

Web Platform to Support Teaching Programming with Snap! and Manage Pupils' Learning

Sébastien Combéfis, s.combefis@ecam.be

Electronics and IT Unit, École Centrale des Arts et Métiers (ECAM), Belgium.

Chantal Poncin, chantal.poncin@uclouvain.be

Computer Science Engineering Department, Université catholique de Louvain, Belgium.

Abstract

Educating young people to computer science, especially in schools, is very important. For that purpose, platforms and educational tools have been developed, in particular to teach programming. This paper is about the development of an online platform, dedicated to pupils between 10 and 14 years old, that supports them when learning programming. The platform uses Snap! with a lightweight learning management system (LMS) added on top of it. The goal of this LMS is to help teachers managing lessons and activities to build courses that they can use in their classrooms. This paper shows the structure of these courses and how teachers can build them. It also draws up features under development that will help teachers to get statistics to follow the progression of their pupils.

Keywords

Snap!, LMS, teaching programming, programming courses design

Introduction

Teaching computer science in schools, and in particular programming, is not a widespread practice all over the world. Some countries do have computer science related activities integrated in their curriculum of primary and secondary schools, but it is not the case everywhere, such as in Belgium, for example.

To offer programming courses in schools, there are two main elements to take into account. Firstly, activities must be developed and teachers must be trained to hold them in classrooms with their pupils. Secondly, software to support these activities must be made available to pupils; they have at least to be installed on the computers of the school.

This paper proposes an online web platform that addresses the two afore-mentioned needs. Activities to be used by teachers and that they can learn by themselves are proposed on an online web platform, consequently avoiding any prior installation.

Motivation

The motivation of this work is to build a platform whose goal is to introduce programming activities for the education of pupils whose age is between 10 and 14 years old. The platform must be intuitive and simple enough to be usable in countries where computer science is not present in existing curriculum and where computer science teachers do not exist.

Teaching programming to young pupils enhance several skills such as logical thinking, problem solving or structured reasoning. Europe indicated recently, in a report published jointly by



Moreover, in order to focus on the teaching, the proposed platform must not require any complex installation, and it should be possible for pupils to work on the activities at home after school if they are willing to. Those two constraints drive this work towards an online web platform. There are several possible ways to teach programming. This work is based on Snap!, a platform that supports block programming. That kind of programming, compared to more classical solutions based on programming languages such as Python or JavaScript, is more suitable for young children without any prior programming experience (Wyffels, *et al.*, 2014).

Related Work

Blockly is an open source library based on web technologies that supports block programming (Fraser, 2013). It has the advantage to allow its users to export source code of the block program in JavaScript or Dart, for example. Blockly is not an educational platform in the sense that it can be used for education as well as for business, games, etc. It is just an organisation of functions written in JavaScript into puzzle pieces that can be arranged together and executed when triggered. *Blockly Games*¹ proposes a learning path composed of several lessons, from easy ones to more difficult ones. *Code.org*² is a non-profit organisation that proposes online lessons to learn programming. These lessons are split into several steps, leading the learners from basic concepts to advanced ones. Code.org extensively uses drag-and-drop and block-based programming. The learning paths of Blockly Games and lessons of Code.org inspired the work presented in this paper.

The platform that is proposed in this work, named *RSnap*, combines the Snap! platform for the activities and a web platform developed in *Ruby on Rails* to manage the activities, and to allow teachers to follow the pupils' progress. Combining those two technologies provides a modern platform accessible from any computer with an access to the Internet and a recent browser.

² <https://studio.code.org/>.



Figure 1 shows the homepage of the platform. On this page, the pupil can choose to start a new chapter (*Chapitres*) or to launch a free project (*Projets libres*). A chapter is a set of missions that the pupil has to realise sequentially. A mission is unlocked only when the previous ones have been realized correctly.



Figure 1. Homepage of the RSnapp platform.

Mission and chapter

One chapter covers one *learning outcome* that the pupil must achieve. For example, the goal of the first chapter that has been designed is to make the pupils discovering the platform and the Snap! environment. The learner has to move a mouse seen from above the ground. In order to accompany the pupil during the learning, a chapter is split into a sequence of missions. Each mission has a sub-goal smaller than the one of the chapter, and those sub-goals are used to lead the pupil towards the goal of the chapter.

Figure 2 shows the four missions of the first chapter. Only the first one is available for the pupils, but the four missions are already visible. The progression between the missions is as follows:

- To achieve the first mission, the pupil has just to write a program that moves randomly the mouse. A block “*move randomly*” has to be used to achieve that goal.
- For the second mission, the mouse has to reach a target, the cheese, following a linear path. A block “*move forward*” has to be used three times.
- In the next mission, the path is more complex (an L-shaped path) and requires the mouse to turn left once. A block “*turn left*” has to be used once.
- Finally, the last mission requires the mouse to follow an even more complicated path that requires to turn left and to turn right. The pupil must combine the three following blocks: “*move forward*”, “*turn left*” and “*turn right*”.



RSnap Chapitres Projets libres formulaire Simon Hock	
Chapitre : La souris.	
Title	Description
Anime la souris.	Apprends à déplacer la souris sur le plateau !! 
Aide la souris à arriver au fromage !	
Attention au tournant !	Virage à gauche !
Le Labyrinthe	Attention à ne pas te perdre !

RSnap : Simon Hock et anciennement Gaetan Collart & Simon Claessens : Qui sommes-nous ?

Figure 2. A chapter is divided into a sequence of missions.

Splitting a chapter into missions makes it possible not to overwhelm the pupils with a lot of blocks at the beginning. Thanks to a feature of the platform, the pupils discover the blocks incrementally, when they are useful to them. It gives them time to assimilate and understand their purpose, before moving to the next, more advanced, mission.



Finally, as shown on Figure 3, a detailed description of the goal of each mission is made available to the pupil. The screenshot shows the guide as presented before launching the mission into Snap!, but it will still be available, as a popup window, within the Snap! environment. This description is populated with the different blocks that the pupil will learn and use in the mission.


RSnap Chapitres Projets libres formulaire Simon Hock


Mission : Anime la souris.

Dans cette toute première mission, tu apprendras à utiliser l'environnement de programmation.


Ton but sera de faire se déplacer la souris

Pour cela, tu devras utiliser le bloc **Bouger aléatoirement**. Si ce bloc est attaché au bloc **Quand est pressé** comme ceci:  la souris s'animerait une fois qu'on appuiera sur démarrer .

Pour cela, il faudra que tu déplaces le bloc **Bouger aléatoirement** de la liste à gauche jusqu'en dessous du bloc **Quand est pressé**. Assure-toi qu'il est bien attaché comme ceci:  **Bouger aléatoirement**

Pour lancer ton programme appuie sur le petit drapeau vert en haut à droite de l'écran .

Tu as aussi le choix de faire avancer avec la souris avec le bloc **Avancer**.

Une fois que tu as fini, tu peux cliquer sur  tout au dessus à gauche de l'écran et sélectionner "Passer à la mission suivante". Fais attention de bien sauvegarder ton travail !

Réaliser cette mission

Retour

RSnap : Simon Hock et anciennement Gaetan Collart & Simon Claessens : Qui sommes-nous ?

Figure 3. Each mission has a guide sheet to help the pupil to achieve it.

The integration between the LMS providing the chapters/missions structure and the Snap! environment is very important and drove the design of RSnap from its beginning. RSnap is not



just a heterogeneous juxtaposition of the two components, it is an effective integration whose goal is to go through the chapters and missions, hand-by-hand with the pupil. As it can be seen on Figure 4, one way to achieve this integration is by providing menus to go from Snap! to the chapters/missions structure, for example.

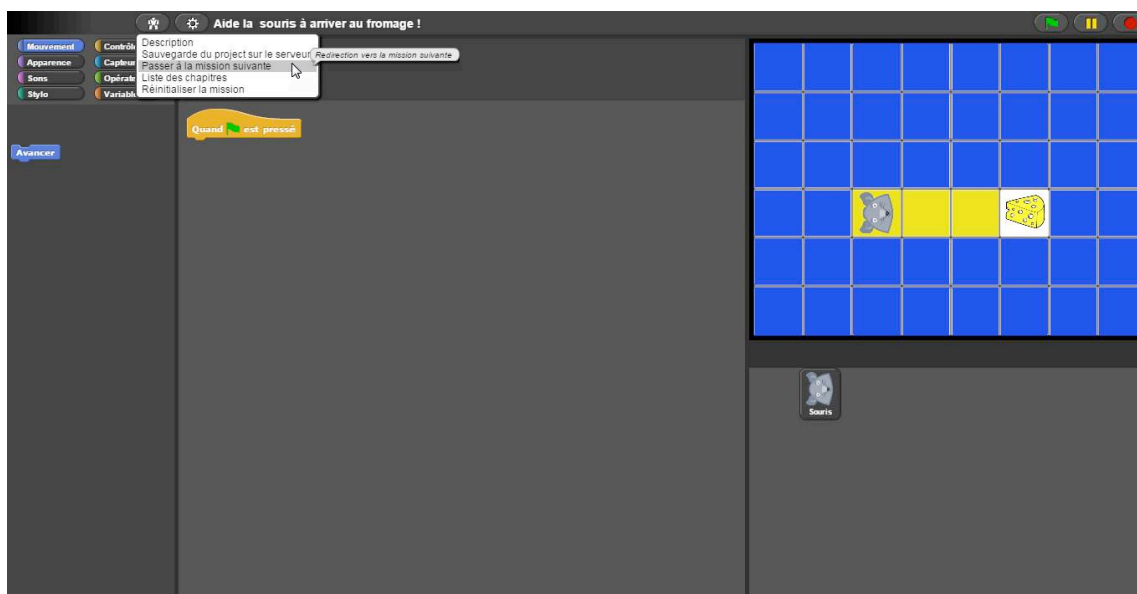


Figure 4. A menu makes it possible to navigate back to the chapters/missions structure.

Checking success of a mission

RSnap proposes a new kind of block, in addition to those existing in Snap!, that is used to check whether a pupil succeeded a mission. The *start block* is activated when the pupil starts his/her program. The *end block* is activated when the program of the pupil has finished its execution. That one is useful for the teacher since it allows him/her to check whether the mission has been successfully completed. For example, the teacher can check whether the character has reached the right location, or has drawn the required drawing, etc.

Teachers' view

The platform also provides support and help to the teachers. First of all, it is possible for teachers to create by themselves new *missions*. The teacher can choose to design a new mission from scratch or to start from an existing one, creating a copy of it. It offers the possibility for a teacher to create a variant or to improve an existing mission. A mission is composed of two parts:

- The LMS part provides a title, a description and several options;
- The program part of the mission is designed on the Snap! interface.

The description of each mission is defined as HTML content that can contain links and pictures, everything being stored in a database. Concerning the design of the program part, teachers have access to more options in the Snap! interface, allowing them for example to decide which blocks to show or hide to the pupils. It is therefore with that feature that teachers can build missions incrementally showing blocks to the pupils.

The platform also offers the ability to organise missions into *chapters*. After having created a new chapter, a teacher can create new missions for it or integrate existing ones. This opens



collaboration possibilities between teachers that can share their ideas of learning paths. A teacher will have access to the pool of all the missions that have been created, and can therefore pick a mission in this pool to populate his/her chapters.

The topmost level of organisation proposed by the platform is *courses*. A course is a set of chapters, for which a pupil can be registered. This level is the one at which a teacher will be able to follow pupils and will get statistics about their progress. For each pupil, teachers can monitor in which chapter they are and which missions they achieved. Given those statistics and monitoring tool, the teacher can precisely follow up how the pupils are performing. It is a precious feedback for the teacher, especially when it is his/her turn to provide feedback to the classroom. The teacher can also manage the chapters/missions that are accessible to pupils, for each course. That feature makes it possible for the teacher to control the progression of the pupils, and to avoid them working on their own on more advanced missions before feedback sessions in classroom, for example.

Technologies

RSnap is built on existing widespread technologies used in web applications development. The block environment that has been used is Snap! because it is open source and uses JavaScript, which makes it easy to adapt and integrate with other web technologies. For example, the interface has been customised by removing some unnecessary features, by adding new menus to cooperate with the chapters/missions structure pages, by adding new blocks, etc.

The chapters/missions structure part of RSnap has been implemented with Ruby on Rails (RoR) following the Model-View-Controller (MVC) pattern. Using RoR features, a management of roles has also been added to the platform, to propose two different roles: pupil and teacher. Details about the implementation are available in (Claessens, & Collart, 2014), and RSnap is open source and is available at the following address: <https://github.com/simonhock/rsnap>.

For now, only the program part of missions can be exported/imported thanks to the Snap! platform. The elements added in this work, and which are stored in a database, cannot be exported/imported for now. A direct consequence is that all the teachers that would like to collaborate must use the same instance of the platform installed on one unique server.

Evaluation

The proposed platform has been evaluated by proposing work sessions organised with teachers with their classrooms. Those experiments have been organised during the “*Printemps des Sciences*”, a week during which activities to promote sciences are organised with primary and secondary schools. Five groups of pupils worked by pair during 1h30. Two classes were from fifth year primary school (10-11 years old), one from first year secondary school (12-13 years old) and one from second year secondary school (13-14 years old).

In addition to oral feedback that was collected during the activities, that allowed us to improve the platform, a questionnaire has been submitted to pupils to collect their impression about the platform. The questions were asked to get information about the difficulty, the success, the entertainment and the understanding.

A preliminary analysis of the results of this survey has been performed on the second year secondary school group. Figure 5 summarises the results. The level of perceived difficulty increases from the first to the fourth chapter, which is expected whereas successive chapters cover more and more difficult new concepts. The success is of 100% for the first chapter, but



pupils were not able to succeed completely the other three chapters. An explanation could be that the increment between the first chapter and the other ones is too large, too much new concepts being added without being understood well by pupils. The different chapters were entertaining for the majority of pupils, and finally their understanding was not complete. It is surely correlated with the success measure.

Looking at the programs they produced showed that even if the pupils succeeded the first introductory missions, they did not manage to get the best solution for example missing the opportunity to use a loop. However, those concepts were correctly used in subsequent chapters, meaning that pupils maybe needed more missions to understand correctly some concepts. Such analyses are useful and must be performed to better design chapters and missions.

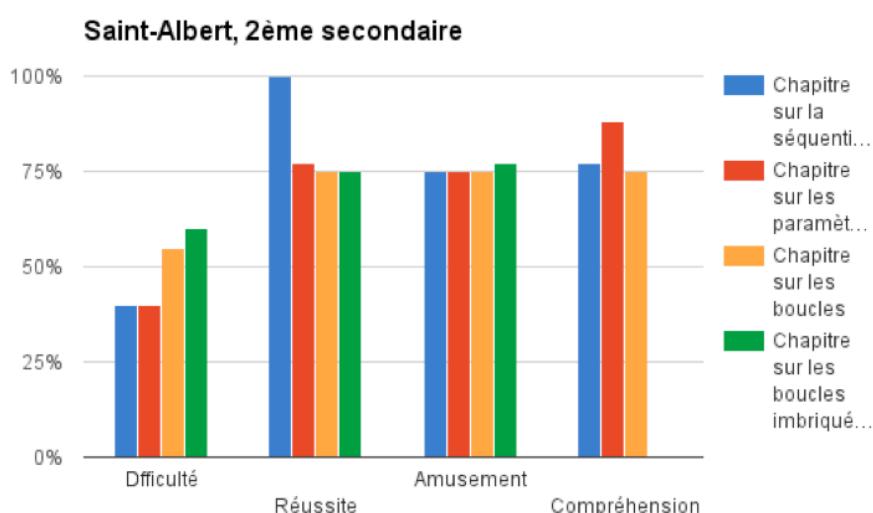


Figure 5. Results of the survey proposed after the “Printemps des Sciences” activity to the second year secondary school students, for the four chapters they worked on, successively measuring difficulty, success, entertainment and understanding (pupils had to answer with yes/no).

Moreover, among all the participants, 55.3% of the pupils told that they would like to continue working on the missions after the 1h30 session, in their school or even at home. Only 15.8% told that they do not want to continue to work on the missions and the others have no opinion. We indeed noticed that half children connected again on the platform after the “Printemps des Sciences” week. They were maybe proud of their programs and showed them to other people, or wanted to achieve more chapters/missions.

The main lessons learned from the informal oral feedback are:

- the activity allowed the pupils to get a better understanding about what programming is;
- most pupils declared that they have fun playing with the activities;
- and finally, teachers were very happy and impressed by the quality of the activities and discovered a new field that they are willing to use in their classrooms.



Conclusion

The proposed platform fulfils its goal, that is, providing activities that teachers can make with their pupils to teach programming and introduce them to computer science. The platform, which is released as open source, does not require any installation difficult for the teachers since it can be accessed through a browser. The split of chapters into a sequence of missions supports the incremental learning of new concepts. Pupils are accompanied and teachers can track their progression, which allows them to provide precise and individual feedback to each pupil.

Improvements are still being done to the platform. This latter will be used to build a training of teachers, so that they can afterwards use it autonomously in their schools. Future work includes adding a full export/import feature for chapters and missions. Tools to follow the progress of the pupils of a course and to get statistics must be developed. Finally, the survey that has been lead during the “*Printemps des Sciences*” event must be deeply analysed to improved the already designed chapters and missions.

References

- Fraser, N. (2013). Google Blockly: a web-based visual programming editor. Available from: <https://developers.google.com/blockly/>. US: Google.
- Claessens, S., & Collart, G. (2014). Plateforme web pour accompagner l'apprentissage de la programmation par les 10-14 ans. Belgium: Université catholique de Louvain.
- Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., Drechsler, M., Mendelson, A., Stephenson, C., Ghezzi, C., & Meyer, B. (2013). Informatics Education: Europe cannot afford to miss the boat. Report of *the joint Informatics Europe & ACM Europe Working Group on Informatics Education*.
- Harvey, B., & Mönig, J. (2015). Snap! Reference Manual. US: Berkeley.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: A sneak preview. In *Proceedings of the 2nd International Conference on Creating, Connecting and Collaborating through Computing*, (pp. 104-109). US: IEEE.
- Wyffels, F., Martens, B., & Lemmens, S. (2014). Starting from Scratch: Experimenting with Computer Science in Flemish Secondary Education. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, (pp. 12-15). US: ACM.

Copyright

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license (CC BY-NC-ND 4.0). To view a copy of this licence, visit <https://creativecommons.org/licenses/by-nc-nd/4.0/>