Scratch but other options such as versatile gaming environments such as Minecraft should not be dismissed as merely a game and have strong suitability for teaching formal skills in a less direct manner.

The maker community has led to an uprise in affordable computing technology for a diverse range of projects that can suit any need; allowing students to get hands on and interact with technology in ways not feasible before. Educators should do their best to immerse themselves in the opportunities available for new ways of learning both about computers and with computers.

Acknowledgments

This work was part of the Technocamps project, funded by the Welsh European Funding Office with additional support from the Infinity interactive science and technology exhibition.

References

[1] The Royal Society, "Shut down or restart?: The way forward for computing in UK schools," 2012.

[2] Y. Eshet, "Digital literacy: A conceptual framework for survival skills in the digital era," *Journal of Educational Multimedia and Hypermedia*, vol. 13, no. 1, 2004.

[3] C. Anderson, *Makers: The New Industrial Revolution*. Crown Publishing, 2012.

[4] S. Hambrusch, C. Hoffmann, J. T. Korb, M. Haugan, and A. L. Hosking, "A multidisciplinary approach towards computational thinking for science majors," ser. SIGCSE '09, 2009.

[5] P. J. Denning, "The profession of it: Beyond computational thinking," *Commun. ACM*, 2009.

[6] J. M. Wing, "Computational thinking," *Commun. ACM*, 2006.

[7] T. Jenkins, "On the Difficulty of Learning to Program," in *3rd annual Conference of LTSN-ICS,*, 2002.

[8] R. Landau, G. Mulder, R. Holmes, S. Borinskaya, N. Kang, and C. Bordeianu, "INSTANCES: incorporating computational scientific thinking advances into education and science courses," *Concurrency and Computation: Practice and Experience*.

[9] D. Short, "Teaching scientific concepts using a virtual world–minecraft," *Teaching Science*, 2012.

[10] S. C. Duncan, "Minecraft, beyond construction and survival," *Well Played*, vol. 1, 2011.

[11] S. Jones, E. Posey, J. F. Roscoe, and P. Harter, "Creative computing with minecraft," *A Child's World Conference*, 2014.

[12] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, 2009.