Scratch: A Tool for Reappropriation in Science Class in Secondary School

Olivier Goletti[1,2]
1. UCLouvain, Belgium
2. SI^2 coordinator
name.lastname@uclouvain.be

Introduction

In the French-speaking part of Belgium, there is no compulsory computer science (CS) course in the curriculum. However, there are four periods per week in lower secondary school (grades 7-8) for complementary activities. Certain schools take advantage of this to propose a computer science course, but most of the time, it is mainly devoted to digital litteracy in the sense of office suite usage.
SI^2 is a working group made of representatives from the higher education institutions (Universities and Colleges) in computer sciences. SI^2 aims eventually to introduce the teaching of computer science in lower secondary education. In particular, we are developing activities and pedagogical material for teachers and lower secondary students.
This poster presents a sequence of lessons that were given to secondary school students (13-14 years old) as part of a partnership with SI^2 and a local school. In the first semester, students engaged in 10 hours of introductory activities (unplugged and online). These activities introduced ideas in computer science (such as algorithms) and basic challenges using block programming. During the second semester, students were asked to create their own project in Scratch to illustrate concepts they learned in the science class.

Methodology

First semester: Discovery of Computer Science

Here is an overview of the activites given during the first semester:

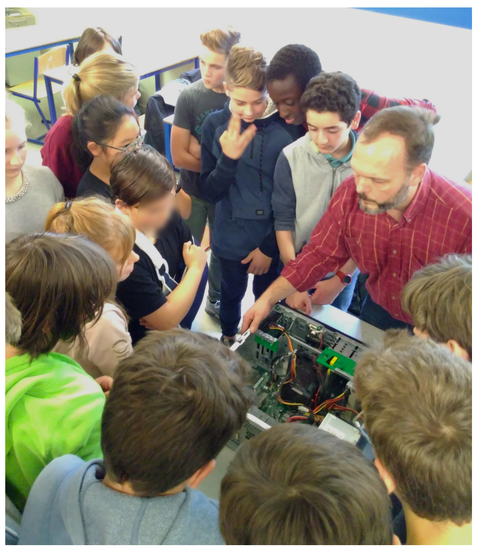


Figure 1: A teacher presenting the insides of a computer to his class

Chapter 1: The computer, this stupid machine (Fig. 1)
Objective: Understand the roles of the basic components of a computer and its limits
Activity 1: The basic components From the students representations towards a definition.
Activity 2: The computer limits Given a text in Polish, the students have to assess the difficulty of tasks such as "Counting the words" or "Write a past-tense version of the text".
Take-away: Computers are only programmable information processors

Chapter 2: This is no magic (Fig. 2-right)
Objective: Read, execute and understand a first algorithm.
Activity: A student plays the *memory* and reads instructions to a second one who plays the *processor* interacting with a third one, the *user*. The instructions are a step-by-step card magic trick where the processor will in the end reveal the card of the user.
Take-away: No magic in computers but clear instructions.

Chapter 3: Butter and jelly sandwich algorithm
Objective: Write a first algorithm.
Activity: Students write a *program* in natural language that their robot-professor will execute in order to make a butter and jelly sandwich.
Take-away: Instructions have to be unambiguous. One has to know to which granularity the program has to be written.

Figure 2: There is no magic in this trick

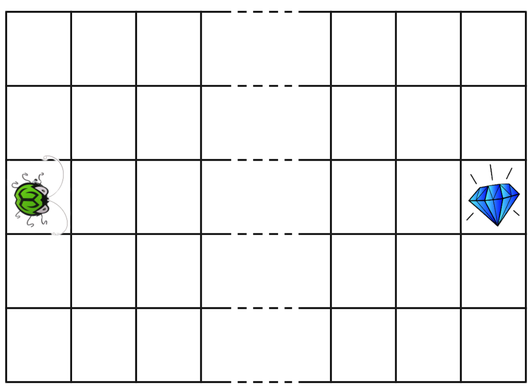


Figure 3: How to get to the diamond not knowing the width of the grid ?

Chapter 4: Find the gem
Objective: Define and use basic instructions in a program to solve a maze-problem.
Activity: The students will first choose a set of instructions then program a character on a grid. Then, once those instructions have been agreed upon by the group, the students use them to solve simple problems (e.g. like the one illustated in Fig. 3)
Take-away: Discovery of the first structures of control in basic programming

Chapter 5: Block-programming Objective: Use a set of predefined instructions in a block-programming language to code an algorithm and solve simple problems.
Activity: The students exercices block-programming in pairs on https://studio.code.org/s/course3. A selection of lessons has been made on the platform and corresponding concepts are explicited each time it's used by the students.
Take-away: Block-programming while manipulating basic programming concepts such as: sequence, repetition, decision, parameters, function, ...

Second semester: Programming as a tool in a science project

The second part of this course was dedicated to Scratch. The main objective was to use it to develop a small project in a science class context. The first part of this sequence was dedicated to discover the Scratch interface by doing some similar exercices than those proposed on code.org.
Then we asked the students to create a project in the context of the physics course. They were instructed to work in pairs and design a quizz to test their comrades knowledge on the subject of volumetric mass. First, they had to do an analysis for the project on paper before implementing their solution in a scratch project that they had to share later. Fig. 4 illustrates some resulting project matching the expected.

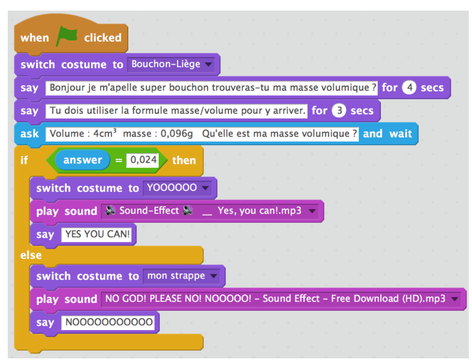


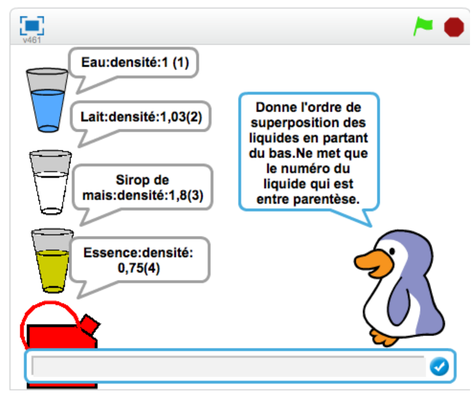
Figure 4: (left) A question is asked with the data as an image (right) A simple code to ask and check the solution of a volumetric mass problem

Results

An evaluation was made for the first semester. From the results for the different concepts evaluated (Tab. 1), we can see that the more abstract concept (i.e. "function") has been without surprise the hardest to master.

	Group 1 (n=24)	Group 2 (n=21)	Group 3 (n=21)
Success rate on topic "function"	25	81	-
Success rate on "program correctness"	70	66	-
Rate below <50%	33	10	5
Rate between 50% and 70%	33	33	24
Rate above >70%	33	57	71

Table 1: Results of the assessment for the first semester



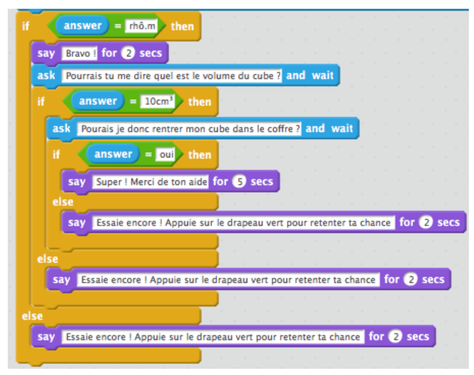


Figure 5: (left) An imaginative QCM in scratch (right) Ambiguous conditions

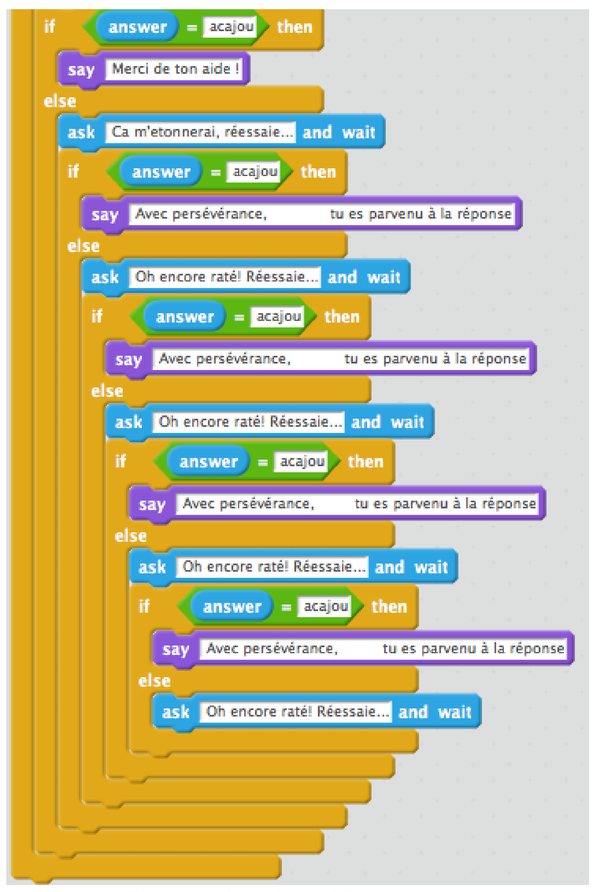


Figure 6: Looping with conditions

Most of the resulting projects developped by the students during the second semester used mainly a main sequence of instructions, one question to the user and a condition to verify the answer as shown in Fig. 4 on the right.
The main difficulties observed with those projects were:

- The design of an exercice making sense from a realistic point of view. It would not make sense to end up with a 2kg golden ring.
- The way to ask and verify in Scratch the result of a physics rule. Fig. 5 on the right illustrates it well. We can see that the expected answer is "10cm^3" while it is not specified the units in the question.
- How to re-ask a question. Fig. 6 shows a long serie of nested if-blocks instead of using a repeat-until-block.

However, some students got way further that what was initially hoped for. Table 2 presents the number of projects using in a meaningful and correct way some scratch features. An incorrect usage of a question/condition means that it is unclear for the user how he is supposed to answer to the question. An incorrect usage of variables means that they were declared but not used or not properly (for example defining an "answer" variable instead of using the blue existing one linked to the the ask-block).

	Question/Condition	Variables	Loop for answer	Sprite interaction
correct / incorrect usage	15 / 7	9 / 3	4 / 4	6 / -

Table 2: Number of projects using correctly or not some scratch features (n=22)

We can add that the feedback from the teachers was enthousiatic. Event though they had no previous background in CS being respectively two science teachers and a technology teacher. This led to some inaccuracies in the speach during class and some discomfort when confronted to some questions. But they clearly began to identify as computer science teachers and will teach this class again. Another aspect that was mentionned is the opportunity of creativity given to the students. But still with rigor and logic in link with a curriculum discipline.
The students were convinced by the course and considered it a playfull way to learn about programming and a good opportunity to rework a discipline topic. Even though the ones interviewed at the end of the year said there were to many exercices on code.org and that they saw no use for the paper analysis of their project.

Conclusion

The main contribution of this poster is to show that there is an actual interest in bringing CS to lower secundary school. As we have seen, this allows the students to reappropriate a concept from a discipline within Scratch. Moreover, as suggested by, and discussed by, it is interesting to do so in coordination with a discipline course.
We have seen in the partnership that novice teachers are more confident giving a CS class thanks to the unplugged activities in the first time. However, there was not sufficient experience on their side, nor enough time in class to allow for completion and good skills developement for the students in a more open environment such as Scratch.