Learning Programming Concepts Using Flowcharting Software

Kwan Chi Kuen
Computer Department, King Ling College, Hong Kong

ABSTRACT

Traditionally, when students learn programming concepts by sticking on a particular programming language, they have to spend time struggling against the syntax of that programming language, which distracts themselves from exploring the algorithm. If only flowchart is taught without programming, students cannot execute their flowchart, making them unable to justify their algorithm. RAPTOR is a flowcharting software which solve the above dilemma—by writing flowcharts which can be executed, student can learn programming visually and easily, without suffering from the tedious syntax.

Keywords: Flowcharts, Programming Concept, RAPTOR

I. INTRODUCTION

According to the Education Bureau in HKSAR, one of the main focuses on Technology Education (TE) in Hong Kong secondary school curriculum is to learn "how human beings solve their daily problems and how the processes involved can be replicated and transferred to solve new problems."[1]. Being one of the subjects in the Technology Education Key Learning Area (TEKLA), Information and Communication Technology (ICT) requires students having intensive skills in solving daily life problems. To do so, ICT students have to learn basic programming concepts, which provide students a basic understanding of the steps and strategies involved in solving a problem systematically.

Traditionally, two approaches are used to learn programming:

Approach 1

Learn a programming language (such as Pascal or C), then write programs using that programming language.

Approach 2

Learn the algorithm, i.e. the steps to solve a problem, and represent the algorithm using appropriate tool(s), such as flowchart or pesudocode.

The disadvantage of Approach 1 is that learning programming using programming language sounds too difficult for less able students, as they have to memorize all the syntax of that particular programming language, and they are strongly discouraged when they fail to get their programs compiled due to some minor but tedious syntax errors (such as having ";" missed, using "=" instead of ":=") – even if their algorithms are correct. Note that the main aim in programming is to solve problems, the key item students have to learn is the algorithm, not the syntax. The complicated syntax of a programming language forces students to fix syntax errors, instead of the logical error of the algorithm. The syntax also discourage them from learning programming, as they have an impression that learning program is difficult, given that their program can never get compiled.

To solve the problem, starting from 2003, the compulsory part of the Computer & Information Technology (CIT) Curriculum (now replaced by ICT) in senior secondary did not require students learning programming languages any more. Such part is left to be taught in an elective module, where more able students can select this module and explore the fun of writing program using programming language(s). Instead, in the compulsory part, CIT/ICT students are required to write the algorithm using flowchart and pseudocode, i.e. Approach 2 is adopted. Without the barrier of syntax, students can focus on the algorithm itself, and the essence of problem solving using the correct algorithm is emphasized.

However, there is a major drawback on such approach. As students are required to construct flowcharts which can never be implemented, students can never know nor prove whether their algorithms are correct. According the my experience, once I asked 20 CIT students to write a flowchart to find out the maximum number among 3 numbers, all of them managed to write a flowchart, yet these 20 flowcharts were all different from teacher's "model answer". They all knew that teacher's "model answer" was correct, yet they were not sure whether their own answers were correct or not. They could not test their flowchart to justify that the algorithms were correct as that flowcharts, written on pieces of paper, were not executable.

II. FLOWCHARTING SOFTWARE

Using flowcharting software can solve the drawbacks in both Approach 1 and Approach 2 mentioned above. Flowcharting software, such as RAPTOR, provides a user

interface for users to create executable flowcharts. Instead of writing program codes which might cause syntax errors, in RAPTOR, the main algorithm is represented by flowchart(s). All students need to do is to drag flowchart symbols to the editing area, and arrows will automatically links symbols together. In such case, students can focus on the correctness of the algorithm (i.e. minimizing logical error), instead of the correctness of the syntax (i.e. minimizing syntax error). In RAPTOR, syntax is minimized. Besides, as the flowchart is executable, students can test their flowcharts, making them being able to justify the correctness of their algorithms.

III. DESCRIPTION OF RAPTOR

RAPTOR stands for Rapid Algorithmic Programming Tool for Ordered Reasoning, which is a free flowcharting software package running in the .NET Framework. When a new RAPTOR file is opened, a blank flowchart is formed with "Begin" and "End" symbols includes. User can drag either one of the six flowchart symbols / structure to the editing area each time:

- Assignment Symbol: For assigning variables
- Call: For calling sub-flowcharts
- Input Symbol
- Output Symbol
- Selection Structure
- Loop Structure (Iteration)

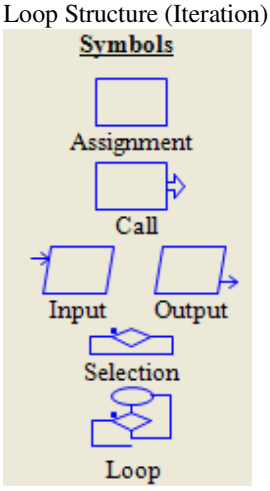

Figure 1. *Flowchart symbols/structures in RAPTOR*

Such arrangement makes sure that the flowchart is well structured, making loops being properly nested, and arrows being properly connected. To minimize syntax error, RAPTOR allows flexible syntax, e.g. the Boolean "AND" operation can be represented by either "&&" or "AND". Besides, RAPTOR supports around 40 build-in functions which provide supports like file operation, drawing graphs, generating random number etc.

To write flowcharts, students need to drag the flowchart symbols/control structures to the editing area, between the "Begin" symbol and the "End" symbol. To execute the flowchart, one can press the "Play" button. Students can also stop or pause the execution by pressing the "Stop" and "Pause" button. There's a "Step to next Shape" button which allows students tracing the flowchart symbol by symbol, in a "step by step" manner. The execution result can either be showed on the console, or on the graphical user interface (GUI) written by the student. During execution, the status of the variables is also shown.

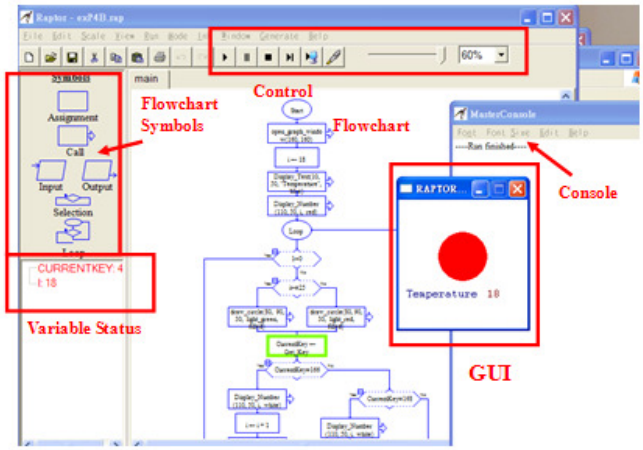

Figure 2. *RAPTOR interface*

IV. ADVENTAGE IN USING RAPTORS IN TEACHING PROGRAMMING CONCEPTS

Besides the advantages mentioned above, below are other advantages which makes RATPOR a desirable tool in teaching programming concepts:

1. The flowcharts in RAPTOR allow students learning programming and tracing algorithm visually.
2. RAPTOR allows students carrying out dry run with adjustable speed, meaning that students can trace the algorithm according to their progress. Learner diversity is catered.
3. Teachers can adjust the speed of the execution of the flowchart during dry run, so that he can expand certain points in details.
4. During execution, RAPTOR displays the value(s) stored in various variables. This allows student understand the changes of the values of variables during different stages of the algorithm.
5. RAPTOR files can be exported to "pseudocode" with the following syntax: Ada, C#, C, Java.
6. RAPTOR supports sub-flowcharts, drawing graphs, sound, array, file, event (mouse, keyboard), sound, vector graphics etc.
7. Students can find most references in "Help", which facilitate self-learning.

V. LIMITATIONS IN USING RAPTOR IN TEACHING PROGRAMMING CONCEPTS

Although RAPTOR has huge advantages, there are two major limitations:

1. RAPTOR provides English interface only. In CMI (Chinese-as-the-medium-of-instruction) schools, it's a major drawback.
2. The flowchart symbols used in RAPTOR is a bit different from the commonly used flowchart symbols. E.g. in RAPTOR, the Input symbol consists of a parallelogram with an arrow, while a commonly used Input symbol consists of a parallelogram only.

VI. RAPTOR AND THE HONG KONG ICT CURRICULUM

In Hong Kong, ICT students can learn programming concepts in two different levels. All ICT students have to learn basic programming concepts, as stated in the compulsory part of the ICT curriculum[2]. This compulsory module does not require students learning particular programming language. For students who are more interested in programming, they can choose the "System Development" module in the elective part of the curriculum.

The Basic Programming Concepts module in the compulsory part of the ICT curriculum is further divided into three parts, namely "Problem-Solving Procedures", "Algorithm Design" and "Algorithm Testing". Using RAPTOR, the following topics stated in the curriculum can be taught:

- Problem-Solving Procedures
  - Solve a problem by breaking it down into sub-problems or modules
- Algorithm Design
  - Design an appropriate user interface
  - Data types and data structures (integer, real, character, Boolean, string, 1D array)
  - Control structure (Sequence, selection, iteration)
  - Modularity
- Algorithm Testing
  - Trace & test algorithm

RAPTOR can in fact cover most of the topics stated in the compulsory module of the curriculum.

VII. DEVEOPMENT OF BASIC PROGRAMMING CONCEPTS TEACHING PACK BASED ON RAPTOR FOR SECONDARY SCHOOLS IN HONG KONG

Due to the usefulness of RAPTOR in learning programming concepts, various activities had been done to introduce RAPTOR to ICT teachers in Hong Kong. A Teaching Pack was also developed for teachers to teach ICT programming concepts using RAPTOR. The details are as follows:

A. *Phase 1. Introducing the software*

Being a seconded teacher in the Curriculum Development Institute in the Education Bureau, in May 2008, I introduced RAPTOR to around 300 Hong Kong Secondary Schools ICT teachers in a seminar organized by the Education Bureau. After the seminar, some teachers asked whether tailor-made teaching pack can be developed which help teachers teaching senior form ICT using RAPTOR.

B. *Phase 2. Developing Tailor Made Teaching Pack*

Based on the requests of the teachers, I've spend months developing a teaching pack which help teachers teaching the whole ICT Basic Programming Concepts module (in the compulsory part) using RAPTOR and some other software packages. The full name of the teaching pack is called "NSS Information and Communication Technology: Basic Programming Concepts Teaching Pack" (Teaching Pack). It allows teacher using RAPTOR to teach most of the programming concepts stated in the ICT curriculum. The Teaching Pack contains 12 chapters, with around 30 RAPTOR tasks, and 6 major projects, which cover areas like control structures, data type, array, modular approaches, testing etc. It takes around 20 hours to go through the whole Teaching Pack.

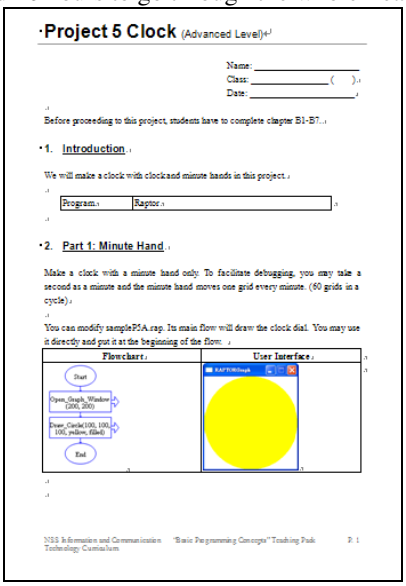

Figure 3. Snapshot of the NSS Information and Communication Technology :Basic Programming Concepts Teaching Pack

The main design principles of the Teaching Pack are as follows (Kwan, 2008)[3]:

1. The Teaching Pack is mainly based on the NSS Information and Communication Technology (ICT) curriculum. It is also suitable for the archived Computer and Information Technology (CIT) curriculum.
2. The Teaching Pack encourages learning through activities (i.e. Task-based learning).
3. All the algorithms in the Teaching Pack are illustrated

with flowcharts and pseudocode so that students can focus on the algorithms in problem-solving instead of the syntax of programming languages.

4. Activities in the Teaching Pack require students to use specific software to execute the flowcharts. Through the executions of the flowcharts, students can know whether the algorithms are correct or not and learn how to perform algorithm testing, debugging and dry running.
5. The activities use real-life examples as exercises.
6. Freeware is used for teaching and learning which is easy to promote and it allows students to install the software at home for self-study or revision purpose.
7. Some of the activities and projects are divided into different levels so as to cater learners' diversity.
8. The Teaching Pack does not aim to replace textbooks. It is activity oriented by providing a number of executable flowcharts for practice while textbooks stress more on the theories behind the activities. The Teaching Pack and textbooks can be used together to complement each other.

C. *Phase 3. Piloting*

The Teaching Pack was first written in Chinese. It was piloted in King Ling College from Sept 2008 to Oct 2008. In the pilot scheme, 40 students were divided into two groups. Group A used RAPTOR and the Teaching Pack to learn programming concepts, while Group B, being the control group, learnt programming by purely constructing flowcharts on paper. One quiz and one examination were conducted afterward, and the results are as follows:

TABLE I. ASSESSMENT RESULTS

	Group A (Using RAPTOR)	***Group B (Control)***
Mean mark of the quiz	46	36
Mean mark of the examination	62	49

D. *Phase 4. Formal Release (Chinese Version)*

The Chinese version of the Teaching Pack was launched in a seminar in Jun 2009. In the seminar, the Teaching Pack was introduced. Around 300 ICT teachers attended the seminar and teachers can freely download the Teaching Pack in the CA online platform provided by the Education Bureau.

E. *Phase 5. Formal Release (English Version)*

As some teachers requested for the English version of the Teaching Pack, the Chinese version was then translated into English version with the help of the Curriculum Development Institute in the Education Bureau. Another seminar was launched in July 2010 and the English version was launched at the same time.

VIII. FUTURE WORK

After the formal launch of the RAPTOR Teaching Pack, recently my ICT students are working developing simple computer games using RAPTOR. A project was conducted in King Ling College, which requires ICT students creating a computer game using RAPTOR. The result was promising, as students managed to use simple flowchart components to create complicated and funny games. For example, one group had created a memory game using RAPTOP. A GUI was developed, where users can choose the level of difficulty.

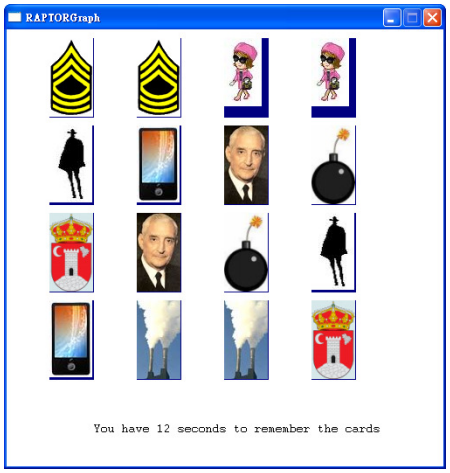

Figure 4. Memory game developed by ICT students using RAPTOR

IX. CONCLUSIONS

Using RAPTOR, ICT students can learn programming concepts easily by drawing executable flowcharts. Tedious syntax problems can be avoided, which helps students focus on the algorithm instead.

X. REFERENCES

[1] The Curriculum Development Council and the Hong Kong Examinatinos and Assessment Authority. "Information & Communication Technology Curriculum and Assessment Guide (Secondary 4-6)", 2007, pp. 1-2.

[2] The Curriculum Development Council and the Hong Kong Examinatinos and Assessment Authority. "Information & Communication Technology Curriculum and Assessment Guide (Secondary 4-6)" , 2007, pp. 7.

[3] C.K. Kwan. , "NSS Information and Communication Technology: Basic Programming Concepts Teaching Pack" ,2008.