Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

A Case Study Illustrating Coding for Computational Thinking Development

Siu Cheung KONG[a*] & Ping LI[b*]
[a] *Centre for Learning, Teaching and Technology, The Education University of Hong Kong*
[b] *Department of Mathematics and Information Technology, The Education University of Hong Kong*
*sckong@eduhk.hk; *pli@eduhk.hk

Abstract: Previous studies suggested that computational thinking is an important skill that schools should start equipping children with from primary school education. This study proposes way to develop a mobile app coding curriculum in primary 4 to 6 to nurture students' computational thinking. It will guide students to undergo learning stages from developing their own codes and combining with others' work to create their own apps and staging their apps to gain confidence and recognition of performance in coding. An example of mathematic game in the curriculum is selected as a case study to illustrate in this study how it incorporates the elements of computational thinking in the coding activities progressively. It is believed that the proposed way of curriculum development paves a learning path which may drive students' interest in coding and students could develop computational thinking skills progressively.

Keywords: computational thinking, coding education, coding curriculum, mobile app coding, primary school education

1. Introduction

The term "computational thinking" had gained considerable global awareness in the past decade. Societies are now in need of people with ability to think like a computer scientist. The importance of computational thinking has been reinforced in primary and secondary schools in recent years. Many countries implemented coding courses to develop students' computational thinking ability. In Hong Kong, computational thinking was also highlighted in the "Report on the Fourth Strategy on Information Technology in Education" in 2015 (Education Bureau, 2015). This paper proposes a computational thinking curriculum in senior primary school to teach students mobile apps coding. It illustrates how coding education develops students' computational thinking by various coding tasks.

2. Background of Study

Since the introduction by Jeannette Wing, computation thinking (Wing, 2006) had gained increasingly heated discussion of its operational definition in multiple disciplines. According to Wing (2006, p. 33), computational thinking is taking an approach to solve problems, design systems and understand human behavior that draws on concepts fundamental to computing. Computational thinking is to think like a computer scientist. It is a set of thinking skills, habits, and approaches integral to solve complex problems (Wing, 2006; Lee, Martin, Denner, Coulter, et al., 2011). Abstraction in computational thinking marks its differences from algorithmic and mathematical thinking (Wing, 2011). By definition, abstraction is defining patterns, generalizing from specific instances, and also a key to attack complexity. Apart from abstraction, computational thinking also requires automation and analysis which instructed computers to carry out repetitive tasks efficiently and to validate whether the abstractions made are correct. In recent years, computational thinking gains educators' attention. It sheds light on the potential practical applications among the youth. Lee et al. (2011) illustrated the application of computational thinking among students and found that it has far-reaching influences on upgrading students from consuming others' creation to creating their own innovative products. Wing's call for computational thinking draws increased research investigations that attempt questions such as

how to apply computational thinking in a class room setting of a primary school? What kind of skills should the students demonstrate? What kind of skills should the teachers possess in order to put computational thinking into practice (Barr & Stephenson, 2011)? This study attempts to answer the aforementioned questions by proposing a way to develop a curriculum of computer coding.

3. Curriculum Design of Coding Education in Senior Primary

This study proposes a curriculum of coding education that forms a consecutive learning pathway across primary grade 4 to 6. The design principle of curriculum aims to drive students' interest in coding through repeated immersion in coding processes. It is hoped that students will eventually self-direct their learning and turn coding into their own habits (Looi, Chan, Wu, & Chang, 2015). In view of this, this course will guide students to undergo learning stages from imitating others' work and combining with their own ideas, to coding their own apps and staging the app for performance. At the end, students will be able to build apps like safe zone detector, name card exchanger, mathematic games etc. This paper selects the mathematic game to illustrate how students can eventually be able to create a two-player fraction game connected via Bluetooth. Starting form Primary 4, students will begin by learning how to code a single-player addition game as a foundation. In this game, the interface only includes a blank addition equation for number input and three functional buttons (see Figure 1a). In primary 5, students will attempt to build a single-player fraction game (see Figure 1b) in which more considerations are required in the coding process to ensure that the checking of fractions equivalence can work properly.

Figure 1a & 1b. Interfaces of single-player addition game (left) and single-player fraction game (right).

In primary 6, students have to build a two-player fraction game with devices connected via Bluetooth. To play this game, one player will set and send out a fraction. On receiving the fraction, the other player inputs a fraction and check whether they are equivalent (see Figure 2). There are two key tasks in the coding process. The first task is building connection, students need to identify how to connect two devices in coding for data transmission. The second task is answer checking, students explore ways to check the equivalence of two fractions through coding. Throughout the course, teachers will guide students to start their coding process by imitating others' work, then gradually develop and combine their own codes with others' to come up with a solution. It is believed that students can apply the knowledge gained to develop new apps in other context at the end.

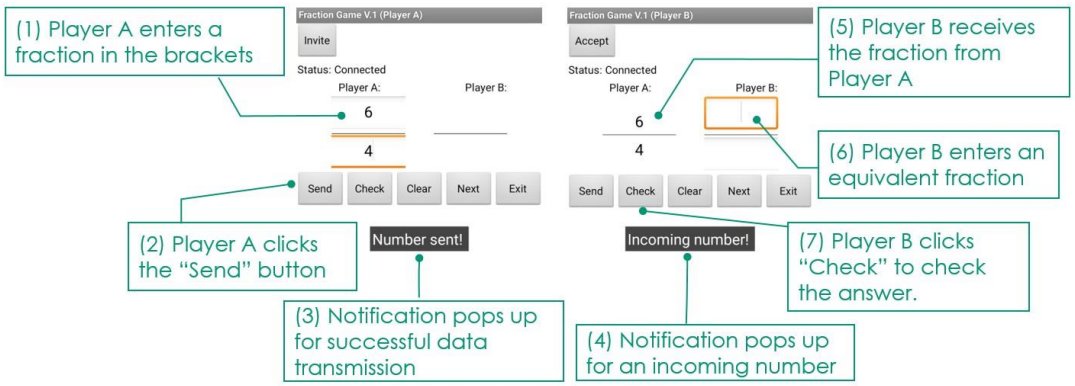

Figure 2. The interface design of the two-player fraction game.

4. Computational Thinking in Coding Education

Grover and Pea (2013) pointed out that there are 9 key elements that are widely accepted as important composition of computational thinking and they form the basis elements of curricula design that aim to support its learning and development. They are "Abstraction and pattern generalization", "Systematic processing of information", "Conditional logic", "Symbol systems and representation", "Algorithmic notions of flow of control", "Structured problem decomposition (modularizing)", "Iterative, recursive and parallel thinking", "Efficiency and performance constraints" and "Debugging and systematic error detection". The following discussion will describe how the coding examples discussed above may incorporate most of these elements in this case study of the mathematic game.

4.1 Computational Thinking in Coding a Single-player Addition Game

In the first year of the curriculum, it is important to get students a first-time hands-on experience of mobile app coding, so the task will be a simple one based on fundamental coding concepts to drive their initial interest. The game interface consists of a blank equation and three functional buttons (see Figure 1a). Students are taught to build the interface with Block Editor by inserting 3 text boxes, "+" and "=" operators and three buttons below. For each item, students need to give a name, set the sizes, properties and locations. In this task, students will develop a concept that each item has a purpose, and each must be named and defined with properties in order to ensure the app can carry out its purpose properly. For instance, "Check" button must be categorized as a button and "+" as a label. If a student wrongly set "check" button as a label, the word "check" would appear as plain text that cannot be clicked. After the interface design, students will add functions to the 3 buttons. The use of symbol here develops students' concept of **"symbol systems and representations"** and the combination of all these items in the interface provides a thinking of **"systematic processing of information"**. Students will also explore the "if…then…else…" conditional statement in the check button. When the addition of two numbers on the left equals to the value of the right number, an "Equal!" notification shows up, or else "Unequal!" appears. Students will learn **"conditional logic"** in this task under teachers' guidance.

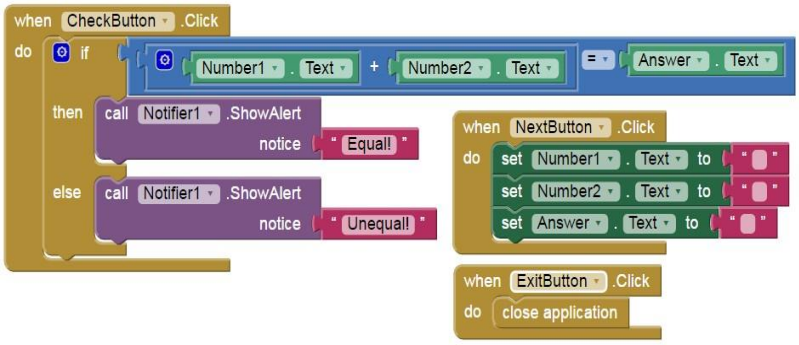

Figure 3. The codes of a single-player addition game in Block Editor.

4.2 Computational Thinking in Coding a Single-player Fraction Game

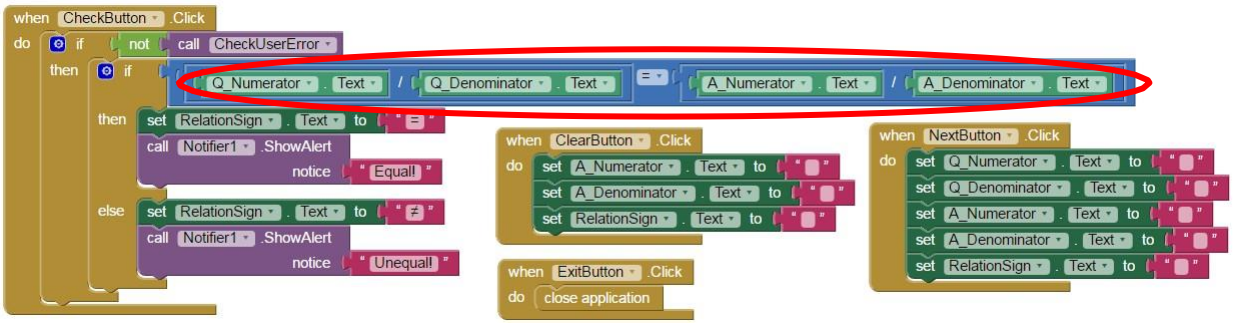

Figure 4. One of the ways to check if the two fractions are equal.

When students reach primary 5, they are required to modify the addition game into a fraction game. Most of the buttons in this coding task are similar, except that a "Clear" button will be added to clear the input of answers and the change of addition equation to the checking of equivalence of two fractions. This needs a more complicated interface design and allows more than one ways of answer checking. To check whether the two fractions are equal, there are two solutions: (1) to check whether the values of divisions on each side are equal; (2) to check if the multiplication of the left numerator and the right denominator equals to that of the right numerator and the left numerator (see Figure 4). Teachers will guide students to decide which would be a better solution. After discussion, students will come to two conclusions. First, when the first solution is adopted, there is a potential error that computer might wrongly judge two equal fractions as unequal due to their different values after rounding off. Second, if users entered non-integral values in input boxes or zero in any denominators, errors appeared. Removal of these bugs requires students' **"debugging and systematic error detection"** skills. Considering that the debugging codes are beyond students' level, teachers will provide a solution of coding that disables inputs of non-integral values and ask students to input non-zero value in denominators. Extended learning materials will be provided to develop students' debugging skills.

4.3 Computational Thinking in Coding Two-player Fraction Game Connected via Bluetooth

In primary 6, students will further develop the app into a two-player fraction game that enables data transmission in two devices via Bluetooth. In order to support data transmission, two devices must undergo Bluetooth connection and client server connection beforehand (see Figure 5). This requires students' understanding of the operation concept of the three-layer connection and that of the client-server relation. To make a Bluetooth connection within the app, the two devices have to build a client-server relation in which the server makes a connection invitation and the client accepts. As the client and server are taking different actions in this task, **"iterative and parallel thinking skills"** are needed here due to the two different versions of coding required for each of them respectively (see Figure 6).

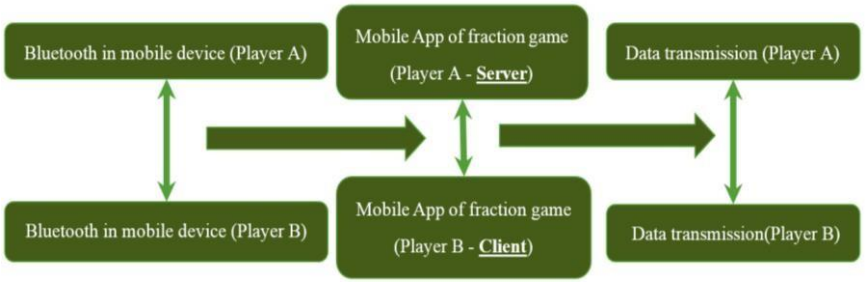

Figure 5. Three layers of device connection in support of data transmission in fraction game app.

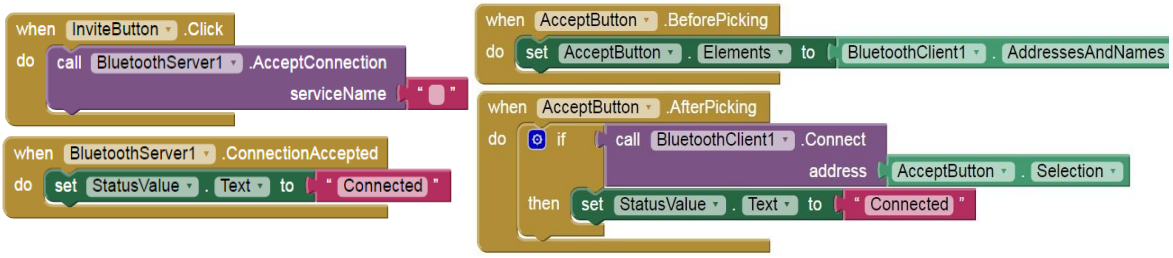

Figure 6. Codes for Bluetooth connection within the app for server side (left) and client side (right).

In the coding for data transmission part, students need to think in a parallel way that the app should contain both the sending fraction and receiving fraction components so that both devices can detect when there is a new fraction sent from the partner device and thus act in response by showing "Incoming number!" and the fraction received on screen (see Figure 7).

Apart from the above task of building Bluetooth connection and transmitting data between two devices, students also need to add one more functional button "Send" to send the fractions to the partner. From the whole coding task, we can see a much more advanced structure at primary 6 level. First, as

students have to decompose a complicated task into sub-parts to tackle problems systematically. The sending and receiving components require the utilization of modularizing skills which we call **"structured problem decomposition (modularizing)"**. Second, students have to consider the flow of actions carried out by the computer by considering "What pre-actions have to be done in order to enable successful data transmission between two devices?" and "Which side should take actions first in making connection, server side or client side?", these thinking processes are what we call **"algorithmic notions of flow of control"**. After completion of the task, teachers will guide students to think of possibilities of broader usage of acquired skills to other coding tasks and the further development of the game into other applications. It is hoped that at the end of the whole module, students' interest can be driven and they can be encouraged to turn coding into their learning habit to further develop their coding skills.

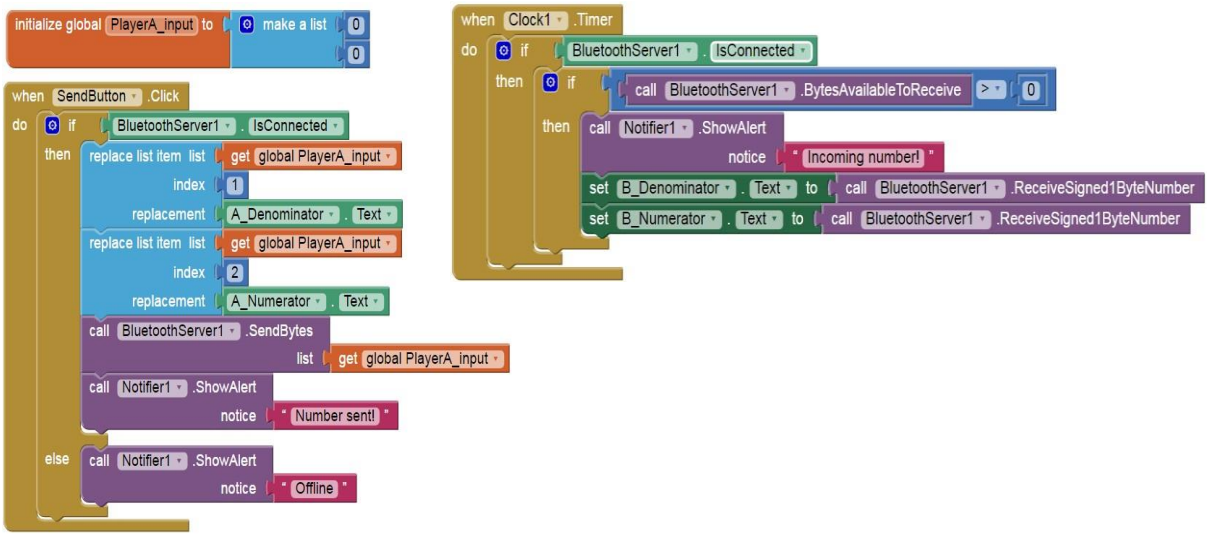

Figure 7. Codes for the sending fraction (left) and receiving fraction (right) for the server side.

5. Summary and Future Work

This case study illustrates how the example of mathematics game coding can nurture students' computational thinking in senior primary. As the game is only one of the activities of the curriculum, it might not cover all 9 elements of computational thinking in Grover and Pea's delineation of computational thinking. To develop a curriculum for senior primary, further work will be attempted to design more practical examples for coding education. Despite the above limitation, the illustration of mathematics game suggests that by reusing and remixing existing features and functions in the app, this example can provide a sustainable learning progression throughout the 3-year curriculum in support of computational thinking development of students in coding education. It is hoped that this example can serve as a reference for educators in their curriculum planning in primary school, and they can take this case as a reference to develop other examples that can be included in their curriculum.

References

Barr, V. and Stephenson C. (2011). Bringing Computational Thinking to K-12: What is involved and what is the role of the computer science education community? *ACM transactions on computational logic*, *2*(1), 111-122.

Education Bureau. (2015). *Realising IT potential • Unleashing learning power: A holistic approach.* Hong Kong: Education Bureau.

Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson. J., Malyn-Smith. J., and Werner. L. (2011). Computational Thinking for Youth in Practice. *ACM inroads*, *2*(1), 32-37.

Looi, C. K., Chan, T.-W., Wu, L., and Chang, B. (2015). The IDC theory: research agenda and challenges. *Workshop Proceedings of the 23rd International Conference on Computers in Education*, 796-803.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-36.

Wing, J. (2011). Research notebook: Computational thinking—What and why? The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh. Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why