Programming and Robotics with Scratch in Primary Education

J.C. Olabe[1], M. A. Olabe[2], X. Basogain[2], I. Maiz[2], C. Castaño[2]

[1] Christian Brothers University, Electrical Engineering Dpt., 650 East Parkway South, Memphis, Tennessee 38104, USA
[2] Engineering and Education School of Bilbao, Basque Country University, Alameda Urquijo s/n,4013 Bilbao, Spain

The chapter includes a description via examples of the: objectives of integrating programming and robotics in elementary school; the pedagogical infrastructure, including a description of constructionism and computational thinking; the hardware-software support of the projects with Scratch and WeDo; and the academic support to teachers and students with LearnScratch.org. Programming and Robotics are areas of knowledge that have been historically the domain of courses in higher education and more recently in secondary education and professional studies. Today, as a result of technological advances, we have access to graphic platforms of programming, specially designed for younger students, as well as construction kits with simple sensors and actuators that can be programmed from a computer.

Keywords Programming; Educational Robotics; Scratch; Computational Thinking; Constructionism.

1. Introduction

Scratch [1] is a visual programming language that allows children to easily create their own personally meaningful interactive content (interactive stories, games, music and art). In addition, the Scratch website is a forum where children can upload their creations and share them with everyone, encouraging a collaborative group work. The construction set WeDo [2] allows young students the construction and programming of simple LEGO models, which connected to a computer, can implement robotic basic tasks. These projects include a crocodile that closes its jaws when a sheet of chapter is placed nearby, balancing bar that balances automatically, or a horse walking on its own. Over a hundred projects are available to schools and students.

The combined use of Scratch and WeDo in the creation of robotic projects exemplifies the paradigm of constructionism learning [3], where the young student is the protagonist of the learning process, and he or she learns by doing. These two platforms (software and hardware), greatly facilitate the introduction of programming and robotics in primary school. This process makes of these young students authentic natives of the new technology world in which they live. They become experts in the areas of programming, programmable logic controllers, robots, control systems, and techniques of computational thinking.

LearnScratch [4] is a website created in 2007 by the authors of this chapter with the objective of offering the users of Scratch, in particular the teachers of primary schools, a set of didactic resources (video tutorials, lesson plans, Scratch projects, extensions and solutions) that would allow them to integrate the study of Scratch in the classroom in an efficient and pedagogical way. LearnScratch is supported in its mission and activities by the team Scratch-MIT, and participates actively in the dissemination of Scratch in schools around the world.

2. Constructionism and the Use of the Computer

Constructionism states that programming, the act of building a system, and debugging, finding obstacles and problems, and solving them, is the most efficient way for a child to learn. It is also the process where the child will explore meta-thinking, and the closest the child will come to experience how we learn to learn.

The programming language Logo was the environment created to allow young students explore the world of constructionism. This programming language was developed early in the recent history of computing and personal computers, and was limited in its functionalities by the technology of the time.

Almost forty years later, the programming language Scratch was made available to the world (Scratch software is free of charge.) Scratch was inspired by Logo and constructionism. It includes many functionalities, graphic interface, multimedia, elimination of syntax errors, etc. that the current technology makes possible. An additional functionality makes Scratch different from Logo. Scratch was created for social computing. Students would share their designs, learn for each other, and as an essential part of Scratch, the projects, the programs, were designed to be shared, to be open, remix, evolved, and then send back to the community.

Computational thinking [5] is a new learning and teaching paradigm based on two fundamental pillars. The first is that in the process of learning how to solve everyday problems we can benefit greatly by learning how computers solve similar problems. When a child packs her books in the knapsack before going to school, or a mother plans her route to take the children to piano lessons and soccer practice, and stop at the grocery store, they both are trying to solve problems for which humans are not inherently well prepared and for which there are good and efficient solutions.

The second foundation of computational thinking is that a computer language offers a set of primitives, syntax rules, and design patterns with which we can practice again and again, finding obstacles and problems in the process, and solving them, in a way similar to the process of creating a set with Lego blocks. In this case we will use computational blocks. This process of practicing, finding obstacles, problems and errors, and solving them, creates in our minds the structures in which these elegant and efficient solutions become well established. Through this process of progressive learning every child has the ability to achieve a certain degree of mastery, mastery so elusive to current children in school as they try to master simple arithmetic processes.

The ideas behind computational thinking, the development of a set of good practices and educational paradigms are still in their infancy. Its formal integration in the classroom is being made in the form of individual projects and experiments. Currently there are more challenges than proven results.

Before the concepts of computational thinking began surfacing during the last decade, the theory of constructionism was being developed at MIT since the 1970's. This theory is an extension of constructivism, which proposes that learning is not a mere acquisition of knowledge, but the actual creation of structures in our minds. The structures constitute the knowledge. Constructionism extends this idea to incorporate the activity of the child in building physical structures as a way to foster their permanency in the mind. If a child builds with Lego bricks a toy garage, and it builds it again and again, her mind now knows, and knows well, what a garage is, how it operates, its functionality, its limitations.

The project AprendiendoScratch (the Spanish language version of LearnScratch) extends the concept of Computational Thinking to reflect its similarity to the traditional thinking process based on natural language. The natural language based on nouns, verbs, adjectives, etc. provides the bases for the creation and expression of humanistic and qualitative ideas. For the quantitative analysis of human life activities, humans use the tools derived from mathematics. This language differs in essential ways from the natural language, and as a consequence it is difficult to teach it in the school environment and therefore it is rarely acquired by students with the level of mastery required. The current state of knowledge of mathematics in schools around the world is a testimony of this reality. Children create and express ideas within the language of mathematics in a way similar to if they do it in a language such as Latin, of which their knowledge is limited and limiting. However this limitation is not determined by human capacity but rather by the lack of a language in which these ideas can be created and expressed.

The concept of Computational Thinking as an environment for the creation and expression of ideas includes: a) the creation of a language for the representation of this type of ideas; b) the development of an environment for the exploration and learning of the language, in a way similar to how a child learns his maternal language with a high level of knowledge in the first years of life; and c) a set of areas of interest in daily life where it is possible to integrate these ideas in order to solve routine problems (geometry, dynamic, probability, cybernetics, etc.)

3. Integration of Programming in the Curriculum

The ideas of educating students in the world of constructionism and computational thinking have received the support of many educators, administrators and policy makers since their inception. However their implementation has encountered different sets of obstacles that have made their integration in the classroom often impossible or at least difficult and in general unsatisfactory.

A critical factor in the success of this integration is the availability of computers in the classroom. Computers constitute the world where students experiment and discover new ideas, and in the process develop the skills that can only be developed through this process. Computers are increasingly found in greater numbers as part of the academic life of students, not only in programming labs and technology courses but as integral part of other academic subjects such as mathematics or language. In recent years there has been a great effort to provide students with computers as integral part of their education. The foundation One Laptop Per Child carries this idea even farther by promoting the idea that all children, regardless of their economic level, have a personal laptop as the essential learning and exploration tool in their education.

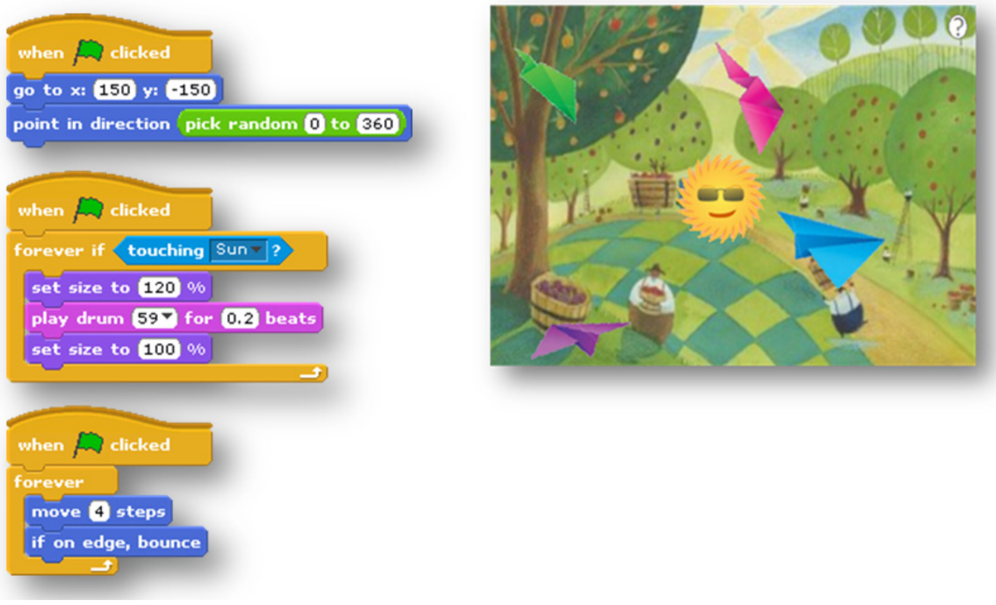
A second limiting factor for the implementation of these strategies has been the identification of the computer as a very specific tool with a particular role in education. The computer belonged in a laboratory and not in the classroom; the computer was useful in teaching programming or word processing but not geography or oral expression; computer classes needed to be taught by computer teachers and not by the classroom teacher. In order to incorporate the computer as an integral element of education it needs to exit the limitations of the laboratory, the domain only of programming teachers, and enter into the regular class, welcome by all teachers in their daily activities.

Finally, the obstacle of creating and expressing ideas with the computer needed to be address and overcome. Environments such as Scratch have made these tasks not only possible but rich and enjoyable experiences for children and teachers all over the world.

3.1 Multimedia and Programming

One of the features that has made Scratch a successful programming environment for the integration of programming in the curriculum in primary school is its ability to allow young programmers the creation of games, videos, and multimedia stories.

Figure 1 shows a simple project in which five moving objects interact on a stage representing a fruit garden. There is an object representing the sun that moves freely from place to place. There are also four paper airplanes that move freely as well. Eventually there will be some collisions when the sun and a plane cross paths. In these events the collision will generate both graphic and sound effects. The airplane will change its size temporarily to mark the moment of the collision, and also it will produce a percussion sound. The combination of the sun of the four airplanes and the sounds they produce in their collisions create a nice effect resembling a small orchestra.



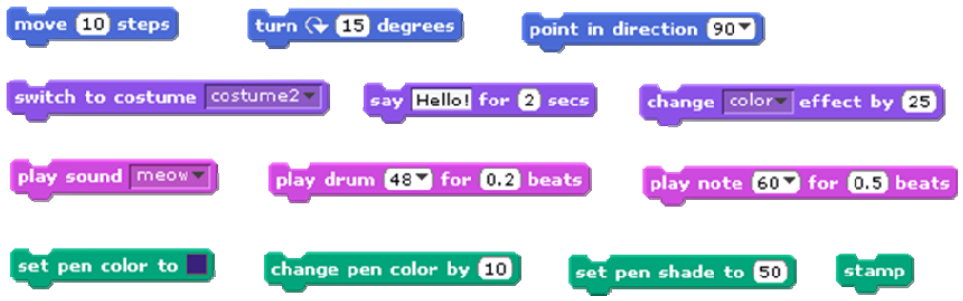
```
when [flag] clicked
go to x: 150 y: -150
point in direction (pick random 0 to 360)

when [flag] clicked
forever if <touching Sun ?>
    set size to 120 %
    play drum 59 for 0.2 beats
    set size to 100 %

when [flag] clicked
forever
    move 4 steps
    if on edge, bounce
```

Fig. 1 A simple multimedia-based application: A flying percussion orchestra.

Scratch dedicates four of its eight block menus to provide tools to manipulate multimedia objects. In Scratch each object, called sprite, has associated with it not only a set of programs or scripts, but also a set of images, called costumes, and a set of sounds. These four block menus allow the efficient manipulation of these resources.



```
move 10 steps        turn ↻ 15 degrees        point in direction 90
switch to costume costume2        say Hello! for 2 secs        change color effect by 25
play sound meow        play drum 48 for 0.2 beats        play note 60 for 0.5 beats
set pen color to        change pen color by 10        set pen shade to 50        stamp
```

Fig. 2 Sample of multimedia blocks: motion, appearance, sound and pen.

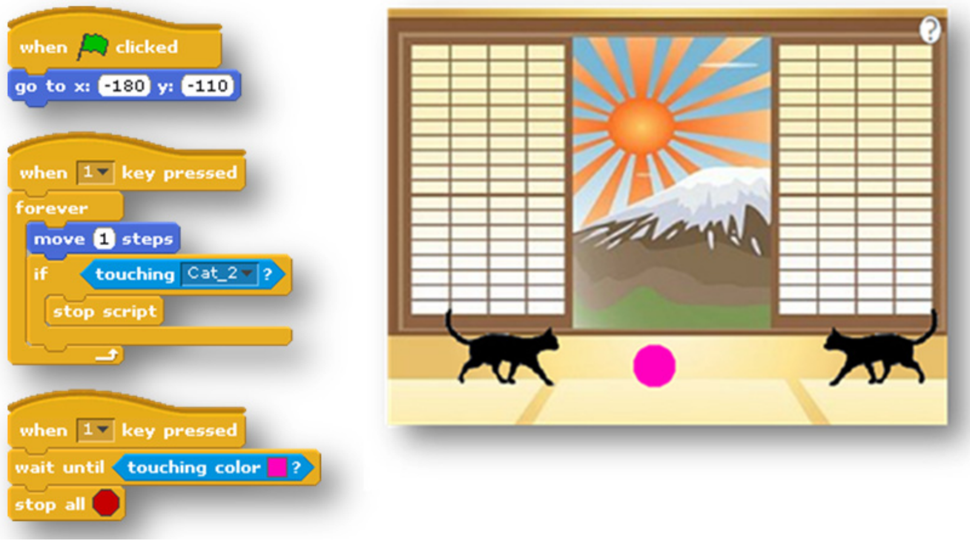
Figure 2 shows a sample group of blocks designed to operate with multimedia. The blue blocks belong to the menu motion and provide resources to move the objects within the stage, for example it makes the airplanes fly in the sky. The lavender blocks are part of the appearance menu and they manipulate the graphical representation of the objects, allowing for example the creation of graphic effects when an airplane collides with the object sun. The magenta blocks manipulate sounds, both recorded or from electronic instruments. The percussion orchestra was easily implemented associating an instrument with each airplane. Finally the green blocks, form the pen menu, allow the creation of images while the program is being executed.

3.2 Programming Language

The programming environment Scratch was designed with a low floor, or the ability to create very simple programs with very little formal training, but also with high ceilings, of the ability to create very sophisticated applications. The purpose to incorporate programming in the curriculum is to develop a language that will support the creation and communication of complex ideas and efficient solutions to daily problems.

Scratch divides the building blocks from which all the programs are created into eight color-coded block menus. Of the eight menus three of them can be identified as having an important role in the creation of the basic structure of a program. The yellow menu, control, includes the blocks that will determine when and how the instructions will be executed. The lime-green menu, operators, provides tools for the implementation of arithmetic, logic and other types of operations. And the orange menu, variables, allows the creation of variables and simple lists.

The project illustrated in Figure 3 includes a simple animation in which we can determine when each of the two black cats start moving towards each other. Later the program will end when one of two events occur: the two cats come into contact with each other, or the cat on the left reaches the magenta circle.



```
when clicked
go to x: -180 y: -110

when 1 key pressed
forever
    move 1 steps
    if touching Cat_2 ?
        stop script

when 1 key pressed
wait until touching color ?
stop all
```

Fig. 3 Project illustrating control blocks to start and stop execution.

Scratch allows parallel processing. Each project may have multiple objects or sprites, and each object may have multiple programs or scripts. The communication between objects and between scripts is via messages. For that reason, in addition to the traditional control structures of a programming language, Scratch provides services to operate with messages. Figure 4 shows a sample of blocks from the control menu.

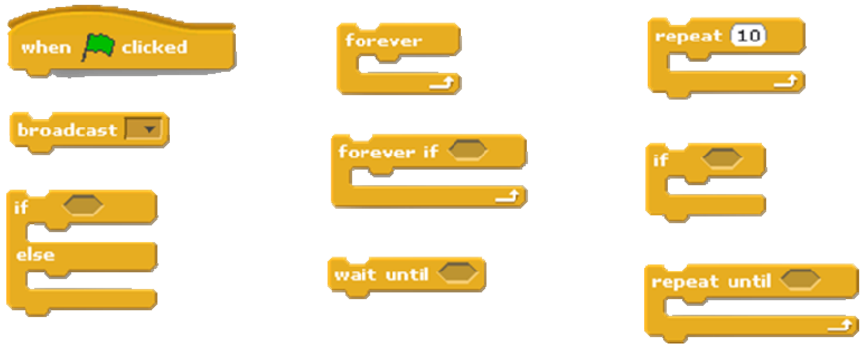


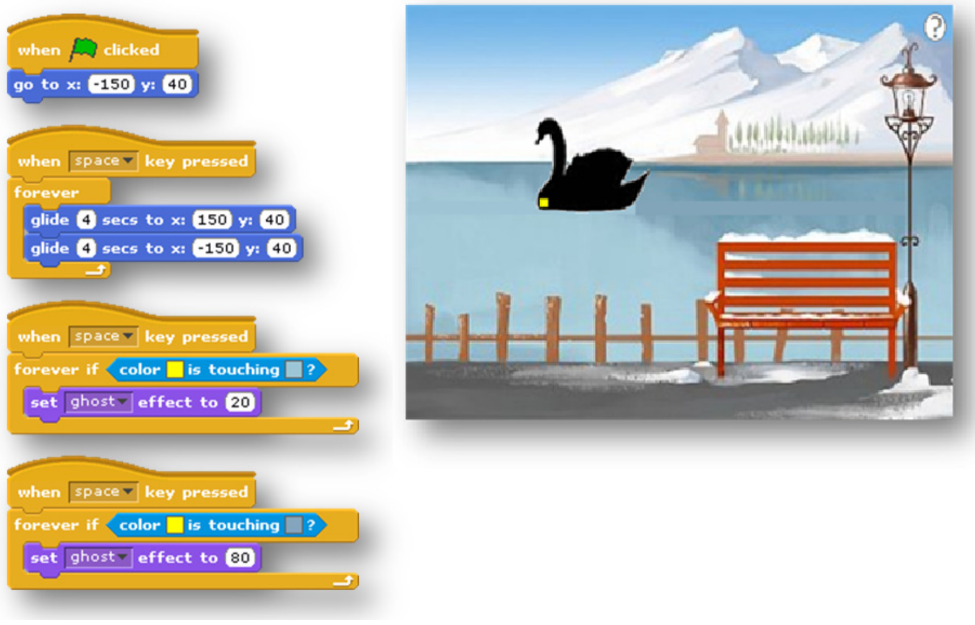
Fig. 4 Sample of control blocks in Scratch.

The default execution of a program is a sequence when each action associated with a block is implemented once, and only once. Scratch provides c-shaped blocks to implement selection structures - if, if/else -, and repetition structures - for, repeat, repeat until, etc. -. A set of blocks with a rounded top implement the event handlers. All scripts must begin with one of these controls.

3.3 Sensors

When implementing projects, is often useful or necessary to have information of the surrounding environment. This will be an essential element when implementing project in the areas of cybernetics and robotics which will be described in the following sections.

Scratch includes a block menu, called sensors, with blocks, many of them reporters, which provide this type of information. The project illustrated in Figure 5 includes a simple example where sensors are used as an integral part of the project. A black swan moves from side to side on the lake. A thin strip of two-tone blue color marks the path of the swan. When the yellow dot of the swan reaches the light color of the strip the swan almost disappears through a vanishing effect. When the swan moves over the darker shade on the strip it regains most of its color.



```
when 🏁 clicked
go to x: -150 y: 40

when space key pressed
forever
  glide 4 secs to x: 150 y: 40
  glide 4 secs to x: -150 y: 40

when space key pressed
forever if color 🟨 is touching ▨ ?
  set ghost effect to 20

when space key pressed
forever if color 🟨 is touching ▨ ?
  set ghost effect to 80
```

Fig. 5 Scratch project illustrating the use of sensors to control a program.

Sensors by definition provide information about the environment. In Scratch a block that provides information, rather than implement an action, can take one of two forms. If the information is numeric, such as the x coordinate of the mouse, or the distance between two objects, the block has a rectangular form with rounded ends. If the information provided is Boolean, true or false, such as is a key pressed? or touching color red?, the block is also rectangular but this time with pointed ends.

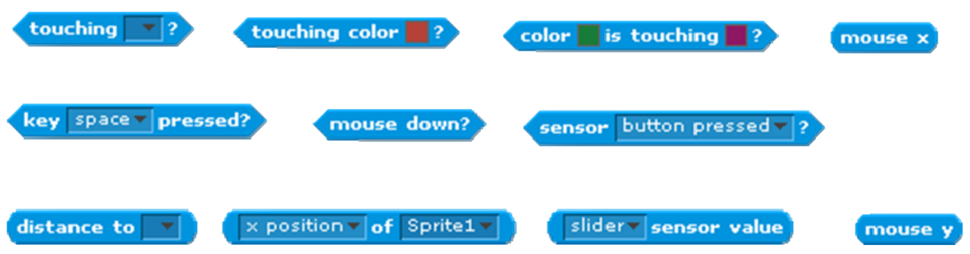


Fig. 6 Sample of sensor blocks in Scratch.

Figure 6 includes a sample of reporter blocks, numeric information, and predicate blocks, Boolean information, from the menu of sensors. These sensors allow the control of a program according to the current conditions of the environment. These conditions include the relative location of objects, the location and state of the mouse, the state of keys in the keyboard, the level of sounds, the relative position with respect to colors, etc.

3.4 Cybernetics

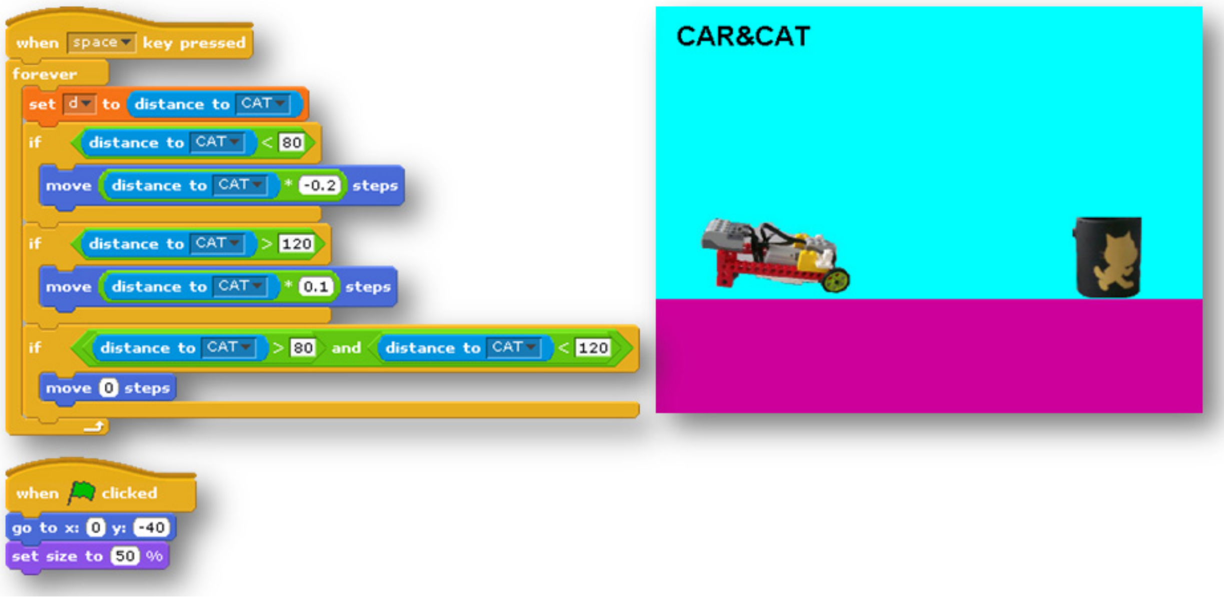
The study of cybernetics in primary school has been proposed as an ideal field of exploration where a young student can experiment with constructions that emulate many aspects of real life. Simple projects such a car that can drive itself in a closed track, or a butterfly that can traverse a maze without external help, have great levels of motivation that can make learning, exploring and discovering a great experience. Cybernetics, such as the examples of the car and the butterfly, is intended to study the structure and functioning of self-regulated systems.

In addition to motivate students by working on projects that reflect life systems, cybernetics has the advantage of introducing students to new areas such as systems theory and control theory. Systems are groups of elements that have been designed to operate in a coordinated mode and to implement a given function. Young students in their daily life use and operate these systems, for example the microwave to heat their breakfast of the heating and cooling system of in their classroom. They know that the thermostat is used to set the desired temperature, and in turn it will activate the heating or cooling system in order to provide desired temperature to the classroom.

This simple example of temperature control helps communicate the basic concept of feedback in control systems. Feedback is the process by which the information of an output signal, for example the room temperature, is sent to the input of the control system. The thermostat will compare that temperature with the desired temperature and it will activate the corresponding actuator: turn on the heating if it is too cold, or turn on the cooling system if the room is too hot.

All control systems that govern the dynamic behavior of a system include three basic components: sensors, actuators, and the control function. The sensors are the devices that report the state of the output signal that needs to be controlled (temperature, distance, etc.) The actuators are the devices that with their action modify the state of the system, for example moving an object, opening a valve, turning on a motor, etc. The control function is the algorithm that determines the type of action that needs to be implemented in order to obtain the correct behavior of the system.

The project Car&Cat has been designed to illustrate the concept of feedback in a control system. In this case the system is implemented by an object with the form of a car for which the desired behavior is to maintain a certain distance of security with respect to a second object, the cat, which is moved randomly by a human hand.



```
when space key pressed
forever
    set d to distance to CAT
    if distance to CAT < 80
        move distance to CAT * -0.2 steps
    if distance to CAT > 120
        move distance to CAT * 0.1 steps
    if distance to CAT > 80 and distance to CAT < 120
        move 0 steps

when clicked
go to x: 0 y: -40
set size to 50 %
```

Fig. 7 Cybernetics: Automatic distance control.

The solution must define the three elements of the control system: the sensor (to measure the distant between objects), the actuator (the motor to move the car in the appropriate direction), and the program to implement the control function.

In this project the sensor, actuator and system (the object car) are internal to Scratch. In the following section of Robotics the sensor and actuator will be implemented with a real system that will move a real car built with Lego pieces.

The program of Figure 7 implements the sensor with the Scratch block distance to, and the actuator of motor with the block move. In the physical implementation of the system we will use the block that reports the distance from the external sensor and the block that controls the external motor of the car.

3.5 Robotics

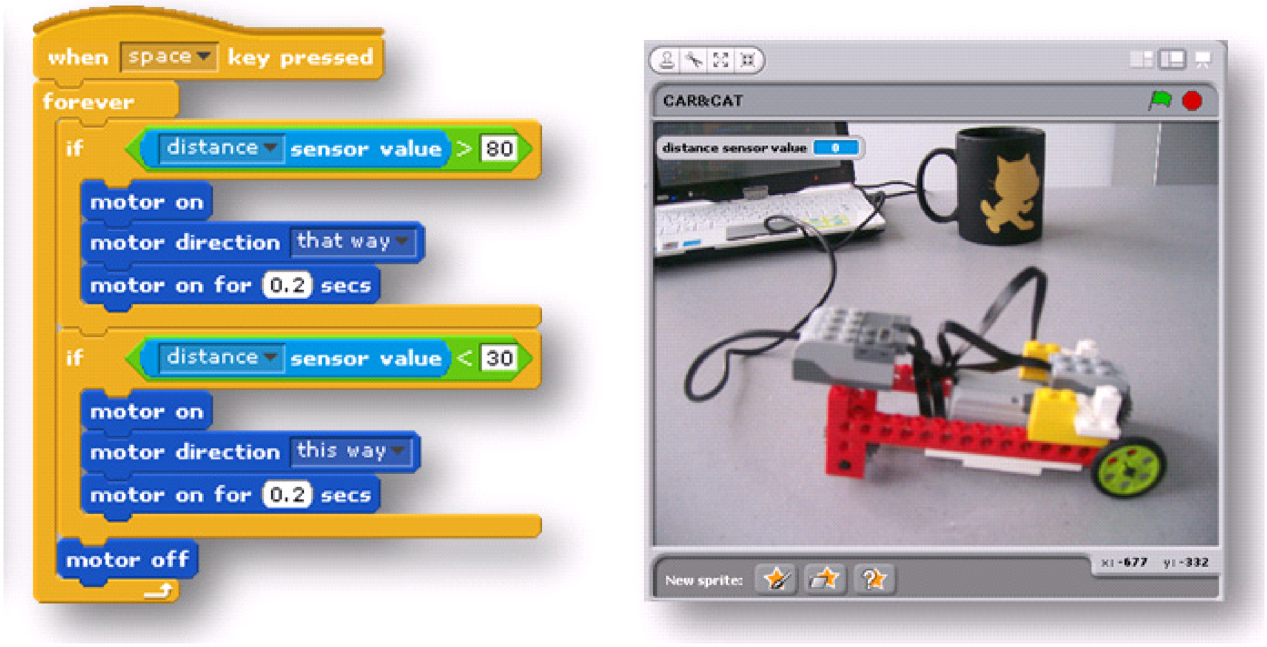
Robotics is the science and technology of robots. Robots are machines or electronic programmable entities able to manipulate objects and implement physical operations until then reserved only to humans. Robotics combines several disciplines, among them control engineering, mechanics, electronics and informatics.

Robotics offers a set of features that are ideal in the development of learning and education for students in primary education. The field of educational robotics provides different sets of pedagogical activities designed to develop multiple competencies of the students. These competencies are developed through the design, construction, programming, debugging and testing of physical systems built with different types of materials and controlled by a computer.

In addition, robotics develops other competencies that are essential in the integral education of children, such as goal setting, team working, problem resolution and communication with other students.

The construction sets of educational robotics include the basic elements of robotic theory: physical elements (multicolor and multiform) to be connected and assembled; physical sensors of magnitudes such as distance, tilt, temperature, and others; electrical actuators such as motors and lights which will implement the physical desired tasks; and the hub connector that will provide the link between the robot and the computer that contains the program controlling the system.

The project Car and Cat is a project of educational robotics with the proposition to create a car that will maintain a certain security distance between itself and a cat (a mug that will be controlled by the hand of the student.) The car is constructed with elements of the Lego Education WeDo toolkit, and controlled by a program created in the environment of Scratch.



```
when space key pressed
forever
  if distance sensor value > 80
    motor on
    motor direction that way
    motor on for 0.2 secs
  if distance sensor value < 30
    motor on
    motor direction this way
    motor on for 0.2 secs
  motor off
```

Fig. 8 Robotics: Implementation of a self-controlled car.

The design and construction of the car is open to all the possibilities of the imagination of the student. The car shown in Figure 8 includes two front wheels connected with two gears to the motor that seats directly behind them. The red frame of the car holds the motor over and a distance sensor that point forward. A third wheel in the back gives stability to the car. The hub that connects the sensor and the motor to the computer seats above the rear wheel.

The programming of the control algorithm is implemented in Scratch and includes the closed feedback loop. A permanent forever block monitors the distance between the car and the cat as it is reported by the distance sensor. If the distance is too large the car moves forward. If the distance is too short, because the cat has moved closer, the car moves back to a safe distance.

The student can experiment with the threshold distances to test different control policies. Additionally the power of the motor can be controlled, and the accuracy of the distance sensor can be explored under different conditions. Finally the student can experiment with different implementations of the structure and mobility of the car.

3.6 Academic support to teachers and students: LearScratch.org

The website Learnscratch.org was designed and created in the summer 2007 by the authors of this chapter soon after the public release of Scratch in the spring of the same year. The key objective in the creation of this site was to fill an apparent void in the overall project of incorporating constructionism and computational thinking in our society.

Constructionism proposes a method by which all children can attain the degree of mastery that is intrinsic to human nature. Therefore constructionism needs to be incorporated into the school system as part of the education of every child in every country.

Scratch is a programming language designed with outstanding features that make it intuitive, motivating and ideal for collaboration. It was designed to have a low floor –easy to make a simple program-, high celling –able to create complex projects-, and wide walls –capable to be applied to many areas of interest-.

It is not difficult to create simple programs in Scratch that validate the premise of low floor. Scratch's ability to easily process images, audio, movement, etc. also illustrates its capacity for wide walls. But Scratch remains still a programming language, and the ability to create projects of certain complexity, high ceilings, it is not acquired by simply tinkering with its elements, or by trial and error.

The LearnScratch site was created with the goal to provide teachers of the world with a roadmap to introduce Scratch into their classrooms oriented for 1:1 model programs. It is based on a set of projects, exercises and teaching materials that incrementally familiarize the young student with a new language and a new way of thinking. In order to master this new language one needs to become familiar with these components. But these elements of language are not sufficient. One needs to master the grammar that gives structure to the projects created with these blocks.

Teachers in primary and secondary schools have great demands on teaching loads, class preparation, and a multitude of class activities. To incorporate Scratch into the classroom presents two important obstacles in the current schools systems. Very few teachers received in their formal education training in programming that could help them learn Scratch at a proficient level. Also, there is a lack of curriculum materials and lesson plans in Scratch that would help teachers plan a coherent schedule. In general these two obstacles do not exist in topics such as mathematics, social studies, or language.

The materials of LeanScratch have been formally adopted by over three thousand schools around the world. This service is provided at no cost to the institutions. These registered schools received the teaching materials, video tutorials, projects, lesson plans, etc. and install them in their computer labs. The same materials are available as well via the internet, which is a practical solution to institutions with a reasonable bandwidth capable to transmit the video tutorials to each student. Many schools, institutions and individuals use daily the academic materials of the site via the internet.

LearnScratch originated as collaborative effort with MIT in order to fill the need of formal education to young Scratch programmers: ("As more and more people start using Scratch, there is a great need for more tutorials and support materials. This site is a great example of how members of the Scratch community can contribute back to the community." Mitchel Resnick, Director of the Lifelong Kindergarten Research Group and the Scratch Team, MIT).

In the process of expanding the community of Scratch and the services provided, LearnScratch closely collaborates with projects such as ScratchEd [6], ScratchDay and the One Laptop Per Child project (OLPC) [7]. The OLPC's goal was to create a computer implementing the ideas of constructionism, learning by doing, by programming and debugging, but a computer that was affordable to those countries still at the margins of the digital divide. The resulting computer is called XO. Eighty five percent of all XO computers distributed around the world are in Spanish speaking Latin America. In Uruguay the totality of the primary and secondary students own their XO, and in Peru, a much larger country, seven million students will benefit from learning with the XO. LearnScratch collaborates with OLPC in Latin America to extend the use of Scratch in the classroom. For that purpose, a new website called AprendiendoScratch (the Spanish language version of LearnScratch) has been launched. In addition to the academic content of the website, the project AprendiendoScratch includes training workshops for teachers (initially in Paraguay, Peru and Mexico) and webinars in OLPC and Latin-American forums. In addition the project includes tasks in the following areas: a) continuing the project in other Latin-American countries; b) development of new academic modules and support textbooks; and c) collaboration with other university research groups in multinational cooperation projects.

Acknowledgements This work was supported in part by the Research Development Grants of the University Basque System (2010-12), Department of Education, Universities and Research – Basque Government. Donostia Kale, 1, 01010 Vitoria - Gasteiz, (Araba), Spain. +34 945 018 000.

References

[1] Scratch-MIT, http://scratch.mit.edu/ Accessed June 12, 2011.
[2] LEGO-Education WeDo, http://www.legoeducation.us/store/detail.aspx?ID=1573&bhcp=1 Accessed June 12, 2011.
[3] Papert S. Mindstorms: children, computers, and powerful ideas. Editorial: New York : Basic Books, 1993.
[4] LearnScratch, http://learnscratch.org/ Accessed June 12, 2011.
[5] Wing J. Computational thinking. Commun. ACM 49, 3(Mar. 2006), 33–35.
[6] ScratchEd. http://scratched.media.mit.edu/ Accessed June 12, 2011.
[7] One Laptop Per Child - OLPC http://one.laptop.org/ Accessed June 12, 2011.