Programming as learning resource in middle school

Ricardo P. Salvador[1], Claudia Pons[2 3] and Guillermo L. Rodíguez[4]

[1] Escuela Superior de Comercio - Universidad Nacional de Rosario - Rosario - Argentina

[2] Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA) - Universidad Nacional de La Plata - La Plata - Argentina

[3] Centro de Altos Estudios en Tecnología Informática (CAETI) - Universidad Abierta Interamericana - Rosario - Argentina

[4] Facultad de Ciencias Exactas, Ingeniería y Agrimensura – Universidad Nacional de Rosario – Rosario - Argentina

rsalvado@unr.edu.ar, cpons@info.unlp.edu.ar, guille@fceia.unr.edu.ar

Abstract. *In this work we study the influence, in the kinematics learning of secondary students, of the construction of a simulation using the Squeak-Etoys graphical programming environment, as learning resource in a Rosario (Argentine) city preuniversity middle school. The pupils are allmost 16 years old and don't have knowledge about programming. Using an habitual curricular exercise the experience took about 3 classes. The results indicate an increase in students' grades and interest in programming as a didactic resource, which encourages designing similar activities and exploring this resource in other disciplines.*

Keywords: *programming, simulation, learning, middle school, physics*

1. Introduction

This paper explores the usefulness of students' design and development of computer models and simulations as a learning resource in middle school, in a discipline other than those related to programming itself, using a user-friendly programming environment (Squeak-Etoys). The contribution sought is to propose a methodology for designing activities in which students should model and simulate school contents.

Section 2, *theoretical framework*, provides the programming concepts and the use of simulations in Bloom Taxonomy, explains the possibilities offered by the construction of simulations as a didactic resource, compiles the contributions of constructionism to education through computers, presents references to the correlation between programming and development of computational thought, highlights initiatives of national and international scope promoting computer science and programming, discusses particularly significant related works and characterizes the Squeak-Etoys programming environment. The *methodological framework*, resources and tools used are set out in section 3. Section 4 describes the *results*: the characteristics of the school

where the fieldwork was carried out, the groups of students participating, the meetings that made up the experimental activity and the results of the diagnoses and evaluations related. The *discussion* of the results is developed in section 5. Finally, the *findings* and *foresight* are presented in section 6, and the *bibliography* is summarized in the seventh and final section.

2. Theoretical Framework

2.1. Intellectual and digital skills

In the 1950s Benjamin Bloom defined a taxonomy of educational objectives and intellectual abilities for teaching planning. This taxonomy has two characteristics: *a)* objectives or thinking skills are ordered in a sequence of increasing complexity, from lower order skills to higher order level, and *b)* mastering one of these skills implies mastery of the preceding ones. Lorin Anderson published a review in 2001; Bloom's Revised Taxonomy, which begins with the ability to *remember* as the simplest of all. It continues with the ability to *understand*, *apply*, *analyze*, *evaluate*, and finish with the ability to *create*, wich is the most complex and whose domain would imply dominion of all previous ones. In 2009 Churches adds a series of skills from working with Information and Communication Technologies, including the *use of simulations* (associated with *evaluation*) and *programming* (associated with to *create*)[Churches 2009].

2.2. Computers, simulation and education

Computers and simulations offer the possibility of particularly active learning, a subject on which we have contributions from Seymour Papert (1999) and Alan Kay (2007).

The modeling and implementation of system simulations is an activity that is used for the study of complex and large-scale situations. Taking its approach into account, articulating it with the programming environments (increasingly powerful and usable) and hardware microcontrollers (increasingly and accessible), it can be a valuable resource to enrich teaching-learning situations, incorporating other dimensions to the usual modes of representation: time, movement, multimedia resources, and design and interaction with these representations.

Analyzing a situation from a systemic point of view implies not only knowing the system, its components and its environment in a descriptive way, but also its properties and the processes through which they are dynamically related and modified among themselves and with the environment. Model construction focuses on the relevant aspects of a system for a certain study purpose[Gordon 1980]. And a simulation is the implementation of a model[Kelton 2008]. Therefore, to consider the construction of simulations as a learning resource implies for the student: *a)* system understanding, *b)* abstraction of elements relevant to modelling, *c)* use of resources and skills related to computational thinking in the implementation of simulation, *d)* represent knowledge and interact with that representation, involving trial/error, hypothesis testing, *e)* interact with others in the development and communication of the activity.

Developing an application for this purpose makes it possible to evaluate the operation of the simulation, intervenes in it and eventually improves it. This is important from a didactic point of view, since the student (or group) performs a complex and integral task with respect to the object of study, carrying out a set of cognitive skills ranging from the simplest to those of greater intellectual complexity.

2.3. Contributions of Constructionism to Computer Education

Seymour Papert created the concept of *constructionism*, which he defines as giving children good things to do so that learning takes place by doing them, as he thought that instead of just interacting with technology they should learn to program their own animations, games and simulations, and in the process learn problem-solving skills and project design strategies[Resnick 2012].

He support the idea that learning occurs indirectly, using knowledge and not just memorizing it. In other words, the way we learn best is through the action of building something external to ourselves: building a tower, writing a story, building a robotic artifact, programming a video game, or making an animation are all examples of building. These kinds of activities are valuable because of what Papert calls "proof of reality": if they don't work, they are a challenge to understand why, and to overcome obstacles. They can be shared and discussed with others. And they serve as transitional objects for the personal appropriation of these ideas [Papert 1999].

In developing the constructionist approach, Papert laid the foundations for teaching with digital technologies, recognising that in the world changes were increasingly rapid , giving a preferential place to the student's interests, for which new knowledge must have a current meaning, valuing the confrontation of diverse and alternative ways of thinking, giving programming a key place, assuming that students can learn to program since an early age, where situations will arise. This, in turn, makes a teacher learn at the same time as his students, rescuing the simulation capacity of computers as an educational resource.

2.4. Computational thinking

Computational thinking is intellectual activity related to the formulation of a problem in such a way that it can admit a solution by computational means, while the solution can be carried out by a human being or a machine, or both[Wing 2010].

Kafai and Burke (2014), point out that programming is now recognized as an answer to the need to develop computational thinking in students because they can solve problems, design systems for everyday life, and generate progress in other disciplines.

2.5. Initiatives promoting Computer Science and Programming in schools

In recent years, the teaching of Computer Science in schools and the teaching of Programming have been revalorized and used as an educational resource and as a general competence in different parts of the world. For example, Raspberry Pi (www.raspberrypi.org), Scratch (scratch.mit.edu), Squeak-Etoys (squeakland.org/about) and Code. org.

In Argentina, the National Education Act No. 26,206 (2006) promotes

information and communication technology skills (Chapter 11, paragraph m), and the teaching of "Programming" is declared of strategic importance by the 65th Assembly of the Federal Council of Education (Resolution 263/15, art. 1°). Program.AR is an initiative of the Sadosky Foundation, created in 2012, and aims to bring Computer Science learning to schools (http://program.ar/quienes-somos/). "La hora del código" (joint Program.AR and Code.org initiative) offers resources for learning to program in a fun and entertaining way as a way to approach Computer Science.

2.6. The Importance of teaching Computer Science in Argentinean Schools: Sadosky Foundation report

According to the report of the Sadosky Foundation "*Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas*", programming provides tools to stop being mere consumers of software, understand how it works and the digital world in general, and improve intellectual skills. For this reason, the teaching of programming in schools is proposed, not only focused on the teaching of a particular language, but also on the development of computational thinking, as a basic skill in our society (Fundación Sadosky 2013).

2.7. Background

In order to collect background information on the use and/or construction of simulations in education, a search in February 2016 resulted in a list of 17 works related to experiences with computer simulations in formal education, which included programming as an object of study, as a didactic resource or as a school content. Of the 17 publications, 3 papers refer to experiences in which students designed and implemented a simulation as a secondary school learning strategy.

Faced with the observation of difficulties in working with physical formulas, Taub and others (2015), measured the influence of programming simulations of systems related to physics, using JAVA in a user-friendly environment, with groups of tenth and eleventh grade students from different schools and selected based on their performance. Developed over three years and often weekly, the study results indicate that programming played an important role in physics learning.

Yohan Hwang, Kongju Mun, Yunebae Park (2016), in a study of perception of programming, computational thinking and attitude about science learning in high school students with Scratch (programming) and BitBrik (physical computing), found an improvement in perception of computer programming and thinking, self-managed thinking and interest in activity.

Lye, S. Y., Koh, J. H. L. (2014), in a review of 27 studies on the development of computational thinking through programming, observed the importance of programming, which exposes students to computational thinking. And that the availability of free and easy-to-use programming languages meant an impulse to investigate the introduction of computational thinking in K-12 contexts. They propose that to encourage such practices, learning environments based on constructionalism and integrating information processing, scaffolding and reflection should be further researched and designed.

Also relevant is the work of Resnick and Brennan (2010), who developed a framework for computational thinking and how to assess achievements on it. Based on their interest in how Programming can help develop computational thinking, they drew up a list of seven computer concepts common to it and to most computer languages: sequence, cycle, event, parallelism of sequences, conditionals, operators and data. They also observed design strategies around the activities they studied, in relation to computational thinking: being incremental and iterative, rehearsing and debugging, reusing and remixing, and ultimately abstracting and modularizing. Study participants reported that they experienced changes in expressing themselves using technology, connecting and interacting with others as a way to enrich their projects, and formulating questions and answers related to technology and learning from programming.

2.8. Experience of teaching programming in middle schools in La Plata

Queiruga and Fava (2013) directed an extension project of the Computer Science Faculty of the National University of La Plata (province of Buenos Aires, Argentina) carried out within the framework of articulation actions with technical schools of the Province of Buenos Aires for students and teachers in which programming and JAVA were taught in order to *a)* strengthen technical education, *b)* improving the teaching of programming and bringing programming closer to students through an innovative, playful and social approach, *c)* improving the articulation of secondary school and university.

The project's activities consisted of updating teachers on object-oriented programming and JAVA and Eclipse (development environment), and generating didactic strategies for the implementation of these contents in the classroom. At this stage of the work speculated on using RITA (Robot Inventor to Teach Algorithms, RITA in the JETs project,[http://jets.linti.unlp.edu.ar/rita]) for the initiation of students into programming, due to its easy use for adolescents. RITA is a programming environment in which virtual robot combat strategies must be designed; it integrates OpenBlocks (http://education.mit.edu/openblocks) and Robocode (http://robocode.sourceforge.net/) and code is generated by dragging and dropping blocks similar to Etoys and Scratch.

Teaching materials were also produced to support the teaching of secondary level programming for teachers and students.

Outstanding results of the project are *a)* interest of teachers at both levels in continuing similar activities on a regular basis, *b)* JAVA teaching was delayed to the last stage of technical secondary school (5th grade), *c)* RITA was used in the third year of the Basic Cycle, *d)* the tests using RITA were conducted with selected students (male and female) aged 16 to 18 from the three schools participating in the project, *e)* after testing more than 90% of the students surveyed had no difficulty in using RITA and found it easier to program with that environment than with traditional environments. In some cases it was experimented with logic circuits

Based on the convergence of problem-solving, the playful element (the robot combat) and programming, Queiruga frames its work with the concept of "serious play", which he defines, citing Sánchez Gómez (2007), as "objects and/or learning tools that

have in themselves pedagogical, didactic, autonomous, self-sufficient and reusable objectives that enable players to obtain a set of predominant knowledge and competencies". Serious play, he says, is a confluence of interactivity and fun, valued by young people, while promoting tolerance for frustration, the ease of relating to others, and high motivation for achievement; all characteristics that make serious games "an ideal technological element for transmitting knowledge" [Queiruga and Fava 2014].

2.9. Squeak-Etoys: a programming environment designed for education

Squeak-Etoys is a multimedia authoring system specially designed to learn ideas by building them. Inspired by LOGO, Smalltalk, Hypercard and Starlogo, he developed from Squeak at Walt Disney Imagineering (the research and development arm of The Walt Disney Company) and was later supported by the Viewpoints Research Institute (VPRI) as one of his projects. Etoys presents a unified style, user interface, media and programming environment for building "things" with computers[Kay 2007]. Integrated in the laptop software of the One Laptop Per Child project, this cross-platform application (Linux, Windows and Mac) supports objects (text, bitmap and vector images, sounds) that can be edited, sensing events, communicating with other objects, and including program code. Its simplicity is proportional to the possibility of obtaining results of great didactic quality: by facilitating code writing, it allows the user to focus more on the problem that must be solved when implementing its simulation, and to worry less about the problems of writing code.

The Etoys graphical interface consists of a desktop called *world* (figure 1), at the top of which there is a *toolbar* (figure 2) that provides objects (figure 3), the basic operations of creating, opening, saving and navigating between projects, and application help. Every object is graphical and from it you get a contextual menu in the form of icons that surround the object, called *halos* (figure 4). One of them opens a panel called *viewer* (figure 5) from which the attributes, instructions, scripts (figure 6) and variables of the selected object are obtained.

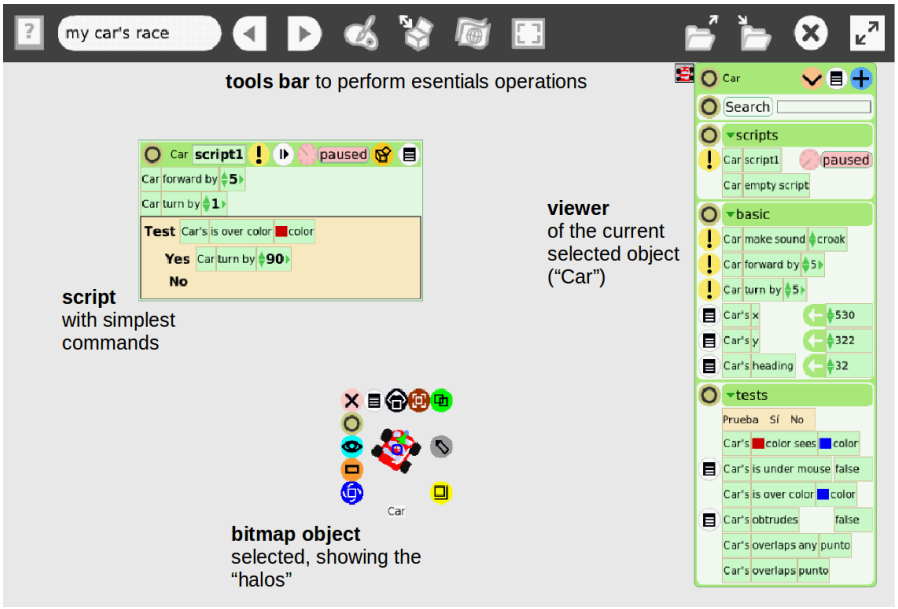

Figure 1: "world" of Etoys

The toolbar is the starting point for the most common operations: it shows the name of the project, and buttons for drawing, getting objects for the project, changing the language of menus and instructions, opening and saving projects, switching to full-screen mode, and exiting Etoys (Figure 2).

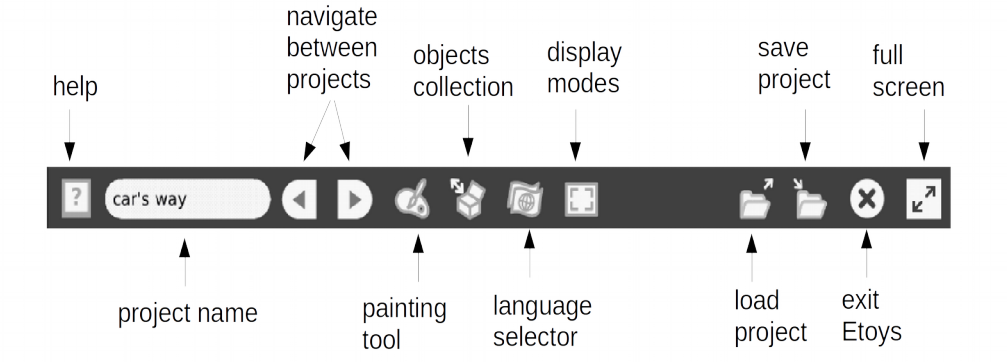

Figure 2: the "toolbar"

Provisions (figure 3) is the name given to the collection of morph objects provided by Etoys. Clicking the corresponding button on the toolbar displays a list of some commonly used objects, which in turn contains the "Catalogue of Objects", which gives access to all objects, by means of categories, sorted alphabetically, or by typing the name of the object.

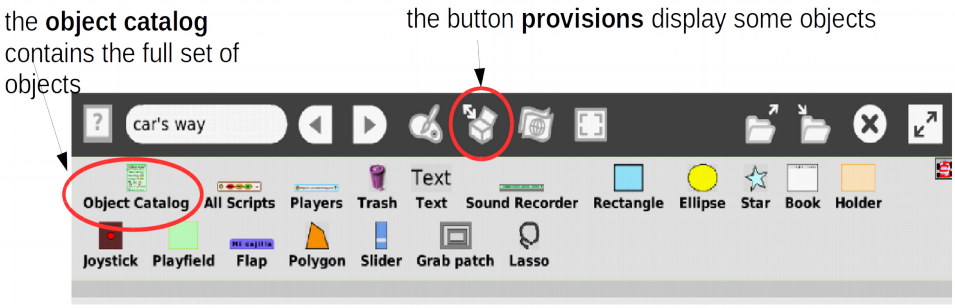

Figure 3: button "Provisions" and the "object catalog"

When an object is selected, its name ("auto", in this case) and halos (figure 4) are shown, as well as a series of icons of different colors and functions: display a menu, move the object, duplicate it, change its size, rotate it, open a "viewer" of the object that is a menu that shows which scripts that object has, if variables have been added to it, its attributes and procedures or operations that the object can perform, and its attributes. According to the type of objects, halos can vary depending on their properties: for example, for bitmaps there is a halo to edit them, and for texts, there are halos to modify font and size.

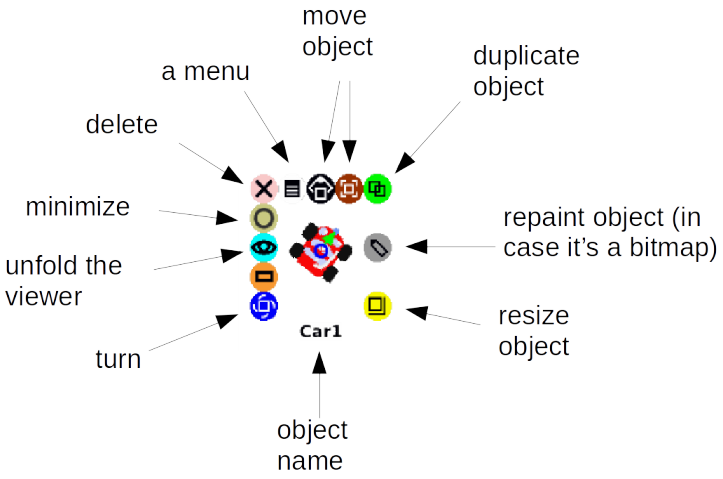

Figure 4: a bitmap object and his "halos"

The *viewer* (figure 5) is the basic tool for generating code: it provides the mosaics of instructions and attributes, and control of scripts. From a viewer you can modify the attributes of the object and execute instructions without generating a hyphen. In a viewer, mosaics are grouped into categories specific to each type of object; for example, the object "sector" has a mosaic that allows you to obtain and establish the value of its angle. Other categories are common to all objects, and refer, among other things, to variables, scripts, color, motion, sound, geometry, and conditionals.

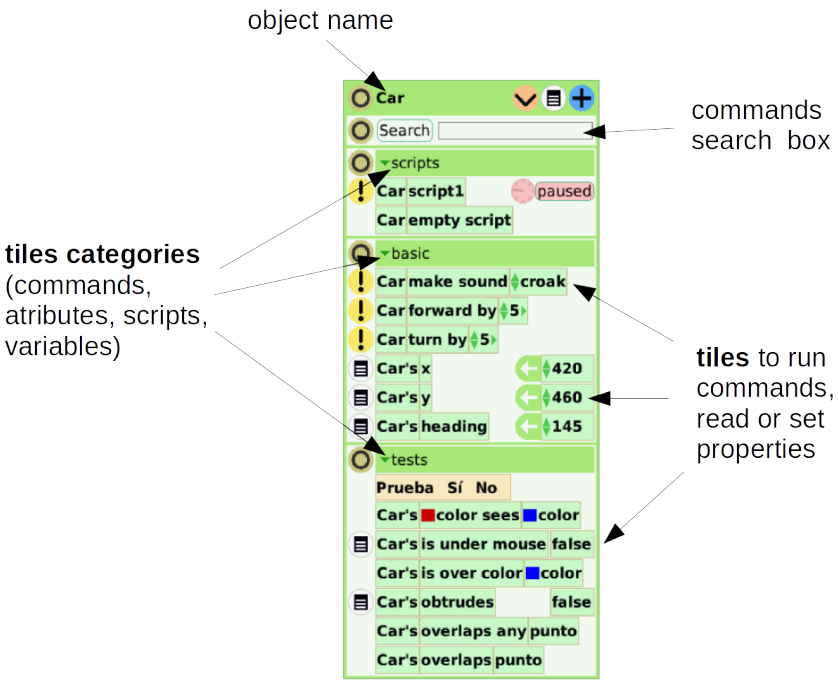

Figure 5: The "viewer"

Scripts (figure 6) are constructed with mosaics obtained from the viewers; once inside the script, each mosaic representing an instruction can be modified using other mosaics or typing inside. The instructions consist of words in the selected language, making it easier to build scripts, and become gradually familiar with all the diversity of objects, and categories of objects, instructions and attributes. The sequence of instructions in a script can be executed once or repeatedly.

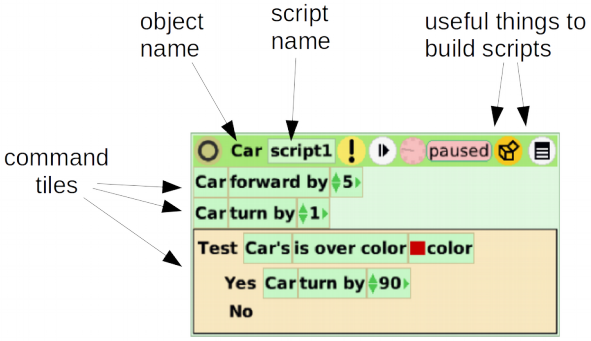

Figure 6: A simple script

3. Methodological framework

The fieldwork consisted of evaluating the effect of the use of modeling and simulation in the construction of a program as a learning resource, through a pre-test/post-test protocol in which a control group and an experimental group participated. A Physics teacher was contacted and facilitated the participation of two of his courses, 30 and 24

students, in the roles of control group and experimental group. The members of the latter group developed the simulation on a Uniform Rectilinear Movement "encounter" exercise.

Prior to carrying out the experimental work, both groups were diagnosed *a)* a diagnosis of the control and experimental groups in order to know characteristics of the group regarding the use of digital technological resources, how to tackle problems, knowledge about modeling and opinion of its usefulness in learning, and *b)* a survey on kinematics contents, with questions on Uniform Rectilinear Motion.

The experimental activity was carried out in the classroom where the group usually has class; netbooks of the Connecting Equality Plan were used, which include Squeak-Etoys. It was foreseen that the work would be done in teams, especially to encourage cooperation in the face of a new resource.

An interactive tutorial was used, designed to provide guidance for the activity and workspace in a single window, so that changing windows continuously was avoided. The activity slogans and guidelines for the activity were also printed to provide an alternative source, in case there was any difficulty in reading, or if you wanted to remove the tutorial guide for more screen space (especially considering that the netbook screens are only 10.1 inches).

Since the students had to intervene by modifying and creating scripts and code, it was expected that many unforeseen events would arise, so the guide was written with the expectation that it would promote autonomous work and that the teacher's intervention would be mainly ready to observe the work of the groups and intervene in emerging situations.

The interactive tutorial was developed using a Squeak-Etoys resource called "*book*" that allows you to add pages, similar to a multimedia presentation, with the difference that each slide can also contain objects with their scripts and the presentation can be intervened while it is being "used" (this means that changes can be made to the presentation, the objects involved and their scripts; there is no difference between editing and playback/reading instances as in other environments of the same type.

On each slide or tutorial page *a)* shows *information on the exercise* you are trying to resolve *b) programming concepts* that are at stake are highlighted, *c)* at the same time it proposes *challenges* related to its simulation, and clues for its resolution, which is done by creating or modifying scripts or program scripts within each slide, without leaving the "book" environment, *d)* by *modifying or creating new scripts while "reading" the tutorial within the same visual space* (there is no distinction of author-reading modes in Etoys), it facilitates the students' work by receiving immediate feedback on their activity, e) there are tutorial *navigation buttons and a progress bar* that indicates progress within the slide show sequence (figure 7).

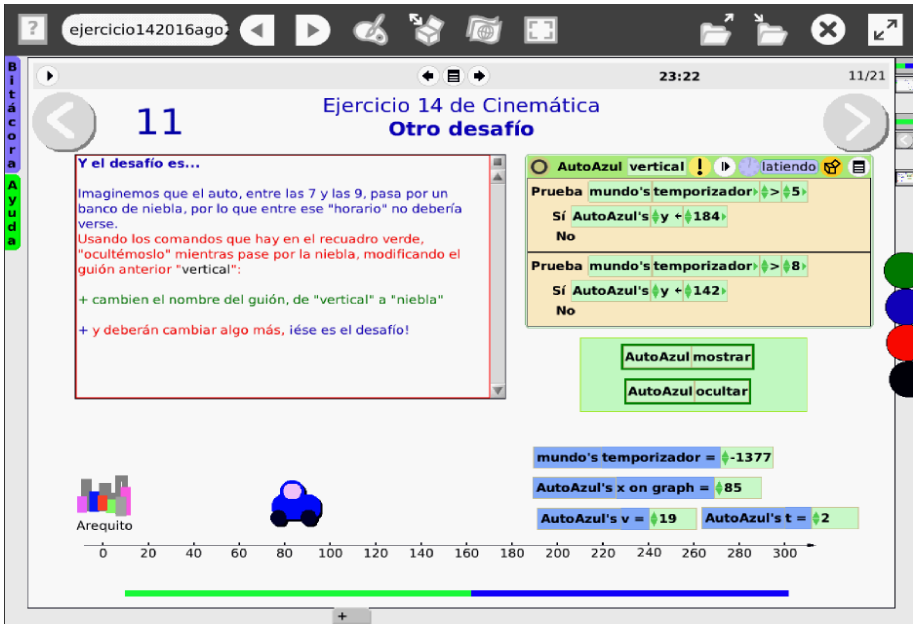

Figure 7: A slide of the interactive tutorial

In addition, text in four colors was used to indicate different message types: *a)* green text, was used to show *operations that students should perform with Etoys*, such as building and modifying scripts, or configuring car parameters, *b)* blue text, to highlight *reflections on the problem* being solved, or on programming resources, *c)* red text, to raise *challenges* and advance the activity, *d)* black text, to refer to *Etoys concepts* such as variable names, instructions or scripts.

The tutorial was organized in a series of partial problems, to an increasing degree of complexity regarding the use of the programming environment as each stage progresses towards the resolution of the exercise by constructing different stages of the simulation. It can be downloaded from the follow link http://hdl.handle.net/2133/13002 at Repositorio Hipermedial de la Universidad Nacional de Rosario.

4. Findings

4.1. The school

The fieldwork was carried out at the Escuela Superior de Comercio "Libertador General San Martín", dependent on the Universidad Nacional de Rosario. The school offers the orientation "Marketing". Situated in the center of Rosario city (Santa Fe province, Argentina), it has a population of approximately 1200 students grouped in commissions of thirty, which are distributed in two shifts (morning and afternoon), five years and four divisions.

The academic structure of the School groups related curricular spaces in Departments directed by a Head and an Assistant or Area Coordinator, who coordinates each particular curricular space.

The "Physics" curricular space is present from 3rd to 5th year, with a weekly 4-hour teaching period; the didactic unit corresponding to "Kinematics" is the first of 4

years.

The "Computing" curricular space: the School has this curricular space from 1st to 4th year, with a weekly 2-hour teaching period in each year. The contents of this curriculum do not include Programming or Algorithms.

4.2. Experimental activity

Prior indications

At the beginning of the first of the three classes that took place during the activity, the following criteria were identified, although they were recalled in general and each group in particular, during the whole experience, and whenever deemed necessary: *a)* It was pointed out that there would be unexplained things, either intentionally for them to intuit, imagine and propose (and put it into practice), or to interact by asking questions, *b)* It was stressed that the activity works best with a proactive attitude, so the students were asked to try and test this material and their own assumptions, without expecting to do everything "right"; or wait for absolute certainty to start working: nothing's going to break! *c)* It was also pointed out that the activity contains instructions on each page of the tutorial so that each group can work at its own rhythm, and when needed, of course, call the activity coordinator.

Development over time

The start of the pilot activity was scheduled for the end of April 2016, with a duration of 2 to 3 modules of 80 minutes, each on weekly bases, expected to be completed. Due to unforeseen events that disrupted the school calendar, the activity took place during three classes on June 9 and August 5 and 12.

At the beginning of the *first class* there were still 3 netbooks blocked, although every day of the previous week the course was taken to unlock the netbooks, and those who still had them blocked were asked to do so (the netbooks used have a security device that forces them to use them periodically inside the school in order to avoid the blockage referred to in this paragraph).

For the development of the activity, it was necessary to load the interactive tutorial (a 1.1 Mb file), so it was planned to download it from the Internet, or from two pendrives, an option that was used because there was no WiFi and the mobile phone signal was not enough to implement an access point from a cell phone. There were 10 netbooks, and the task of copying the tutorials took almost 40 minutes out of 80 minutes of the class, partly because some machines didn't recognize the pendrives (or took a long time to do so) and others didn't have the original platform configuration and Etoys had to be installed.

In the little more than 40 minutes of activity, the 10 groups that were formed worked showing interest, with some distractions but exceeding 50% of the activity.

This class included the presentation of the activity, its objectives, the way of working and the programming environment that was used, on which the tutorial includes instructions.

In the *second class*, work continued using a modified version of the tutorial, in

which editorial corrections were made.

On this occasion, most of the groups reached the stage of the work in which the behavior of the cars "protagonists" of the simulation is fully simulated. Since the two cars involved in the exercise respond to the same pattern of behaviour, once one of the cars has finished, it is only necessary to duplicate and modify its name, the parameters relating to speed, on time, and appearance (the latter, to visually differentiate it from the first car). When it came to modifying the look (one of the exercise cars is blue and the other is green) almost everyone made a parenthesis to change something more than the color to their new car.

In the *third class*, most of the groups solved the challenges of the penultimate slide, in which they simulated the situation posed in the exercise; however, not all of them finished the activity of the last slide, where they had to obtain and show the answers to the exercise about time and distance of the encounter between the two cars. When reaching the middle of the class, an evaluation of each group's work was made and when the majority of them found it difficult in the last two challenges, it was resolved to explain them to the whole group using a projector, so that each group could then try to finish their work.

Throughout these experimental classes, motivation was shown and concerns were expressed about how to change other aspects of the simulation, which in principle was not foreseen as part of the activity: a) how to modify the appearance (colors, shapes) of the simulation, b) simulate another type of movement (carrying out changes of direction, describing curves, for example), c) also questions arose about the possibility of creating simulations in other curricular spaces, about the programming environment used and about how to learn how to program. It was observed that learning the user interface, at the beginning of the experience, took up a lot of time that decreased in the following classes.

4.3. Survey results

Control group

In this group, including of 62% of female students and 38% of male students, 70.37% stated in the pre-test that they had used software at school to represent situations that are usually done with pencil and paper, 48.15% to solve problems and 18% to build applications (Figure 8).

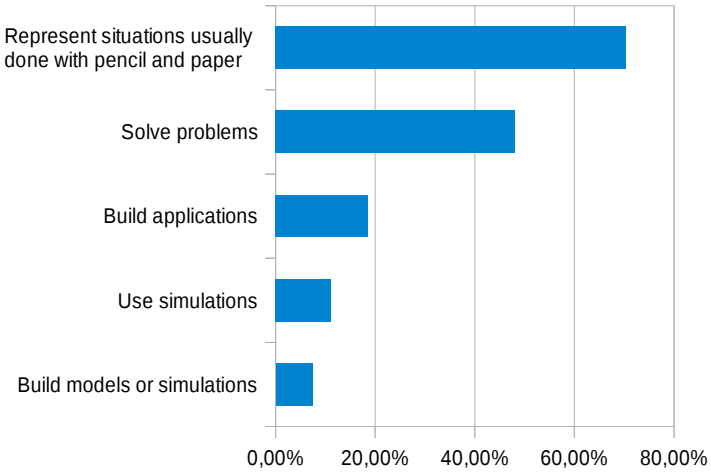

Figure 8: Control group: students that had used software at school to...

About use of computer technology to solve problems on a daily basis, 88.88% said they do it, and 55.56% prefer to solve problems in a group setting.

In comparison to the question on the concept of simulation, in the pre-test 29.63% answered correctly, while in the post-test 46.67% answered correctly (figure 11). Regarding the concept of model, 18.52% answered correctly in the pre-test and 40% in the post-test.

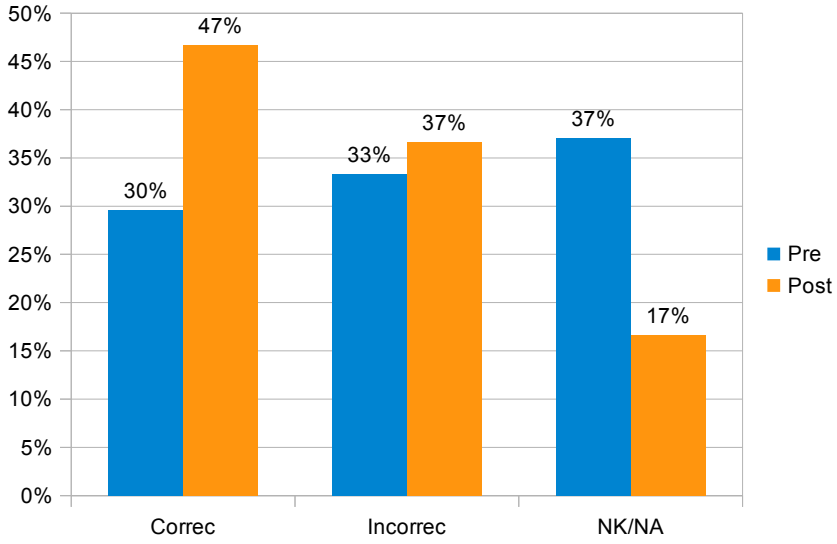

Figure 9: Control group: what do you mean by simulation?

The post-test observed a reduction in uncertainty regarding these concepts, from 37.04% to 16.67% and from 40.74% to 6.67%, respectively. 77.78%, in the pre-test, answered affirmatively about whether the use and/or construction of models and/or simulations could contribute to their school learning, and in the post-test 76.67% did só (figure 9).

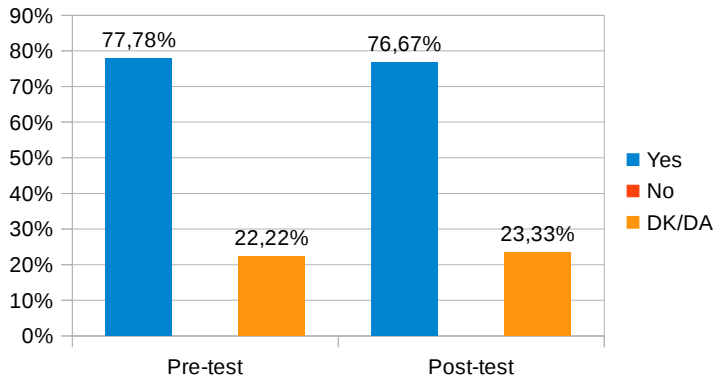

Figure 10: Control group: can the use and/or construction of models and/or simulations contribute to your school learning?

In the evaluation of kinematics contents, in the pre-test, the average of correct answers was 59.26%, while in the post test, it was 57.06% (figure 10).

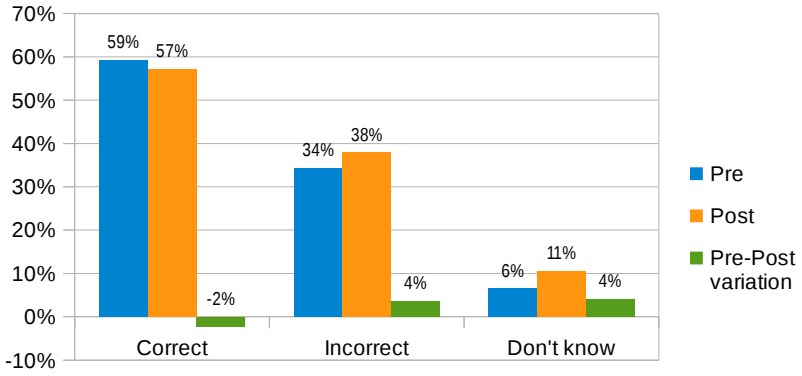

Figure 11: Control group: evaluation of kinematics contents

Experimental group

The experimental group included of 79% of females students and 21% of males students, of which 54.17% in the pre-test reported using software at school to represent situations that are usually done with pencil and paper, and 37.5% to solve problems. 66.67% said they use computer technology to solve problems on a daily basis and 70.83% prefer to solve problems on an individual basis.

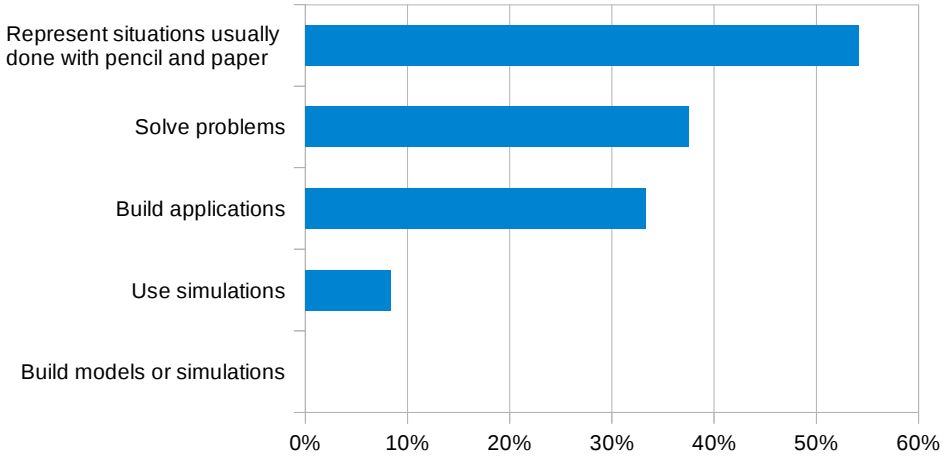

Figure 12: Experimental group: students that had used software at school to...

As for the question on the model concept, 25% answered correctly in the pre-test, while 36.36% in the post-test. Regarding the simulation concept, the correct answers in the pre-test were 33.33%, with 59.59% in the post-test (figure 15).

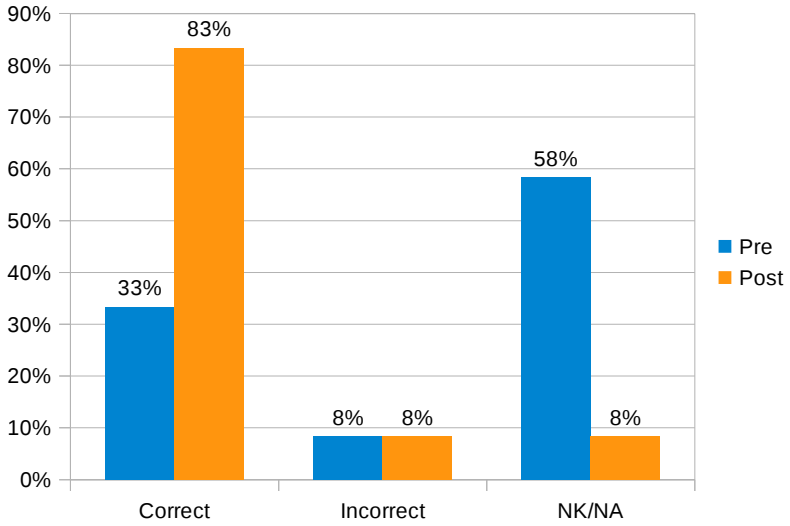

Figure 13: Experimental group: what do you mean by simulation?

The post-test observed a reduction in uncertainty regarding these concepts, from 70.83% to 33.33% and from 58.33% to 8.33%, respectively (figure 12). In the pre-test evaluation, about whether the use and/or construction of models and/or simulations could contribute to their school learning, 54.17 answered affirmatively and reaching 72.73% of these responses in the post-test (figure 13).

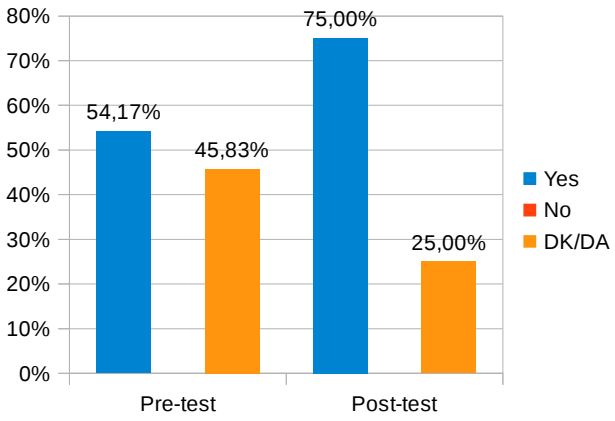

Figure 14: Experimental group: can the use and/or construction of models and/or simulations contribute to your school learning?

In the evaluation of Kinematics contents, in the pre-test, the average of correct answers was 56.25%, while in the post test, it was 60.42% (figure 14).

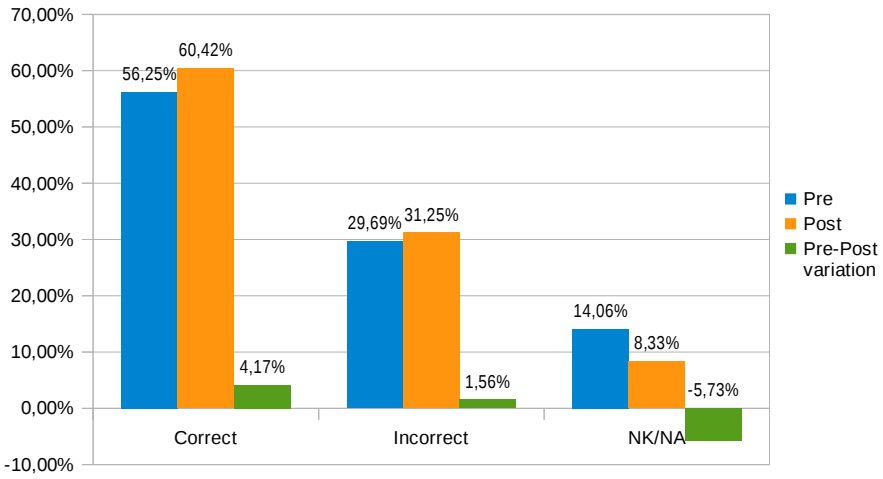

Figure 15: Experimental group: evaluation of kinematics contents

5. Discussion

The results of fieldwork indicate that the construction of a computer simulation using the Squeak-Etoys programming language favoured the learning of school contents related to kinematics, to a greater extent than developing classes with usual resources.

Comparison of the results of the pre- and post-test evaluations of kinematics from the control group shows a decrease in ratings, while the results of the evaluations of the experimental group in the post-test are superior to those of the pre-test.

These results coincide with those mentioned by Taub and others (2015), who found that programming simulations of physical phenomena was an important factor in learning physics, both in the visual representation of physical phenomena and in

structural knowledge, procedural knowledge and systemic knowledge.

This research, however, lasted three years during which weekly classes took place and JAVA was used using a friendly environment, while the experimental work of this thesis consisted of only three classes, and the language used is totally mediated by an interface specially designed for non-experts where the orders are words of the natural language and are translated into the native language, a factor that favored the use of the application and a rapid learning of it.

In this work, the participating students belonged to different schools and were selected for being especially talented students, while in the case of the students participating in the fieldwork of this thesis, all of them belonged to a complete course, without any kind of selection, and therefore, the group was heterogeneous in terms of academic performance, which may have affected the grades, at the same time, that the resource proposed in the activity can be considered valid.

Unlike the experience of Queiruga (2013) who worked with students from technical schools, and Taub (2015) (a computer science course), the experimental work of this thesis was carried out with groups of students who had no programming or algorithmic knowledge; moreover, the orientation of their (commercial) career is alien to the specific field of computing, computer science and other technical disciplines. This difference in relation to the previous knowledge of the students participating in the experiences mentioned above, meant a disadvantage for the students of the experimental group that nevertheless obtained favorable results with respect to the contents of the course, as well as the evaluation of the didactic resource used.

In both control and experimental groups, the uncertainty in the post-test evaluation was reduced with respect to questions about the concepts of modeling and simulation, although in all cases it was not directly matched by an increase in the correct answers. However, this reduction indicates a construction of meaning on the part of the students between the pre-test and post-test evaluations.

In the experimental group, the responses "don't know / don't answer" with respect to the concept of "model" decreased almost 38%, mainly increasing the incorrect responses in the post-test, while the responses "don't know / don't answer" about "simulation" decreased from 50% to 8% while at the same time increased the correct responses by 50%. It should be noted that the term most commonly used in experimental activity, both in tutorial and oral communication, was "simulation".

Papert (1999) assimilates the figure of the apprentice to the teacher who directs an activity in a constructionist key, since he will encounter new and unforeseen situations even for the teacher himself. Situations in which he will have to learn while the pupil, participating in the activity, will in turn learn from his teacher. During the experimental classes unforeseen situations occurred, probably due to the fact that the interactive tutorial remained open to modifications (as well as the environment itself) because it was its primary property. However, these same emerging situations made each group's experience unique, reflecting consequences of their own work and allowing for solutions that were both unique and challenging for both students and teachers.

He also acknowledges that his work will be harder but much more creative and interesting. The contingencies were frequent, and made the teacher's work harder by constantly responding to students' requests for help, and also creative as each situation was different in each case.

The previous work to the activity with the students, especially the development of the interactive tutorial, required many hours of work where each slide was part of a strategy which use could be known only by testing it, and the experience met expectations.

The work time required to develop the tutorial included tasks that were built for the first time:

- choice of a topic whose learning is enriched by the type of work used in fieldwork,
- re-create (teacher) the simulation and simplify it from the point of view of the Etoys code so that it was as accessible to students as possible,
- deconstruct that simulation to create the slide show or tutorial page sequence and its specific content; this involves writing the tutorial script, graphic objects, and code components, and *d.* create a graphical layout common to all slides for easy navigation

However, the material resulting from these tasks can be reused as a starting point for new tutorials, using the design criteria indicated in the last point as a template, and also recovering not only the content but also the methodology followed for the accomplishment of the first three mentioned tasks. Astudillo (2016) highlights the importance of reuse as a characteristic inherent to Learning Objects and desirable in Digital Educational Materials in general. On the other hand, the idea of reusing or recycling material is central to the programming and was weighed both by Papert and Resnick, while at the same time being one of the characteristics of computational thinking (on another scale, no operating system is completely rewritten in a new version or distribution, but the previous one is modified).

The presentation of the interactive tutorial to the students included a recommendation that they take the activity as a game. The importance of the playfulness of a didactic activity is highlighted by Queiruga (2014), insofar as it involves interaction with peers, fun, tolerance for frustration and motivation towards achievement. Also, in a review of 27 works on teaching and learning computer thinking through programming, Lye and Koh (2014), citing Jonassen, Kaffai and Resnick, found better results when the problems they propose to students are of interest, making them more committed to the activity, and suggesting working around issues such as game design, game strategies and digital story making. Notwithstanding the differences between the objective of this review and that of this thesis, the suggestions of Lye and Koh (2014) make it necessary, when including the construction of simulations, to reformulate conventional activities in order to achieve significance.

One of the features of computational thinking[Wing 2010] is the possibility of generalizing and transferring procedures to different fields of Computation and Systems, which in this experience has been successfully carried out through the programming of a simulation, coinciding with the findings of positive results in most of

the 27 experiences analyzed by Lye and Koh (2014).

Knowledge of the user interface initially took up little exclusive learning time; in a few minutes, working groups were focused on solving the first challenges. In the project "Teaching Programming in Secondary School", Queiruga(2013) points out a change of language: JAVA for RITA (a more user-friendly environment, in the style of Scratch and Squeak-Etoys), because it considers the user interface more appropriate for the age range of the students. This adaptation of the interface, in line with the above-mentioned work, reinforces the viability of this type of interfaces in the introduction of programming as a learning resource in non-league contexts, such as education in general, even in the initiation of specific careers in systems or computing.

During the activity, motivation was observed and in dialogue with the students, they expressed interest in this way of working. At the same time, after the fieldwork, the number of students increased by 34% who indicated that using and building simulations could contribute to their learning. These results coincide with those obtained by Yohan, Kongju and Yunebae (2016), which mention improvements in the perception of computational thinking and students' engagement and interest in activities.

In addition, the group of students who participated in the experimental work co on the possibility consulted f creating simulations related to other contents and other curricular spaces, which indicates motivation and interest in the methodology.

These concerns expressed by students, when valuing learning activities that emphasize the protagonism of the learner, are encouraging and remind Papert (1999) when he points out that education was too focused on its informative facet and digital technologies could correct the imbalance and strengthen the constructive facet, while highlighting the ability of computers to create simulations as an educational resource.

In addition to trying to solve the proposed challenges, some students expressed curiosity about how to modify various aspects of the simulation beyond those proposed, such as the colors, shapes and size of objects (the cars), and whether it would be possible to simulate other types of movement. Questions about making these modifications seem to indicate interest in putting their personal mark on the task, in expressing yourself, beyond what the guidelines request, in short, appropriation of the activity. Coincidentally, the expression was one of the first findings in Brennan and Resnick's research (2012).

6. Conclusions and future work

The construction of a simulation by Squeak-Etoys favoured the learning of Kinematics contents, evidenced in the increase in the qualifications of the post-test evaluations. The foregoing coincides with other similar investigations with axis in Physics contents [Taub 2015].

The results of the experimental activity were satisfactory; for example, after carrying out the experimental activity, the students who understood the concept of simulation increased almost three times. Even for a heterogeneous group, unlike the previous case [Taub 2015] in which the students were selected based on capacity. Also, with that work, there is a diference regarding the duration and number of classes

between the two investigations, as well as the programming language used.

In related works, is pointed out that in the first steps when programming give better results using friendly graphical interfaces similar to Scratch and Squeak-Etoys, the application used in the field work described in this paper, and with good results, in accordance with same kind articles.

Programming the simulation was shown to be a valid teaching resource even for students without programming or algorithmia knowledge; and proposing the experience with playful characteristics, challenges, meaningful contents for the students and mediated by new technology, favored the learning and interest of the students in the media used. Also, building the simulation of the kinematics exercise generated motivation in the students, as well as positive expectations regarding using the same way of working in other learning curricular topics. The students, in addition to assessing the type of activity they did, were curious about the ways to express themselves that the programming environment brings and put their personal stamp on the activity using the tools of that authoring environment.

The work around the interactive tutorial resulted in a series of reusable resources in future activities, and feedback to improve them, reducing the time of new developments.

May be, as innovative experience, the activity presented numerous contingencies and challenges for the teacher, which meant greater intensity in the task while promoting flexibility in the asymmetry between teaching and student roles, as they tackled these challenges together and worked enthusiastically.

May be, because was an innovative experience, the activity presented numerous contingencies and challenges for the teacher, which meant greater intensity in the task while promoting flexibility in the asymmetry between teacher and student roles, as they tackled these challenges together and worked enthusiastically.

Regarding future work, the following possible lines of action are considered:

1. exploring this way of working in other disciplines
2. the construction of sequences composed of more interrelated activities
3. deepen the analysis of results from the point of view of student motivation and expression
4. include programming as another means of representation
5. recording the tasks common to several experiences in order to produce a methodological guide

References

Astudillo, Javier, “Estrategias de diseño y ensamblaje de Objetos de Aprendizaje”, M.S. thesis, Universidad Nacional de La Plata, 2016

Churches, Andrew, “Taxonomía de Bloom para la era digital”, http://www.eduteka.org/TaxonomiaBloomDigital.php, traducido de Churches, Andrew “Bloom's Digital Taxonomy”, 2009 http://edorigami.wikispaces.com. Accessed on 11/06/2015

Fundación Sadosky, “CC-2016 - Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas”, pg. 23, 2013, http://www.fundacionsadosky.org.ar/wp-content/uploads/2014/06/cc-2016.pdf

Gordon, Geoffrey, “Simulación de sistemas”, Ed. Diana, México D.F., 1° Ed. 1980, 6° reimpresión 1991

Kafai, Y., Burke, Q., “Connected Code – Why Children Need to Learn Programming”, The MIT Press, 2014

Kay, Alan, "Children Learning By Doing – Etoys on the OLPC XO", Viewpoints Research Institute, 2007, http://wiki.laptop.org/images/2/28/OLPCEtoys.pdf. Accessed on 16/12/2016

Kelton W., Sadowski R., Sturrock D., “Simulación con Software Arena”, 4° edición Ed. McGraw-Hill Inc., España, 2008

Lye, S.Y., Koh, J.H.L., "Review on teaching and learning of computational thinking through programming: What is next for K-12? ", *Computers in Human Behavior*, Dec2014, Vol. 41, p51-61. 11p. URL: http://www.sciencedirect.com/science/article/pii/S0747563214004634. Accessed on 20/02/2017

Papert, Seymour, "Logo Philosophy and Implementation", Logo Computer Systems Inc. Introduction, 1999, http://www.microworlds.com/company/philosophy.pdf. Accessed on 15/11/2015

Queiruga, C., Fava, L., “Enseñar a Programar en la Escuela Secundaria. Experiencias del proyecto Java en Escuelas Técnicas”, Memorias del 1ª Congreso de Extensión de la Asociación de Universidades Grupo Montevideo -AUGM-Extenso 2013, Ed: Universidad de la República, ISBN 978-9974-0-1038. URL: http://www.linti.unlp.edu.ar/uploads/docs/ensenar_a_programar_en_la_escuela_secundaria_experiencias_del_proyecto_java_en_escuelas_tecnicas.pdf. Accessed on 21/02/2017

Queiruga, C., Fava, L., Gómez, S., Kimura, I., Brown Bartneche, M., “El juego como estrategia didáctica para acercar la programación a la escuela secundaria”, XVI Workshop de Investigadores en Ciencias de la Computación 2014. http://jets.linti.unlp.edu.ar/uploads/docs/wicc_2014.pdf. Accessed on 11/06/2015

Resnick, M., Brennan, K., “New frameworks for studying and assessing the development of computational thinking”, 2012, http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf Accessed on 30/11/2016

Resnick, Mitchel, “Reviving Papert's Dream”, *Educational-technology – the magazine for managers of change in education*, Volume 52, Number 4, JulyAugust 2012, http://web.media.mit.edu/~mres/papers/educational-technology-2012.pdf. Accessed on 29/08/2015

Sánchez Gómez,M., “Buenas Prácticas en la Creación de Serious Games (Objetos de Aprendizaje Reutilizables)”, Universidad de Málaga. 2007, Disponible en: http://ceur-ws.org/Vol-318/Sanchez.pdf. Accessed on 14 de julio de 2016

Taub, Armoni, Bagno, Ben-Ari, "The effect of computer science on physics learning in a computational science environment", *Computers & Education* Volume 87, September 2015, Pages 10–23, URL: http://www.sciencedirect.com/science/article/pii/S0360131515000913. Accessed on 21/02/2017

Wing, Jeannette M., “Computational Thinking: What and Why?, Scholl of Computer Science”, Carnegie Mellon University, 2010, http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf. Accessed on 29/11/2016]

Yohan Hwang, Kongju Mun, Yunebae Park, "Study of Perception on Programming and Computational Thinking and Attitude toward Science Learning of High School Students through Software Inquiry Activity: Focus on using Scratch and physical computing materials", *Journal of the Korean Association for Science Education*, 2016, URL: http://koreascience.or.kr/article/ArticleFullRecord.jsp?cn=GHGOBX_2016_v36n2_325. Accessed on 21/02/2017