This is a post-print version of the following article: López, V., Hernández, M.I. (2015). Scratch as a computational modelling tool for teaching physics. *Physics Education*, 50 (3), 310-316.

DOI: 10.1088/0031-9120/50/3/310
https://doi.org/10.1088/0031-9120/50/3/310

Scratch as a computational modelling tool for teaching Physics

Abstract:

The Scratch online authoring tool, which features a simple programming language that has been adapted to primary and secondary students, is being used more and more in schools, as it offers students and teachers the opportunity to use a tool to build scientific models and evaluate their behaviour, just as can be done with computational modelling programs. In this article, we briefly discuss why Scratch could be a useful tool for computational modelling in the primary or secondary physics classroom, and we present practical examples of how it can be used to build a model.

Keywords: Computational modelling, ICT, Scratch, programming

MODELLING AS A SCIENCE LEARNING PROCESS

Although the word 'model' is polysemic, its most common meaning is that of a representation of an idea, object, event, process or system that is created for a specific objective (Gilbert & Boulter, 1998). The basic characteristics of a model are:

- It does not represent the complete reality of the system being modelled, but only the aspects that are of interest to its user, and therefore, as they are human creations, they do not exist in the physical world.
- The same reality can be represented by multiple models, depending on the interests of whoever is using the model.

Gutiérrez (2005) defines the term 'scientific model' as a representation of a system, made up of a set of objects with certain properties or variables, and a set of laws that declare the behaviour or functions of these objects or the relationship between their variables. The essential functions of a scientific model are to explain and predict. At the same time, scientific models are considered to be representations that enable scientists to reason, and which are used to simplify or idealise complex phenomena, to help visualise abstract ideas, to support interpretations of experimental results and to help produce explanations and predictions (Justi & Gilbert, 2002).

Despite the importance of building models in the development of scientific knowledge, many authors agree that there are no general rules for doing so. This non-linear process of constructing, evaluating and using models is what is known as 'scientific modelling'. As Hodson (1992) said: "*the ways in which scientists work are not fixed and not entirely predictable, and involve a component that is experience-dependent in a very personal sense. These ways of working are not teachable. So, the only effective way to learn to do science is by doing science*". In accordance with this viewpoint, there is a need to promote among students a more realistic image of science, trying not to portray the existence of a single scientific method but instead encouraging creativity as one of the essential skills used in the construction of scientific conceptual models at school.

Many authors have highlighted not only that conceptual models are powerful for helping students to learn science, but also how doing activities of construction and use of conceptual models in the science classroom can help students to learn about science (i.e. about the nature of models and their role in the development and communication of the results of scientific inquiry) and to do science (i.e. to create, express and evaluate their own models).

The educational approaches or proposals that foster the teaching of science through the construction, evaluation and use of models do not imply that students should think like scientists but rather that this way of working in the science classroom could be closer to, and more realistic in comparison with, the way in which science is constructed in the scientific practice of modelling.

The fostering of modelling processes in the science classroom seems to be a good way of promoting an understanding among students of certain scientific phenomena and systems, by promoting the construction, testing, review and application of conceptual models. In short, the aim of modelling in the classroom is to create an environment in which students become better thinkers (Feurzeig & Roberts, 1999).

COMPUTATIONAL MODELLING IN THE SCIENCE CLASSROOM

In the same way that professional science uses computer systems that can build computational models to simulate behaviours from the physical world, in education different applications have also been developed that can enable students to express their own mental models. In most cases, these applications use much simpler languages than those of professional science in order to encourage their use by younger students, but maintaining the basic character of modelling activity: to create, express and evaluate one's own models. We are referring, for example, to **VnR** software (acronym for "Variables and Relationships"), which presents an iconic language based on the direct or inverse proportional relationship between variables and between each variable's rate of change (Lawrence, 2004). A similar system using relationships between lineal and exponential variables is that used by **Model-it** (Jackson, Krajcik & Soloway, 2000). In parallel, **Modellus** (Teodoro & Neves, 2011) is designed for more advanced physics students (high school and university), as it requires a certain command of algebraic language. Finally, **Stella** language based on flow diagrams, despite not being exclusively designed for didactic purposes (Hilger & Lusiana, 2014), has also achieved interesting results in education. The common denominator of all these programs is that they enable students to create models and then run them, to thus visualise their behaviour and compare them with the real system that they seek to represent. It is this feature what distinguishes computational modelling tools from educational animations and simulations, where students can visualise virtual phenomena and modify the values of the variables involved, but cannot define or modify the relationship between these variables.

Studies that have analysed the impact of modelling programs on students' learning have found positive results (Mellar et al., 1994; Lawrence, 2002). Unlike simulations, they do not correspond to any specific content and are therefore appropriate for modelling a wider range of phenomena. Moreover, and also unlike simulations, modelling programs require users to specify their model of a certain phenomenon. Therefore, these programs are suitable for encouraging individual exploration and generation of explanations, as the students can express their own ideas, specifying which variables intervene in a phenomenon and investigating how these variables could be related. Modelling programs help students express their mental models and can therefore help develop these mental models into scientific conceptual models when discussed in class with classmates and/or teachers. In addition, these modelling programs require their users to take a qualitative approach to any phenomenon during the development of the model, just as experts tend to do when problem-solving as opposed to novices (Hsu et al., 2006). The adoption of a qualitative approach might help students to focus on the nature of the relationships between the components of the model.

Despite the educational potential of these tools, their use in education is not extensive or widespread, and in some cases they have not caught on at all. This is in contrast to the use in education of scientific animations and simulations, which have been widely accepted in many schools (Hennessy, Deanes & Ruthven, 2006; Rutten, van Joolingen & van der Veen, 2012). Many factors have influenced the unsuccessful application of computational modelling tools in schools, including how difficult it is for teachers to include them in their teaching practice, the fact that before students can start using them they need to spend a substantial amount of time learning their language, the high cognitive demands required in order to use them to build models and the technical difficulty of installing them in computers (Hernández, Pintó & Couso, 2007).

To conclude, it is generally considered that computational modelling offers many opportunities as an activity to teach and learn physics in particular and science in general. However, its use in education has not caught on substantively. It is within this context that we ask whether other computer tools that have not strictly originated from the field of science education, but that have been better accepted at schools, could also serve as computational modelling tools for the teaching of Physics. In fact, this question has already been tackled with other more generalist programs, such as Excel spreadsheets (Lingard, 2003) and Power Point (Duarte, 2012). In our case, we propose the use of Scratch for this didactic purpose.

SCRATCH: CAN IT BE USED AS A COMPUTATIONAL MODELLING TOOL?

Scratch is a programming language developed by the MIT (Massachusetts Institute of Technology) that has been learned by thousands of people in recent years and is being used increasingly more at schools (Monroy-Hernández & Resnick, 2008; Maloney, Resnick, Rusk, Silverman & Eastmond, 2010), especially since the release of the new 2.0 version that can be run online. Following on from the LOGO programming language (Papert, 2005) and promoted using the tagline "Imagine, Program, Share", Scratch offers a wide variety of creations (animations and narratives, presentations, interactive images, simulations, games, etc.) and has generated a rich and dynamic community of users around it, ranging from young children to adults all around the world (Brennan, 2013). Unlike sophisticated programming languages, Scratch is based on the building of piece-based programming blocks, rather like a simple jigsaw puzzle. Its programming interface consists of three different windows (Figure 1): the visualization window that shows the 'backdrop' and the 'sprites' over which we define a behaviour, the window for editing the backdrop and sprites (which can be graphically edited), and the programming window (where pieces of code can be added to build blocks).

Whenever users want to program with Scratch, they must open and save a document called 'Project'. For each project, the required sprites should be defined, and a small program is built to connect the programming pieces sequentially, creating one or several programming blocks. These pieces allow for a wide variety of options, which include conditional functions ('if', 'only if', 'while', etc.), and also logic functions and mathematical operators ('and', 'or', '+', etc.). The system also permits definition of the 'inputs' that make each program run ('when key pressed', 'when this sprite clicked', etc.), create and modify all kinds of variables, make changes to the position and speed of the sprites or modify their appearance.

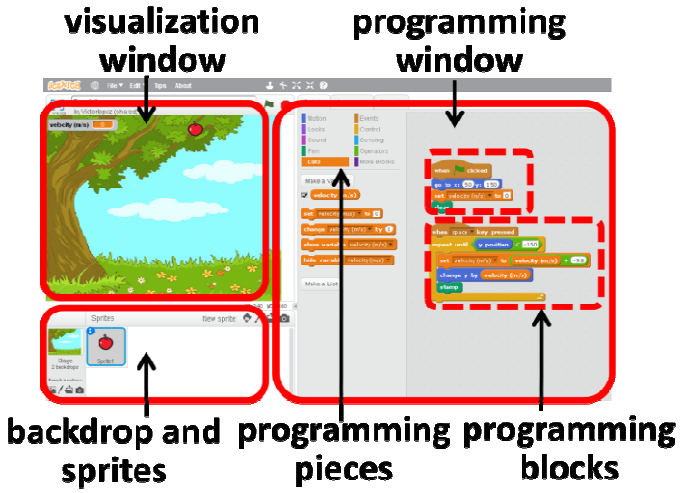

Figure 1. Scratch interface

Using these options, any user can create their own projects and then save them for sharing with other Scratch users, as with a social network. They can also modify these projects using code from other programs (a process called 'remix') or access the code via the 'seen inside' option.

AN EXAMPLE OF A COMPUTATIONAL MODEL: FREE FALL

Figure 2. Backdrop and sprite (apple)

To illustrate how a scientific computational model is built using Scratch we describe the design of a small project called "Free Fall" [1]. Figure 2 shows the display that appears in the project's visualization window, which consists of a single sprite (in this case, an apple). When the spacebar is pressed, this apple reproduces free fall motion, which is described by Rectilinear Uniformly Accelerated Motion equations, with an acceleration of -9.8m/s2. It is therefore a model that describes the motion of an apple and can be used to predict where it will be at any given time and at what speed it will be travelling.

How can students build a model to describe the motion of a free-falling apple?

Building a model with Scratch requires, first of all, the expression of mental models and explanation of the variables, relationships and conditions implied. Thus, students must identify which variables are involved in the model (in this case, position, velocity and acceleration), and must also assign initial values to these variables (the initial conditions). After that, they must establish the set of relationships between variables. In this case, they must be expressed on the basis of the idea that:

- velocity determines the rate of variation in position per unit of time,
- acceleration determines the rate of variation in velocity per unit of time,
- acceleration has a constant value.

To translate these variables and relationships into Scratch language and build a computational model, it is enough to build two programming blocks like those in Figure 3, which a secondary school student might be able to do with the right support and after being familiar with the program's language. First of all, they should define the initial conditions for the sprite (apple) each time that the program is restarted, which in the Scratch language is usually done by pressing on the small green flag in the top right hand corner of Figure 2. So, in the first programming blocks in Figure 3, when we press the flag:

- We define the initial velocity v0=0.
- We define the initial position of the apple using the x and y coordinates defined in the visualization window.

Next, in the second block in figure 3 we define the program's behaviour, which will be activated when the spacebar is pressed. To do so, we need to define a 'repeat until'-type iterative process with two instructions:

[1] It can be accessed online at http://scratch.mit.edu/projects/15060411/

- In each unit of time, increase velocity by -9.8 units, in accordance with the idea that vf=vi–a·t
- In each unit of time, increase the position of the value of velocity at this moment, in accordance with yf=yi–v·t

Once the students have built their model, they can introduce new programming 'pieces', in order to review the model or make it more elaborate. For example, they could record the position of the apple at each moment, adding the 'stamp' piece to the iterative cycle. Likewise, the 'clear' piece should be added to delete the stamped images of the apple whenever the program is restarted (Figure 4). By adding these two pieces to the programming blocks, the apple will fall in accordance with free fall motion, and a display will appear on the screen when the program is run (Figure 5).

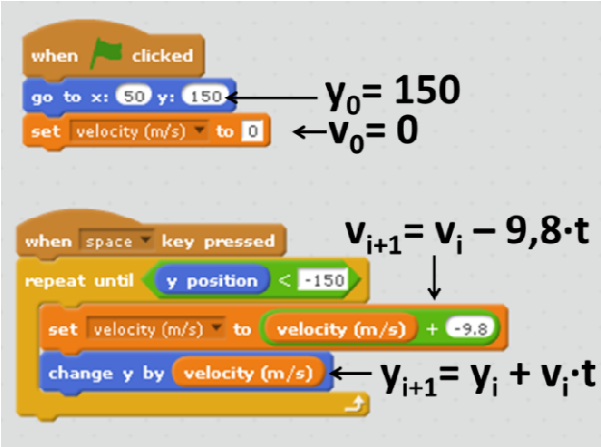

Figure 3. Program determining the motion of the apple sprite.

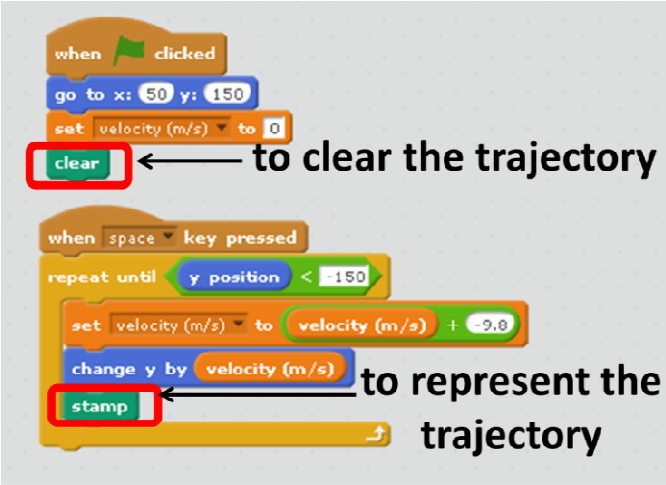

Figure 4. Program to represent the trajectory of the apple sprite.

From here, students can also make the model more complex by adding air friction, or by including a horizontal force (for example, wind), etc. Moreover, once the free fall model has been built, the same students could try building new models by 'reinventing' the first one. They could, for example, build a computational model that describes parabolic motion.

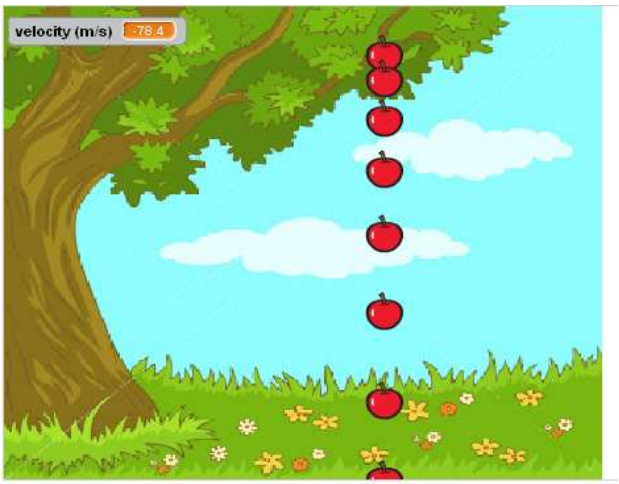

Figure 5. Set of representations of the apple shown using the 'stamp' code. For each unit of time, the distance covered by the apple is greater, because it is moving at a greater velocity.

AN EXAMPLE OF A COMPUTATIONAL MODEL WITH FEEDBACK LOOP: SIMPLE HARMONIC MOTION

Scratch's potential for building scientific models also allows, like many of the computational modelling programs presented earlier, relating variables in the form of a feedback loop, i.e. the relationship between variables is chained in a circular manner, so that periodical or stationary behaviours can be reproduced. Students might be asked to build a model to represent the behaviour of an object based on simple harmonic motion, which can be exemplified using a small program[2] that describes the motion of a buoy bobbing up and down as it floats on the sea.

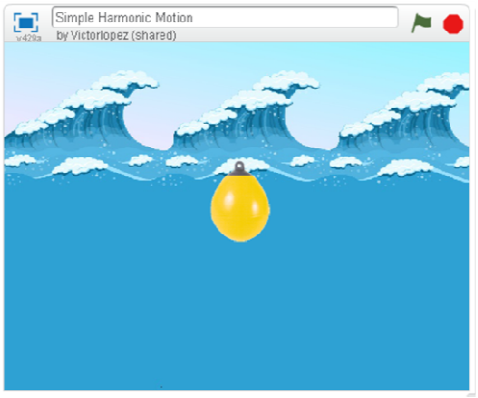

Figure 6. Representation of a buoy floating on the sea.

In this case, to define the behaviour of the buoy, we need to define the following relationships:

- velocity determines the rate of variation in position per unit of time,
- acceleration determines the rate of variation in velocity per unit of time,
- distance with respect to an equilibrium point determines the value of acceleration.

These relationships translated into Scratch language are shown in Figure 7, where it is also shown how, when clicking on the flag, the initial position and velocity values are established, and also that motion begins when the spacebar is pressed, as it was the case in the previous example. However, this time, acceleration varies per unit of time, and will have the value of ai=0 – yi. In this case, the vertical equilibrium position has been considered to be y=0, and the elasticity constant governing motion is considered k=1. However, as occurred in the previous case, the students can also be asked to change these values and introduce small variations to the model that they have built.

[2] It can be accessed online at http://scratch.mit.edu/projects/22451327/

Figure 7. Program determining the movement of the buoy sprite.

Once again, after helping the students to build this system of relationships, it is also possible to add much complexity, turning this simple harmonic motion into damped or forced motion.

CONCLUSIONS

Although Scratch was not solely designed as a program to be used for computational modelling, primary and secondary school students can use its programming language to express their own models, specifying which variables are involved in a phenomenon and, within the possible limitations of the program, research how these variables could be related. So the use of Scratch as a modelling tool not only enables students to express their own models using a specific language, but also to use it to make predictions, to evaluate the results of running the expressed models and to discuss their different models with classmates and/or teachers in a similar way that scientists do. Moreover, unlike other modelling programs, the 'See inside' option allows any Scratch user to examine the 'heart' of any Scratch project that is available online, and thereby access the code that was used to program it.

Given all the opportunities offered by Scratch, the challenge is how physics teachers can take advantage of a digital innovation like this to improve their classes and create environments in which the students' building and evaluation of models is truly encouraged.

Acknowledgements
This article was produced with the support of funding from the MINECO EDU2011-28431 project.

BIBLIOGRAPHY

Brennan, K. (2013). Learning computing through creating and connecting. IEEE Computer, Special Issue: Computing in Education.

Duarte, V. T. (2012). Learning science (and mathematics) with mathematical modelling: can students learn by doing? CBLIS 2012 Keynote Conference.

Feurzeig, W., Roberts, N. (1999). Modeling and Simulation in Science and Mathematics Education. Springer-Verlag New York, Inc.

Gilbert, J., Boulter, C. (1998). Learning Science through Models and Modelling. International Handbook of Science Education. In: B. J. Fraser and K. G. Tobin, Kluwer Academic Publishers. Vol. 1, 53-66.

Gutiérrez, R., Pintó, R. (2005). Teachers' conceptions of scientific model. Results from a preliminary study. ESERA 2005 Conference Proceedings.

Hennessy, S., Deanes, R., Ruthven, K. (2006). Situated Expertise in Integrating Use of Multimedia Simulation into Secondary Science Teaching. International Journal of Science Education. Vol. 28 (7), 701-732.

Hernández, M.I., Pintó, R., Couso, D. (2007). Teachers' perceptions of computer modelling applications. ESERA 2007 Conference Proceedings.

Hodson, D. (1992). In search of a meaningful relationship: an exploration of some issues relating to integration in science and science education. International Journal of Science Education, 14(5), 541-562.

Hsu, Y., Hwang, F., Wu, H., Li-Fen, L., I-Chung, K. (2006). Analysis of Experts' vs. Novices' Modeling. Proceedings: Modelling in Physics and in Physics Education. GIREP 2006. University of Amsterdam.

Jackson, S., Krajcik, J., Soloway, E. (2000). Model-It: A Design Retrospective. In Jacobson, M. and Kozma, R (Eds.), Advanced Designs For The Technologies Of Learning: Innovations in Science and Mathematics Education. Hillsdale, NJ: Erlbaum.

Justi, R. S., Gilbert, J. (2002). Modelling, teachers' views of the nature of modelling, and implications for the education of modellers. International Journal of Science Education, 24(4), 369-387.

Lawrence, I. (2002). A Modelling tool for expressing some thoughts in the Sciences. Educational Technology, Vol. 1. Proceedings ICTE, Formatex. Badajoz, 52-57.

Lingard M 2003 Using spreadsheet modelling to teach about feedback *Phys. Educ.* 38 418–22

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. 2010. The scratch programming language and environment. ACM Trans. Comput. Educ. 10, (4). Article 16.

Mellar, H., Bliss, J., Boohan, R., Ogborn, J., Tompsett, C. (1994). Learning with Artificial Worlds: Computer-Based Modelling in the Curriculum. The Falmer Press, Taylor & Francis, Inc.

Papert, S. (2005). Teaching Children Thinking. Contemporary Issues in Technology and Teacher Education, 5 (3), 353-365.

Duarte, V. T., & Neves, R. G. (2011). Mathematical modelling in science and mathematics education. Computer Physics Communications, 182(1), 8–10.

Thomas Hilger and Betha Lusiana (2014) Modeling Agroforestry Systems to Improve the Lives of Farmers. Keeping System thinkers of the loop. Fall 2014.

Rutten, N., van Joolingen, W. R., & van der Veen, J. T. (2012). The learning effects of computer simulations in science education. *Computers & Education*, *58*, 136–153.