International Journal of Engineering Education Vol. 32, No. 4, pp. 1823–1832, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00

Application of Robot Programming to the Teaching of Object-Oriented Computer Languages*

J. M. RODRÍGUEZ CORRAL, A. MORGADO-ESTÉVEZ, D. MOLINA CABRERA and F. PÉREZ-PEÑA
School of Engineering. Avenida de la Universidad de Cádiz, 10, 11519, Puerto Real (Cádiz), Spain. E-mail: josemaria.rodriguez@uca.es; arturo.morgado@uca.es; fernandoperez.pena@uca.es; daniel.molina@uca.es

C. A. AMAYA RODRÍGUEZ and A. CIVIT BALCELLS
Technical School of Computer Engineering. Avenida Reina Mercedes, s/n, 41012, Seville, Spain.
E-mail: claudio@atc.us.es; civit@atc.us.es

Object-oriented programming (OOP) abstract concepts are often difficult to understand for students, since it is not easy to find the equivalence of such concepts in daily life. In this paper we will study if an interdisciplinary approach based on an introduction to robotics and robot programming helps the student in acquiring the OOP concepts. For our experiments, we selected a sample of thirty individuals among students with an adequate knowledge of procedural programming. This sample was divided into two groups of fifteen students each: for the first one we used a standard introductory approach to C#, whereas for the second one we developed an experimental course that included a demonstration program that illustrated OOP basic concepts using the features of a specific type of commercial ball-shaped robot with sensing, wireless communication and output capabilities. After the courses, both groups were evaluated by completing a multiple-choice exam and a C# programming exercise. Our results show that the student group that attended the course including the robot demo showed a higher interest level (i.e. they felt more motivated) than those students that attended the standard introductory C# course. Furthermore, the students from the experimental group also achieved an overall better mark.

Keywords: interdisciplinary projects; mobile robots; object-oriented programming; robot programming; teaching-learning strategies

1. Introduction

Object-oriented design can be, in principle, very natural, since in real life we usually think in terms of objects, which have certain properties and behaviors. However, it has been shown that when teaching the abstract OOP concepts students often have difficulties, as they find that it is difficult to match these concepts to situations in real life [1].

Also, when writing programs, those students initiated in procedural programming tend to adopt the traditional view of considering a program as a set of instructions and control structures [2]. However, understanding an object-oriented program requires understanding what objects are and how messages are exchanged among them in order to accomplish tasks [3].

In our previous work [4], we tried to give a response to student's difficulties relating to the understanding of OOP concepts using a set of tangible user interfaces (TUIs) which operate as a sensor wireless network [5], as a physical support where such concepts are represented in a visible and tangible way.

In this work, we intend to use a robot—instead of a TUI—as a didactical resource, given its movement capacities and its possibilities as a didactic tool [6] and, specially, in the teaching of sciences and engineering [7–9]. More precisely, we will try to answer—using preliminary quantitative results—if a robot could also be a suitable didactic tool in order to help students to understand better OOP basic concepts. The lack of quantitative research on robot uses in education has been criticized in the past [6, 10].

Nowadays, the use of robots [11] for educative purposes is widely extended: ''Robotics is a true multidisciplinary field that forces us to cross traditional disciplinary boundaries to develop working systems. In addition to the electromechanical systems that endow mobility, most autonomous robots also contain one or more computers and the software and hardware scaffolding necessary to support them'' [12].

''Robotic technology offers an excellent platform providing a hands-on learning environment for reinforcing theoretical topics in Computer Science, Computer and Electrical Engineering and Mathematics'' [13].

The educational approach proposed in this work aims to introduce the students to the fields of robotics and robot programming [13] from an eminently practical point of view, and also to familiarize them with OOP [14, 15], thus leading to a learning experience in an interdisciplinary context. In fact, Robotics and OOP are fields closely related

* Accepted 20 April 2016.

in the practice, since there are currently a number of object-oriented robot programming frameworks [16–19].

In this regard, there are teaching tools based in programming microworlds in order to help students to understand better OOP concepts [20, 21]. *Jeroo*, *Karel J. Robot* and *objectKarel* software tools model microworlds of robots which students can operate sending messages to them. However, from our point of view, it is an incomplete experience, since students do not work with real robots, but with a software which models them as objects.

The use of Sphero robot [22] allows us to explain basic OOP concepts as object, attribute and method, and thus to stimulate significant learning [23] on students. Sphero has mobile capabilities—like any other robot—, as well as internal lighting (a set of color RGB LEDs). It can also be used to play augmented/mixed reality games [24, 25]).

To find out if our approach produces successful results, we will start by developing a demonstration program that illustrates some basic OOP concepts using the features of Sphero (i.e. sensing, lighting and movement). After this, we will develop two introductory OOP courses using the C# language: One of them will include the mentioned software demo, whereas the other will be a traditional C# course. To finish, we will design two tests: a multiple-choice exam for evaluating the acquisition of the OOP concepts, and a programming exercise also based on multiple choices.

Once the two student groups—experimental and control—that participate in the experiment have completed the two exams, we will proceed to discuss the results and to extract the conclusions. These will be shown in section 7.

2. Learning principles

In this section, we discuss two principles that, in our opinion, underlie the learning process of the students of the experimental group: *interdisciplinarity* and *motivation*.

First, these students are taking part in an integrated learning activity consisting of a practical application of knowledge and skills from three disciplines: Robotics, Object-oriented Programming and Event-based Programming.

Interdisciplinarity is an important learning principle: ''New thinking and innovation often occurs in the intersection between existing competencies and knowledge, and in the encounter between persons with different professional backgrounds. Firms are therefore seeking out knowledge workers who possess the ability to think across disciplines and to work together with others on common goals and tasks'' [26].

Furthermore, the students' motivation in their learning process is an important factor to be taken into account: ''Robotics has an inherent appeal on both emotional and intellectual level that makes it attractive to a broad range of learners across multiple dimensions, such as age, gender or academic interest'' [13].

3. Technological foundations

Mobile robots are autonomous or remotely operated programmable mobile machines capable of moving in specific environments. They use sensors to perceive their environment and make decisions based on the information obtained from them [27–29]. They range from the sophisticated space robots to the military flying robots.

Ball-shaped robots [30] have specific advantages, such as robustness and stability, that make them especially suitable for remote inspection tasks like taking measures of physical magnitudes (e.g. temperature, humidity and luminosity) in different environments [31], and for security-related applications [32]. They can also be used for educational purposes [33].

Sphero is a ball-shaped robot designed by Sphero (previously Orbotix) that can be controlled by a computer or by means of a smartphone (it can be used with iOS 4.0 or higher versions, and with Android devices starting from 2.2 operating system version). Sphero 2.0—the second generation of the robot [22]—is twice as fast, rolling at a speed of about two meters per second, and it is three times as brightly lit as the first generation robotic ball.

Sphero 2.0 is currently compatible with more than twenty-five applications and games, along with the standard Sphero app. The upgraded robot also comes with an inductive charger for extremely easy charge-and-go capability (Fig. 1). Its on-board technology offers automatic stabilization and precision control features, with a low slung

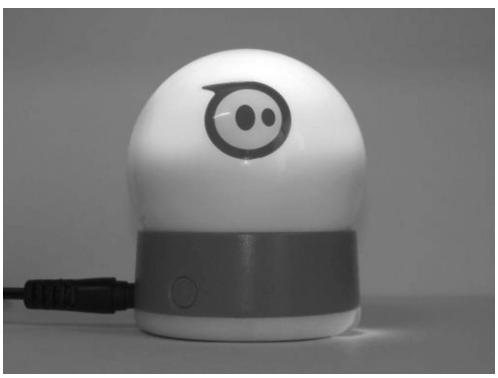

Fig. 1. Sphero 2.0 into the charging cradle.

center of mass for increasing energy efficiency and drivability.

The robot has the following features (Fig. 2):

- 72 MHz 32-bit ARM Cortex-M4 processor [34].
- Dual channel motor control loop running at ~400Hz.
- On-Board Bluetooth 2.0 connection [35].

Fig. 2. Inside Sphero 2.0.

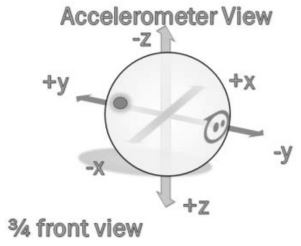

Fig. 3. Sphero accelerometer.

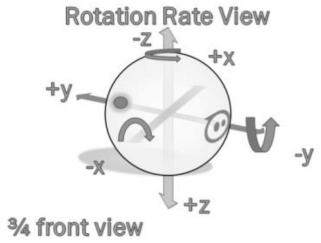

Fig. 4. Sphero gyroscope.

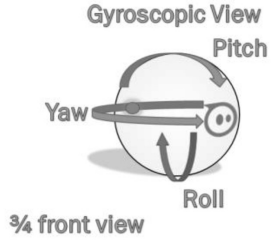

Fig. 5. Sphero IMU.

- Two 350 mAh lithium polymer (LiPo) rechargeable batteries.
- Fully programmable routines and behaviors with two built-in languages (*Macro* and *orbBasic*).
- 3-axis accelerometer +/–50mg precision and 3-axis gyroscope 2000°/second precision.
- Two 5x5mm RGB super bright LEDs.
- Two carbon brushed high-torque motors.

Sphero supports asynchronous data streaming of certain control system and sensor parameters [36]. As of Firmware 1.20, Sphero can stream—among others—values from the *accelerometer* for determining collisions and shake gestures, the *gyroscope* for measuring the angular velocity rate, and the *inertial measurement unit* (IMU) for determining the orientation of the robot (Figs. 3, 4 and 5).

4. Application of a software demo to the study of the C# object-oriented programming language

In this section we describe a simple C# software demo (*MovingSphero*) that uses the Sphero robot in order to illustrate some important OOP concepts [14, 15] such as *class*, *object*, *attribute* and *method*. Also, some event-based programming concepts such as *events* [37] and C# *delegates* [38, 39] can also be explained in the same way.

This demo, whose code can be accessed through the link http://www.atc.us.es/~josemari/MovingSphero.cs, has been developed using the *Microsoft Visual Studio 2010* IDE. Also, the entire ZIP package[1], that includes the API (*SpherOOP*) along with the demo, can be downloaded from the link http://www.atc.us.es/~josemari/MovingSphero. zip. Finally, a short video-clip of the demo execution can be watched accessing the URL http://youtu.be/swPNGf8RqZI.

C# [38] is an object-oriented and type-safe programming language [39]. It has its roots in the C family of languages, and has been standardized by ISO/IEC as *ISO/IEC 23270* and ECMA International as *ECMA-334* [40].

Although C# shares many characteristics with Java, it includes certain features that Java does not [41], such as *operator overloading*, *reference parameters*, *properties*[2], *enums*, *iterators* and the *foreach loop*, *delegates* (a sort of type-safe function pointers) and *a more consistent object model*. Some of these characteristics have been clearly taken from C++.

[1] This software is experimental and incomplete, and should be used only for demonstrative purposes.

[2] A C# *property* is a member that gives access to a feature of an object or a class. It can be accessed using the same syntax as a data field, though the compiler translates each access into calls on *get( )* and *set( )* accessors specified in the property. Therefore, properties are a natural extension of data fields.

The Web pages indicated in [42] provide various software development kits (SDKs) for writing Sphero applications. For example, there are SDKs for iOS, Android and Windows platforms. There are also "unofficial" SDKs for Mac OSX, as well as for Phyton and Ruby programming languages, and a basic library for the *Arduino* open-source electronics platform [43].

Windows 8.1 SDK [44, 45] provides a C# API for developing Windows 8.1 Store Apps around Sphero using the *Microsoft Visual Studio* Integrated Development Environment (IDE). However, in order to carry out a study about the teaching of OOP and C# language basic aspects, we think that a simple console program is the most suitable option.

So, we have found one API for developing Java applications (*Sphero Desktop API* [46]) and other two for writing C# applications: *BallControl* [47] and *SpheroNET* [48]. The first one is an open-source Sphero controller: A Bluetooth, accelerometer, camera and voice control application that can be compiled in various platforms, as Windows Desktop, Windows Store, Windows Phone 8, Mac and Android.

SpheroNET is essentially a wrapper for the low-level API developed by Sphero [49], which is based on the transmission of commands to a Sphero device over a Bluetooth connection. *SpheroNET* provides a class whose instance (or object) represents a Sphero robot. Thus, such object has a set of methods for managing the Bluetooth connection to the computer, the current color of the RGB LED and the load/execution of *orbBasic* programs, which are written in a special version of the BASIC language adapted for working with Sphero robots.

Compared to *BallControl*, *SpheroNET* has a simpler architecture, and can be extended with new properties and methods in order to use the moving and collision detection capabilities of a Sphero device without many difficulties. Thus, the new *SpherOOP* API provides a derived class that inherits all the features of the original *SpheroNET* objects, but it also includes new functionalities relating to movement and collision detection.

MovingSphero is a key interpreter that allows a simple user interaction with a Sphero robot. First, we use the namespaces containing the necessary classes for our program, including those ones from the *SpherOOP* API. Then, we declare the class variables whose access is shared by the *main( )* and *OnCollisionDetected( )* static methods, including the two sound objects and the RGB color array.

In the *main( )* method we declare a Sphero object, which will allow us to control our Sphero device after setting up the corresponding Bluetooth connection. We also declare an object for storing the current pressed key, to be used by the interpreter. The next statements initialize the RGB color array, the two sound objects, the collision counter and the variables that control the speed and the angle of the Sphero device trajectory.

After displaying the introductory messages on the screen, the program tries to set up a Bluetooth connection with the first available Sphero device. If a successful connection is established, the connection sound is played and the Sphero device RGB LED flashes in white eight times. Otherwise, an error message is displayed and the program execution finishes.

In the next step, the Sphero behavior is configured by initializing the corresponding object properties, such as the motion timeout, the back LED intensity and the collision detection capability. Also, the RGB LED is set to red and the *OnCollisionDetected( )* handler method is associated to the *CollisionDetected* event.

The interpreter code, based on a *switch-case* statement nested inside a *do-while* loop, starts executing right after the user instructions have been displayed. The user can then control the Sphero device using the cursor keys to set the angle (0°, 90°, 180° and 270°) that indicates the movement direction, as well as the <ENTER> key and the <SPACE> bar to make the robot start and stop rolling respectively.

At the end of the loop, the *Roll( )* method call upon the Sphero object makes the robot roll in the direction defined by the given angle value (first parameter) with the specified speed (second parameter). When the Sphero device is stopped, the cursor keys make the robot rotate around its own axis in order to get oriented according to the required angle.

When the user presses the <ESC> key, the execution flow of the program leaves the interpreter loop and the Sphero device movement is stopped by a call to the *Stop( )* method. The subsequent statements switch off the back LED, set the RGB LED to white for one second and, finally, turn off the robot.

Finally, the *OnCollisionDetected( )* handler method is called when a collision is detected by the Sphero robot and so, the corresponding collision event is raised. This method accepts two arguments: The *Sphero* object that has experimented the impact, and an instance of the *CollisionDetectedEventArgs* class, that stores information about it such as the values read from the accelerometer at the highest peak of the impact, or the speed of the robot at the time of the reported impact [49].

After the collision sound is played, a message indicating the collision number and the current angle of the robot trajectory is shown to the user. Next, the RGB LED is set to the next color in the RGB color array, the angle value is updated so that

the new movement direction is now the opposite to the previous one, and the Sphero device is made roll in the new direction. Finally, the collision counter is incremented by one and the execution flow is returned to the interpreter loop in the *main( )* method.

This demo is useful to illustrate some object-oriented and event-based programming basic skills for those students that are being initiated to these programming paradigms using the C# language. The main skills are:

Use of classes: The program class (*MovingSphero*), the robot class (*Sphero*) and a class to play sounds (*SoundPlayer*).

Use of objects: A Sphero robot, the last pressed key and a sound are represented and handled as objects.

Use of properties: The last pressed console key (*Key*) is a property of a *ConsoleKeyInfo* object. *BackLed* and *MotionTimeOut* are properties of a *Sphero* object.

Use of methods: *Roll( )* and *Stop( )* for a *Sphero* object and *Play( )* for a *SoundPlayer* object.

Use of event handlers: *OnCollisionDetected( )* allows a set of specific actions to be performed when the corresponding event (*CollisionDetected*) is raised.

5. Results and analysis

We have conducted these experimental tests under the same conditions as the previous ones—described in [4]—in order to be able to establish some relationship between the results obtained from the group of students who had the C# course with the software demo for the Sphero robot, and the achieved ones from the group who had the C# course with the software demos for the Sifteo cubes [50]. In short, we aim to provide a response to the following question: *Could a robot also be a suitable didactic tool in order to help students to understand better OOP basic concepts?*

The students of *Fundamentals of Computer Science*, taught in the first year of the Degrees in Mechanical, Electrical, Industrial Technology and Industrial Electronic Engineering (School of Engineering, University of Cadiz), are initiated in the C procedural programming language [51]. In the course they do not receive any formation on OOP. These initial conditions make of them a very suitable group for our study. Moreover, C++, Java and C# are languages based on C to some extent and thus the syntax in general, the basic data types and the control statements are known by these students.

As a sample for our study, we selected students that demonstrated a high performance level and a positive attitude during the course *Fundamentals of Computer Science*. In this way we can ensure that they were really interested in computer programming[3]. We selected a sample of thirty students aged 18–19. All of them were men, since the percentage of women registered in the studies of Industrial Engineering (at least, in the University of Cadiz) is usually very small. None of these students had a previous contact with Robotics.

Fifteen students—the control group—attended a standard C# OOP course (http://www.atc.us.es/~josemari/IntroCSharp.pdf [in Spanish]) that included practical demonstrations (i.e. computer execution) of example programs, without the technological contribution of the Sphero robot. The other fifteen students—the experimental group—attended a course that made use of such contribution (in addition to the standard material, it also included the Sphero demo described in the previous section). The total duration was ten hours (five daily sessions of two hours) for both courses, and the instructor for both groups was one of the authors of this work.

Once the courses (the standard course and the one that included the contribution of the Sphero robot as a didactic innovation) were taught, both student groups were asked to indicate their perception of the interest and the clarity level of the exposition (*IT* and *CL* variables), using a scale between one and four (to avoid the central tendency), as well as the time spent studying the course contents (*ST* variable). All the students also completed the same two tests: a multiple-choice exam for evaluating their understanding of basic OOP concepts, and a programming exercise also based on multiple choices in order to reduce the subjectivity in the marking process. From both tests, we obtained the following data for each student: the overall mark based on the number of correct answers (*MR1* and *MR2* variables), the time taken to solve each test (*TM1* and *TM2* variables) and the perception of the difficulty level (*DF1* and *DF2* variables).

Data from Table 1 has been obtained using the software *IBM SPSS Statistics*. It can be observed that for all the variables except *TM1* (the time taken by the students to complete the test for evaluating the understanding of basic OOP concepts), the Mann-Whitney U test has found significant differences between the samples corresponding to the experimental and the control group. The fact that this test has not found significant differences in that variable is not surprising, since it is feasible that the students of both groups have taken a similar time to complete an exam of a short duration (about ten minutes).

[3] Since the sample has been obtained using a pre-established selection criterion instead of a random sampling method, this work represents a pilot study: A starting point from which more detailed studies can be carried out.

Table 1. Mann-Whitney U test for independent samples ($\alpha = 0.05$)

Null hypothesis	p-value	Decision
The distribution of CL is the same between the categories of Exp.	0.005	Reject the null hypothesis.
The distribution of IT is the same between the categories of Exp.	<0.001	Reject the null hypothesis.
The distribution of ST is the same between the categories of Exp.	<0.001	Reject the null hypothesis.
The distribution of MR1 is the same between the categories of Exp.	0.009	Reject the null hypothesis.
The distribution of MR2 is the same between the categories of Exp.	0.007	Reject the null hypothesis.
The distribution of TM1 is the same between the categories of Exp.	0.217	Retain the null hypothesis.
The distribution of TM2 is the same between the categories of Exp.	0.037	Reject the null hypothesis.
The distribution of DF1 is the same between the categories of Exp.	0.011	Reject the null hypothesis.
The distribution of DF2 is the same between the categories of Exp.	0.005	Reject the null hypothesis.

Tables 2 and 3 show the values of a set of indicators, calculated as the mean values of the results from each group (experimental and control), along with the corresponding standard deviations.

The provided indicators for student perceptions (Table 2) are:

- Subjective perception of the clarity level (from 1 to 4) for the course exposition.
- Subjective perception of the interest level (from 1 to 4) for the course exposition.
- Subjective perception of the difficulty level (from 1 to 4) for tests 1 and 2 respectively.

The provided indicators for student learning data (Table 3) are:

- Time spent (expressed in minutes) to study the course contents.
- Achieved mark (from 0 to 10) for tests 1 and 2 respectively.
- Spent time (expressed in minutes) for tests 1 and 2 respectively.

The values of the indicators and the standard deviations are shown for each student group that have participated in the experiment:

- Experimental group: Students that have participated in a C# OOP course using the Sphero robot as a technological and didactic resource.
- Control group: Students that have participated in a standard C# OOP course.

From the obtained results, we can clearly observe that, for the *clarity level* and the *interest level* indicators (Table 2), as well as for the *achieved mark (test 2)* indicator (Table 3), the experimental group achieves higher values than the control group. A higher interest level for the exposition of a topic is related to a greater motivation in the student's learning process. As we have already mentioned, the C# course for the experimental group includes the software demo using the Sphero robot, and thus the achievement of higher values for the indicators could be explained as the result of using more meaningful and illuminating examples instead of the typical set of standard C# sample codes executed in a computer.

On the other hand, the experimental group has obtained a lower value than the control group for the *spent time (study)* indicator (Table 3). In our opinion, this result is related to a better comprehension of the contents explained during the course exposition. Therefore, less study time is needed to consolidate the learning of such contents.

Finally, although the most important differences

Table 2. Values of indicators and standard deviations (student perceptions) for the two groups of students

	Experimental group		Control group	
Indicator name	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
Clarity level (presentation)	3.40	0.49	2.53	0.81
Interest level (presentation)	3.80	0.54	2.53	0.88
Difficulty level (test 1)	2.67	0.47	3.33	0.60
Difficulty level (test 2)	2.93	0.44	3.67	0.60

Table 3. Values of indicators and standard deviations (student learning data) for the two groups of students

	Experimental group		Control group	
Indicator name	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
Spent time (study)	81.07	7.16	171.20	28.65
Achieved mark (test 1)	6.88	0.53	6.08	0.68
Achieved mark (test 2)	4.73	0.57	3.67	1.01
Spent time (test 1)	8.33	1.14	8.87	1.09
Spent time (test 2)	8.27	1.44	9.60	1.45

Table 4. Values of indicators and standard deviations (student perceptions) calculated in previous work [4] for the two groups of students

	Experimental group		Control group	
Indicator name	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
Clarity level (presentation)	3.57	0.62	3.32	0.69
Interest level (presentation)	4.01	0.65	2.93	0.57
Difficulty level (test 1)	2.43	0.90	2.53	0.83
Difficulty level (test 2)	2.57	0.49	2.81	0.56

Table 5. Values of indicators and standard deviations (student learning data) calculated in previous work [4] for the two groups of students

	Experimental group		Control group	
Indicator name	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
Spent time (study)	134.36	30.81	139.93	15.87
Achieved mark (test 1)	8.24	0.54	8.22	0.31
Achieved mark (test 2)	6.64	1.23	4.20	1.38
Spent time (test 1)	12.07	0.70	13.40	0.48
Spent time (test 2)	12.64	0.81	12.80	1.05

between the experimental and the control group results are found in the values of the *clarity level* and *interest level* indicators (Table 2), and the *spent time (study)* and *achieved mark (test 2)* indicators (Table 3), all the differences are in favor of the experimental group. Despite this work is a preliminary study, in our opinion such fact is important and must be taken into account.

Tables 4 and 5 show the results obtained in our previous work [4], in which the experimental C# course used the Sifteo Cubes [5]—a distributed tangible user interface[4]—as a teaching resource. Those devices can be handled as C# objects and have properties and methods that can be used in order to operate their internal peripherals (e.g. the clickable screen). They support event handler methods that allow specific actions to be performed when the corresponding events (i.e. shake, flip, tilt, screen click, approximation, etc.) are raised.

In order to be able to compare the results of both works, the material of the standard course for the control group, the test for evaluating the understanding of basic OOP concepts and the programming exercise were the same in both works. The only difference was the inclusion of the demo for the Sifteo Cubes as part of the experimental course. Also, the size of the two samples was the same: fifteen students for each group.

In the previous work, the results for the experimental group were also better than the ones for the control group. However, as it was also a pilot study, we can only conclude that the use of both devices as didactic tools—original Sifteo Cubes and Sphero 2.0 robotic ball—has been valid and suitable for achieving the purpose of facilitating the learning of C# language and OOP basic concepts to the students.

In order to obtain qualitative information that, in some way, could complement the quantitative experimental results, those students who took part in the experimental group for the teaching of the C# course were asked to summarize their experiences in the form of brief comments. Next, the more meaningful comments are shown:

Working with the Sphero robot has been interesting and enjoyable. It has encouraged me to take part in other robotic projects like this in the future.

Teaching the C# course with the help of a programmable robot has been a great idea since, in this way, a student does not need to wait three or four years for applying the knowledge acquired in such period. Students will be more encouraged and interested in learning the C# language in order to program the robot and work with it. Their levels of satisfaction and self-confidence will be increased, and their learning process will be more pleasant and efficient.

It has been a satisfactory and rewarding experience to carry out a practical activity which allows us to glimpse one of our options for the future (i.e. Robotics) without having to wait for several years. In short, this course has been appropriate, useful and motivating. I wish that more similar courses were organized in my university.

The experience of learning the C# language and consolidating the achieved knowledge later through the work with the Sphero robot—which can be programmed in such language—was highly satisfactory. Working with Sphero and understanding how it was put into operation—identifying and studying the different parts of the C# demo program corresponding to the establishment of the Bluetooth connection to Sphero, and the performance of movements and color changes—was so interesting.

I liked this course very much. I have found very interesting how a robot can operate under the control of a computer program.

I think that it is a good idea that a course about a programming language (C#) has also served us as an introduction to robotics and robot programming.

[4] Compared to Sifteo Cubes, Sphero has mobile capabilities as well as internal lighting (a set of color RGB LEDs), although it lacks a screen for providing visual feedback.

6. Discussion

In our opinion, the learning experience of the experimental group using the robotic ball in order to facilitate the study of the C# language, has helped to understand better the utility of computer programming to students of university careers not directly related to the Computer Science discipline, such as Industrial Engineering.

As shown in [4], students understand better OOP basic concepts if they are explained using a physical support that they can see and touch. In addition to confirming the results of our previous work, we consider that the contribution presented in this paper is valuable in itself, since it introduces students to the fields of robotics and robot programming, as well as to object-oriented programming and event-based programming, in a practical and enjoyable way. Thus, students carry out an integrated learning process in an interdisciplinary environment.

From our experience, we think that the use of the Sphero robot, as a didactic resource, not only has a positive influence on the student's learning process, but it also gives a greater quality to the professor's teaching activity. Certainly, the teaching of the C# language and the explanation of OOP basic concepts are both improved and made easier when they are carried out with the help of this technological support.

One of the possible drawbacks of the work presented in this article is its condition of pilot study, which represents a first step from which further and more detailed studies can be performed. Anyway, the obtained results as well as the student's comments highlight the robot utility as a didactic tool for teaching the C# language and OOP basic concepts.

Furthermore, only object-oriented and event-based programming basic skills are currently illustrated by the *SpherOOP* API. In the future, the possibility of developing an extended version of the API could be studied in order to implement OOP advanced skills, such as the use of inheritance and polymorphism.

An advantage of this work consists of the portable infrastructure used, which makes possible the replication of the experiments. The Sphero robot is a portable device due to its low weight and small size, whose price is affordable. Also, the software is open source [42–48] and there is available documentation [49].

7. Conclusion

In this work, we have applied the use of a simple mobile robot to the teaching of the C# object-oriented programming language, since the operation of the Sphero robot can be controlled by programs written in this language. The robot communicates wirelessly with a computer through a Bluetooth link.

From the analysis of the results presented in this work, we can conclude that the use of the Sphero robot as a didactic tool—through which OOP basic concepts are represented in a tangible and a visible way—for the teaching of C# has exerted a positive influence on the learning of both the language and the OOP basic concepts.

Furthermore, the educational approach proposed in this work has introduced the students to the fields of robotics and robot programming from an eminently practical point of view, and it also has familiarized them with OOP, thus leading to an enriching learning experience in an interdisciplinary context.

Acknowledgements—Illustrations 2, 3, 4 and 5 are provided by courtesy of Sphero.

References

1. L. Yan, Teaching Object-Oriented Programming with Games, *Proceedings of the 6th International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, Nevada (USA), 2009, pp. 969–974.
2. M. Overmars, Learning Object-Oriented Design by Creating Games, *IEEE Potentials*, **23**(5), 2004, pp. 11–13.
3. M. B., Rosson and J. M. Carroll, Climbing the Smalltalk Mountain, *ACM SIGCHI Bulletin*, **21**(3), 1990, pp. 76–79.
4. J. M. Rodríguez Corral, A. Civit Balcells, A. Morgado Estévez, G. Jiménez Moreno and M. J. Ferreiro Ramos, A game-based approach to the teaching of object-oriented programming languages, *Computers & Education*, **73**, 2014, pp. 83–92.
5. D. Merrill, E. Sun and J. Kalanithi, Sifteo Cubes, *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, Austin, Texas, 2012, pp. 1015–1018.
6. F. Barreto and V. Benitti, Exploring the educational potential of robotics in schools: A systematic review, *Computers & Education*, **58**, 2012, pp. 978–988.
7. B. A. Maxwell and L. A. Meeden, Integrating Robotics Research with Undergraduate Education, *IEEE Intelligent Systems*, **15**(6), 2000, pp. 22–27.
8. R. Mitnik, M. Recabarren, M. Nussbaum and A. Soto, Collaborative robotic instruction: A graph teaching experience, *Computers & Education*, **53**, 2009, pp. 330–342.
9. I. M. Verner and S. Gamer, Robotics Laboratory Classes for Spatial Training of Novice Engineering Students, *International Journal of Engineering Education*, **31**(5), 2015, pp. 1376–1388.
10. D. Alimisis, Educational robotics: Open questions and new challenges, *Themes in Science & Technology Education*, **6**(1), 2013, pp. 63–71.
11. B. Siciliano and O. Khatib (eds), *Springer Handbook of Robotics*, Springer-Verlag, Heidelberg, Germany, 2008.
12. J. Arlegui, E. Menegatti, M. Moro and A. Pina, Robotics, Computer Science Curricula and Interdisciplinary Activities, *Proceedings of the First International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Venice, Italy, 2008, pp. 10–21.
13. S. Kurkovsky, Mobile Computing and Robotics in One Course: Why Not?, *Proceedings of the 18th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, Canterbury, UK, 2013, pp. 64–69.
14. B. Meyer. *Object-oriented Software Construction, 2nd ed.*, Prentice-Hall PTR., Upper Saddle River, NJ, 1997.

15. J. R. Rumbaugh, M. R. Blaha, W. Lorensen, F. Eddy and W. Premerlani, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.
16. A. Angerer, A. Hoffmann, A. Schierl, M. Vistein and W. Reif, Robotics API: Object-Oriented Software Development for Industrial Robots, *Journal of Software Engineering for Robotics*, **4**(1), 2013, pp. 1–22.
17. T. H. Collett, B. A. MacDonald and B. Gerkey, Player 2.0: Toward a Practical Robot Programming Framework, *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*, Sydney, Australia, 2005.
18. A. Elkady and T. Sobh, Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography, *Journal of Robotics*, 2012, pp. 1–25.
19. C. Zieliński and T. Winiarski, Motion Generation in the MRROC++ Robot Programming Framework, *The International Journal of Robotics Research*, **29**(4), 2010, pp. 386–413.
20. S. Xinogalos, M. Satratzemi and V. Dagdilelis, An introduction to object-oriented programming with a didactic microworld: objectKarel, *Computers & Education*, **47**, 2006, pp. 148–171.
21. S. Xinogalos, An evaluation of knowledge transfer from microworld programming to conventional programming, *Journal of Educational Computing Research*, **47**(3), 2012, pp. 251–277.
22. Sphero (2015), *Sphero. The Original App-enabled Robotic Ball*, http://www.sphero.com/sphero, Accessed 4 November 2015.
23. L. D. Fink, *Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses*, John Wiley and Sons, Inc., San Francisco, CA, 2003.
24. J. Carroll and F. Polo, Augmented Reality Gaming with Sphero, *Proceedings of the 40th International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Anaheim, CA, 2013.
25. Sphero (2015), *A New World of Gaming: Creating Mixed Reality*, http://blog.sphero.com/blog/a-new-world-of-gaming-creating-mixed-reality, Accessed 5 November 2015.
26. Danish Business Research Academy (DEA/Danmarks ErhvervsforskningsAkademi), Danish Forum for Business Education (FBE), *Thinking across Disciplines—Interdisciplinarity in Research and Education*, Copenhagen, Denmark, 2008.
27. G. A. Demetriou, Mobile Robotics in Education and Research, in Z. Gacovski (ed), *Mobile Robots—Current Trends*, InTech, Rijeka, Croatia, 2011, pp. 27–48, DOI: 10.5772/26295, http://www.intechopen.com/books/mobile-robots-current-trends/mobile-robotics-in-education-and-research
28. J. L. Jones, A. M. Flynn and B. A. Seiger, *Mobile Robots. Inspiration to Implementation. 2nd ed.*, A. K. Peters, Ltd., Natick, MA, 1999.
29. J. M. Rodríguez Corral, A. Morgado Estévez, F. Cordón González, R. González Chacón and I. García Vargas, Microbots: Foundations and Applications (in Spanish), *Novática*, **192**, 2008, pp. 42–47.
30. T. Ylikorpi and J. Suomela, Ball-shaped Robots, in H. Zhang, (ed), *Climbing and Walking Robots: towards New Applications*, InTech Education and Publishing, Vienna, Austria, 2007, pp. 235–256.
31. J. D. Hernández, J. Barrientos, J. del Cerro, A. Barrientos and D. Sanz, Moisture Measurement in Crops using Spherical Robots, *Industrial Robot: An International Journal*, **40**(1), 2013, pp. 59–66.
32. M. Seeman, M. Broxvall, A. Saffiotti and P. Wide, An Autonomous Spherical Robot for Security Tasks, *Proceedings of the 2006 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety (CIHSPS)*, Alexandria, VA, pp. 51–55, 2006.
33. F. Michaud, J. F. Laplante, H. Larouche, A. Duquette, S. Caron, D. Létourneau and P. Masson, Autonomous Spherical Mobile Robot for Child-Development Studies, *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, **35**(4), 2005, pp. 471–480.
34. ARM Ltd. (2015), *ARM Cortex-M4 Processor*, http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php, Accessed 6 October 2015.
35. Bluetooth Technology Special Interest Group, Inc. (2015), *Bluetooth Specification*, https://www.bluetooth.org/en-us/specification, Accessed 6 October 2015.
36. GitHub, Inc. (2015), *Sphero Sensor Streaming*, https://github.com/orbotix/Sphero-iOS-SDK/tree/master/samples/SensorStreaming, Accessed 20 October 2015.
37. T. Faison, *Event-Based Programming: Taking Events to the Limit*, Apress, Breinigsville, PA, 2006.
38. M. Michaelis, *Essential C# 4.0, 3rd ed.*, Pearson Education, Inc., Ann Arbor, MI, 2010.
39. Microsoft Corporation (2012), *C# Language Specification. Version 5.0*, http://www.microsoft.com/en-us/download/details.aspx?id=7029, Accessed 22 October 2015.
40. ECMA International, *C# Language Specification (4th ed.), Standard ECMA-334*, 2006, http://www.ecma-international.org/publications/standards/Ecma-334.htm, Accessed 22 October 2015.
41. S. Reges, Can C# Replace Java in CS1 and CS2?, *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, Aarhus, Denmark, 2002, pp. 4–8.
42. Sphero (2015), *Sphero Developers build games, have fun*, https://developer.gosphero.com, Accessed 4 November 2015.
43. GitHub, Inc. (2015), *Arduino Sphero Library*, https://github.com/cmonr/Arduino-Sphero-Library, Accessed 4 November 2015.
44. GitHub, Inc. (2015), *Sphero Win SDK*, https://github.com/orbotix/Sphero-Win-SDK, Accessed 13 October 2015.
45. Microsoft Corporation (2015), *Windows 8 Sphero SDK C# Sample for Visual Studio 2013*, http://code.msdn.microsoft.com/windowsapps/Sphero-SDK-Sample-2b18913c, Accessed 13 October 2015.
46. GitHub, Inc. (2015), *Sphero Desktop API*, https://github.com/nicklasgav/Sphero-Desktop-API, Accessed 29 October 2015.
47. GitHub, Inc. (2015), *Ball Control. A BlueTooth/Accelerometer/Camera-/VoiceControl app for fun and for Developer Competition*, https://github.com/slodge/BallControl, Accessed 29 October 2015.
48. T. Bladh (2013), *Balls out fun with the Sphero and .NET*, http://thomasbladh.com/2013/01/01/balls-out-fun-with-the-sphero, Accessed 29 October 2015.
49. GitHub, Inc. (2013), *Developer Resources. Sphero Docs*, https://github.com/orbotix/DeveloperResources/zipball/master, Accessed 20 October 2015.
50. Sifteo Inc. (2011), *Sifteo Cubes*, https://www.youtube.com/watch?v=dF0NOtctaME, Accessed 9 November 2015.
51. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed., Prentice Hall, NJ, 1988.

José María Rodríguez Corral received his Master degree in Computer Engineering and his Ph.D. from the University of Seville (Spain) in 1993 and 2002 respectively. From November 1993 to June 1995 he worked on robot control with the "Robotics and Computer Technology" research group at the University of Seville. After working as Associate Professor at the University of Cadiz (Spain) from 1995 to 1998, he is currently "Profesor Titular" of Computer Languages and Systems at the same University, and his research interests include robotics, bus emulation and parallel systems. He is a member of the "Applied Robotics" group at the University of Cadiz, and he is author of various papers and research reports on computer architecture.

Claudio Antonio Amaya Rodriguez received his Master degree in Computer Engineering and his Ph.D. from the University of Seville (Spain) in 1992 and 1999 respectively. Since 1994, he worked with the ''Robotics and Computer Technology'' research group at the University of Seville. After working as Associate Professor at the same University from 1994 to 2003, he is currently ''Profesor Titular'' of Computer Architecture, and his research interests include embedded systems, real-time architectures and machine-to-machine communications.

Antón Civit Balcells received his Master degree in Physics (electronics) and his Ph.D. from the University of Seville (Spain) in 1984 and 1987 respectively. After working for several months with Hewlett–Packard he joined the University of Seville where he is currently Full Professor of Computer Architecture. He is the author of various papers and research reports on computer architecture, rehabilitation technology and robotics. He is the director of the ''Robotics and Computer Technology'' research group at the University of Seville. His research interests include advanced wheelchairs, robotics and real-time architectures.

Arturo Morgado-Estévez received his Master degree in Industrial Organization Engineering and his Ph.D. from the University of Cadiz (Spain) in 1997 and 2003 respectively. After working on ASIC design with the ''Microelectronics'' research group at the same University from 1989 to 1998, and with the ''Robotics and Computer Technology'' group at the University of Seville from 1998 to 2010, he is currently the director of the ''Applied Robotics'' group at the University of Cadiz. He is ''Profesor Titular'' of Computer Architecture at the University of Cadiz since 1991, and his research interests include robotics and life-inspired systems. He is author of various papers on computer architecture.

Daniel Molina Cabrera received his Master degree in Computer Engineering and his Ph.D. in Artificial Intelligence from the University of Granada (Spain) in 1995 and 2007 respectively. After researching at the University of Granada, he started teaching from 2006 at the University of Cadiz, where he is currently Associate Professor. He is a member of the research group ''Soft Computing and Intelligent Information Systems''. His research interests include metaheuristics, automatic optimization (in continuous and high-dimensional problems), and he has published several papers about statistical testing. He has authored or co-authored more than 40 refereed papers in journals and conferences, and his H-index is 10 till date.

Fernando Pérez-Peña received his Engineering degree in Telecommunications from the University of Seville (Spain) and his Ph.D. degree in the field of Neuromorphic Engineering from the University of Cadiz (Spain) in 2009 and 2014 respectively. While studying for his engineering degree, he worked as system engineer for the main ship-builder of the Spanish navy (Navantia) from 2007 to 2010. In the second semester of 2009, he joined the University of Cadiz as part-time lecturer. Currently, he is full-time lecturer at the same University, and his research interests include neuromorphic engineering, FPGA digital design, motor control and robotics.