British Journal of Educational Technology Vol 42 No 4 2011 624–637
doi:10.1111/j.1467-8535.2010.01056.x

# Improving teaching and learning of computer programming through the use of the Second Life virtual world

**Micaela Esteves, Benjamim Fonseca, Leonel Morgado and Paulo Martins**

*Micaela Esteves is a lecturer in computer science at the Polytechnic Institute of Leiria, Portugal. Benjamim Fonseca is a researcher at CITAB—Centre for the Research and Technology of Agro-Environment and Biological Sciences at the University of Trás-os-Montes e Alto Douro, Portugal, where he lectures in computer science. Leonel Morgado and Paulo Martins are both researchers at GECAD—Research Group on Knowledge Engineering and Decision Support, and lecture at the University of Trás-os-Montes e Alto Douro, Portugal. Address for correspondence: Micaela Esteves, Polytechnic Institute of Leiria, ESTG, Morro do Lena—Alto do Vieiro 2411-901 Leiria Apartado 4163, Portugal. Email: micaela@estg.ipleiria.pt*

**Abstract**
The emergence of new technologies such as three-dimensional virtual worlds brings new opportunities for teaching and learning. We conducted an action research approach to the analysis of how teaching and learning of computer programming at the university level could be developed within the Second Life virtual world. Results support the notion that it is possible to use this environment for better effectiveness in the learning of programming. The main results are the identification of problems hampering the teacher's intervention in this virtual world and the detection of solutions for those problems that were found effective to the success in using this environment for teaching/learning computer programming.

## Introduction

Technology has become crucial in educational development and for the revolution in learning systems (Olapiriyakul & Scher, 2006). Technology creates and transforms the learning and teaching processes, which brings new opportunities to the educational system. There has been recognition in the scientific literature about the use of virtual worlds in higher education (Dickey, 2003; de Freitas & Neumann, 2009), and Second Life® (SL) is currently the most mature and popular multiuser virtual world platform that has been used for this purpose (Warburton, 2009). However, clear guidelines for practice remain difficult to find (Warburton).

We conducted a study observing students' apprenticeship in SL and teachers' experience, with the aim of analysing how the processes of teaching and learning computer programming occur within this virtual world. Thus, some questions come up to our mind: What are the problems, for both teachers and students, in using SL for teaching/learning computer programming? Can these problems be solved, and how?

By beginning to address whether the difficulties of using SL for teaching programming can be overcome, the door has been opened for both qualitative and quantitative studies to determine if SL really does improve students' comprehension of basic programming concepts. Hence, the aim of this paper is to present a framework for teaching/learning of computer programming within the SL virtual world, to help students improve their programming apprenticeship, based on rigorous academic research. Essentially, we argue that the advent of teaching/learning of programming within SL brings new opportunities for students to improve their performance. Most conclusions can also be applied to the OpenSimulator environment as it replicates most of SL's functionality, lacking only on social aspects, at least for the moment.

This paper is organized as follows: first, our motivation and related work is outlined; then, the project methodology is described; we follow with the findings and the discussion of the framework presented; finally, the paper presents suggestions for using SL for teaching and learning of computer programming and concludes with suggestions for future research.

## Related work and motivation

Programming is a fundamental skill that all computer science students are required to learn. However, programming courses are generally regarded as difficult and often have the highest dropout rates (Gomes, Areias, Henriques & Mendes, 2008; O'Kelly & Gibson, 2006; Robins, Rountree & Rountreen, 2003). In the scientific literature, many reasons are pointed out for this, such as the following. *Methodology and tools used*—traditional teaching methods, normally based on lectures and specific programming language syntaxes, often fail in what concerns the students' motivation in getting involved in meaningful programming activities (Lahtinen, Mutka & Jarvinen, 2005; Schulte & Bennedsen, 2006). Programming languages typically used in programming classes are professional in nature, such as C, C++, C# and Java; they have extensive and complex syntaxes, rendering learning difficult for beginners (Jenkins, 2002; Motil & Epstein, 2000). *Students' difficulties with abstract concepts*—knowing how to design a solution to a problem, subdivide it into simpler code able subcomponents, and conceive hypothetical error situations for testing and finding out mistakes (Esteves, Fonseca, Morgado & Martins, 2008); difficulties in understanding even the most basic concepts (Lahtinen *et al*, 2005; Miliszewska & Tan, 2007) such as variables, data types or memory addresses as these abstract concepts do not have direct analogies in real life (Lahtinen *et al*; Miliszewska & Tan); and not knowing how to use the programming language correctly to create a program (Lahtinen *et al*; Winslow, 1996).

The use of animation to show program execution has been used to minimize the students' difficulties (Soloway, 1986; Stasko, Domingue, Brown & Price, 1998). In the scientific literature, we can find programs that only show the animation of a specific algorithm (Dershem & Brummund, 1998; Michail, 1996), in which students cannot make any changes, and even tools that simulate/animate any program made by them (Ben-Bassat Levy, Ben-Ari & Uronen, 2003; Esteves & Mendes, 2004). One of the most successful program simulation software packages, used for program visualization, has been Karel the Robot (Pattis, 1981). Karel the Robot is a programmable visualization software that uses the same principles as the well-known LOGO language (Papert, 1980). The robot executes a sequence of commands that the user has written as part of a program moving about two-dimensional grids. Over the last decade, curricula for computer science has made a transition to C and then to object-oriented languages. In response, Karel has undergone several updates, the latest being Karel++ (Bergin, Stehlik, Roberts & Pattis, 1997), a C++-like version that brought Karel into the object-oriented age.

Environments such as ALICE (Dann, Cooper & Pausch, 2000), JELIOT (Ben-Bassat Levy *et al*, 2003), BlueJ (Kölling, Quig, Patterson & Rosenberg, 2003) and RAPTOR (Carlisle, Wilson, Humphries & Hadfield, 2005) have been used to teach imperative programming in undergraduate introductory computer science courses. All these environments generate concrete visual representations of a program. However, ALICE is a three-dimensional (3D) interactive graphics programming environment for Windows that makes it easy to create an animation for telling a story, playing an interactive game or creating a video to share on the Web (Cooper, Dann & Pausch, 2000). In ALICE's interactive interface, students drag and drop graphic tiles to create a program, where the instructions correspond to standard statements in a production-oriented programming language, such as Java, C++ or C#. It is also object based by writing simple scripts in which its users can control 3D object appearance and behaviour. The benefit of using it is that it allows students to be involved and at the same time have the ability to develop an intuitive understanding of basic concepts in a visual feedback environment (Dann, Cooper & Pausch, 2001). Thus, students using ALICE are immediately able to see how their animated programs run because the highly visual feedback allows them to relate the program 'piece' to the animation action, and this leads to an understanding of the actual functioning of different programming language constructs (Dann, Cooper & Pausch, 2001).

We find in SL a persistent online 3D virtual world, containing the same characteristics as ALICE, ie, the possibility of students programming the objects' behaviour by writing simple scripts, and receive an immediate visual feedback of how their programs run. In addition to these characteristics, SL allows several users to connect, interact and collaborate simultaneously at the same time and in the same (virtual) space (see Figure 2). SL enables synchronous collaboration among students because the system allows two or more avatars to edit the same object and include their own scripts, which act concurrently on the object (and may exchange messages). Also, it is possible to share scripts so that students can access and edit the same piece of code while programming it. Asynchronous collaboration is also supported because the SL world is persistent: students and teachers may access and leave in-world objects (with scripts) and messages to the other members (group messages and private messages are supported). When a user logs in, all his or her messages are shown. On the other hand, he or she can also see some objects left in the world by others (and edit them, if adequate permissions have been set), and set up or edit his or her own objects with scripts to interact with those other objects/scripts present in the world.

At the time of writing, some authors consider SL to be the environment which allows more experimentation, collaboration and immersion compared with other virtual learning programs (Salmon, 2009). Collaborative environments can offer important support to students in their activities for learning programming. According to Newman, Goldman, Brienne, Jackson and Magzamen (1989), collaboration in problem solving provides not only an appropriate activity but also promotes reflection, a mechanism that enhances the learning process. Students that work in groups need to communicate, argue and give opinions to the other group members, encouraging the kind of reflection that leads to learning.

Another important aspect about SL is that students are integrated in an international community of programmers, as well as exposed to authentic content and culture (Warburton, 2009). Bauman (1991) claims that in the change to the modern functionally differentiated society, individual persons are no longer firmly rooted in one single location or subsystem of society, but rather must be regarded as socially displaced. The individual needs to establish a stable and defensible identity to differentiate the self from the outer world, but at the same time needs the affirmation of social approval. The socialisation and transition to work is immediate, and when the novices show evidence of professional skills in practice, this leads to legitimate participation in the professional community (Wenger, 1998). The practical applications of the acquired knowledge in the community, its reflection and exchange are some of the strategies suggested by Fleury and Oliveira Junior (2001) and Dillenbourg (2000). These results influenced some of the opinions we have had when we think about using the SL 3D virtual world as an environment to teaching/learning programming.

## Application of the action research (AR) methodology

This research adopted AR methodology, which can be described as a family of research methodologies that involve an intervention or change (action) on part of the researcher while research (or understanding) occurs. AR is a cyclical process that incorporates the four-step processes of

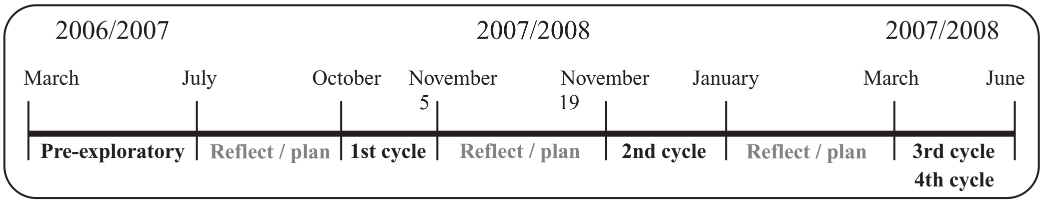


*Figure 1: Timeline for the action research of our study*

planning, acting, observing and reflecting on the results generated from a particular project or body of work (Dick, 1999; Zuber-Skerritt, 2002). This choice was made in view of the two main advantages of AR (Zuber-Skerritt). Firstly, it allows us to study a problem while introducing controlled changes, with the aim of discovering how to improve the process of teaching/learning. Secondly, it also provides the researcher with a large degree of flexibility, something of great importance when acting within an evolving information technology project concerning a problem about which little was known beforehand (Zuber-Skerritt).

When there is not enough research literature on the field of study, as in this case, it is necessary, before the first research cycle, to make a preliminary exploratory experience in order to identify the basic problems and feed the first planning (Lessard-Hébert, Goyette & Boutin, 1990). The research process is never completed, but a plateau is reached when the reflection at the end of a cycle deems that the amount of collected knowledge on the process is significant (Zuber-Skerritt, 2002).

## Research strategy

Our research strategy was to focus on the understanding of the teaching-learning process of computer programming using SL. We employed four cycles in this study, from March 2007 to July 2008. Pre-exploratory research took place during the second semester of the academic year 2006/2007, the first and second cycles of AR during the first semester of 2007/2008, and the third and fourth cycles in the second semester of that same academic year. Figure 1 presents a diagram of the full process.

### *Data collection design*

Data collection was planned at the beginning of the research. We used several data sources for triangulation and reducing bias in our analysis, summarized in Table 1. In this study, the researcher was the main instrument of the observation and action. This participant observation was an attempt to discover the meaning, dynamics and processes involved in the events. Through this method, some reflections on the lessons undertaken were done. The observation of these lessons focused on

- how students and teacher interact with each other and with the environment;
- the virtual classroom activities;
- the use of the SL interface; and
- challenges, constraints and possibilities of teaching and learning within the SL.

Furthermore, questionnaires were also developed, aimed at establishing intuitive ideas about how the participants understood the difficulties and potential of tasks. These questionnaires were not only important indicators of how students saw the learning process in-world but also contributed to a better understanding, on the part of the investigator, of the issues related with the students' motivation and performance, the students' difficulties in learning to program objects, and the impact of the community and the teaching process.

Table 1: Data sources

| Data sources | Explanation |
|---|---|
| Daily session reports | The teacher-researcher at the end of each session wrote a report with the key points of the each session. |
| Virtual classroom images | Captured as screenshots by the teacher, when witnessing key points during classes. |
| Questionnaires | At the beginning, middle and end of the process, with open-ended questions, on the learning/teaching methods and the students' views about the project they had developed and their learning in this environment. |
| Student and teacher communication: participatory observation | All communication between teacher-researcher and students, which was text based, was entirely recorded and saved at each session, as a tool to provide context at a later time for students' doubts and opinions. |

### *Participants*

Our participants were computer science students from the University of Trás-os-Montes e Alto Douro (UTAD), in Vila Real, Portugal, and from the Higher School of Technology and Management (ESTG—Portuguese-language acronym) of the Polytechnic Institute of Leiria, Portugal. These students took part in the research process while developing elective alternative assignments on the following compulsory subjects: Laboratório de Informática I of the first curricular year (Laboratory. I); Laboratório de Informática II (Laboratory. II) and III (Laboratory. III) of the second curricular year (at UTAD); and Projecto I (at ESTG). In all the cases, these subjects had as their main goal to allow students to develop a semester-long project in order to improve their programming skills. Other students, taking the same subjects, were developing assignments using other programming languages and environments. In these cases, there were no prescribed lectures.

We had three different types of students: beginners (Group A); students with some knowledge of programming (Groups B1 and B2); and students with relevant experience in programming semester-long projects (Group C). The students from group A, at UTAD, were enrolled in the second semester of the first curricular year. Although they were at the initial stage of learning how to program, they had been exposed in the previous semester to its introductory aspects in about 30% of two subjects. The project developed in SL was the students first contact with a semester-long programming project (ie, not just a class assignment). In Groups B1 and B2, the students were more advanced in learning how to program: some were at the UTAD, enrolled in the second year, first semester, and had already studied introductory aspects of the C programming language in the previous semester and developed a semester-long command-line project in C. While they were participating in this research, they were also taking a different course on object-oriented programming in C++. Others were at the ESTG, where they were enrolled in a postsecondary technical course (CET—Portuguese-language acronym) and had previously studied C programming for one semester. In group C, the students were from UTAD, enrolled in the second year, second semester, and thus, at a more advanced stage of learning how to program: they had completed courses in C and C++ programming and developed a semester-long command-line project in C++. Although these students still required teacher support, they had some autonomy in using and studying programming.

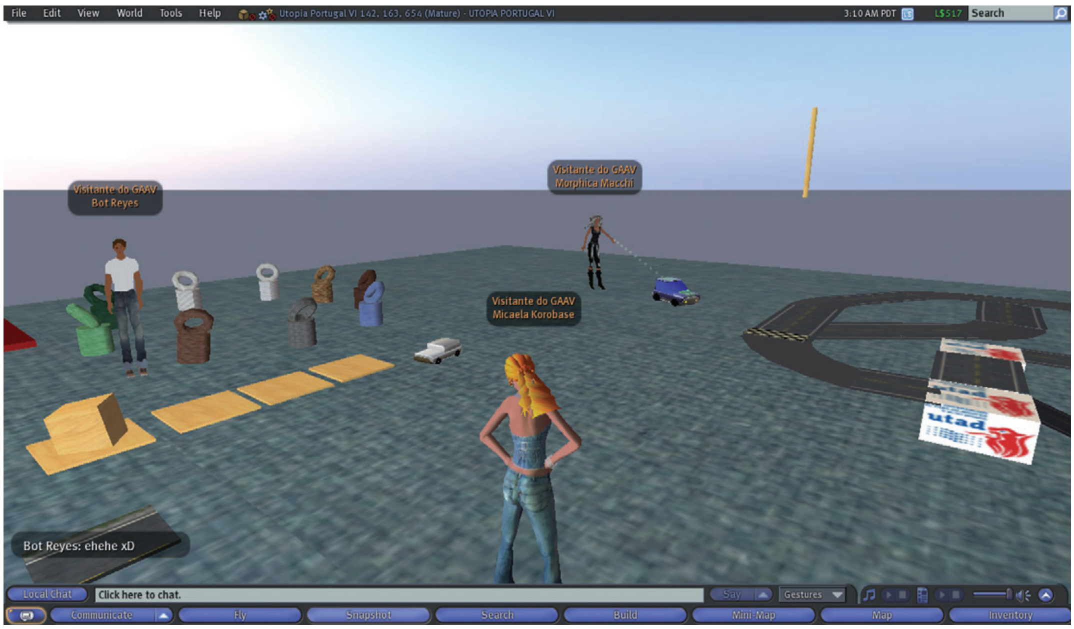
In this study, five students from Group A and four from Group C, both from UTAD, participated in the preliminary exploratory research phase. In the first and second AR cycles, the students were only from Groups B1 and B2 (10 students from UTAD and six from ESTG). In the third and fourth cycles, there were nine students from Group A.

*Procedure*

We used SL itself as a programming environment. We proposed students for their participation in this research to develop a project inside SL using its scripting language, Linden Scripting Language (LSL), which has C-style syntax and keywords. The 3D objects created in SL can receive several LSL scripts that are executed concurrently. Each script has its internal state machine: program flow is sequential, uses common methods from imperative/procedural programming, such as procedures and flow-control primitives, but structured by triggering events and responding to them (events are triggered through either environment interactions or programmatic components).

An identical project description was presented to all students enrolled in the course: after that some volunteered to participate in this research. From this point on, they formed pairs and developed their projects inside SL, collaborating with each other. Teachers and students met remotely, in-world, SL, once a week, for about 2 hours, to keep track of students' progress, exchange ideas and make suggestions. Figure 2 illustrates one of the sessions in SL, with the teacher's avatar in the centre and two students' avatars developing their work. Face-to-face meetings in the physical world did not take place in this process because the teacher-researcher was in Leiria and the students were in Vila Real, 270 km apart. Only once a month did they meet to talk about the project in Vila Real. In all the occasions, students had some difficulties about the code they had implemented, and they shared it with the teacher. Consequently, they observed it together and at the same time found out what was wrong and followed the teachers' indications/instructions. In this way, students could correct the code and proceed.

SL enables voice-based or text-based communication through public or private channels. However, voice-based communication requires that students and teacher are in a physical context with little environmental noise and where they can talk freely without disturbing other people nearby. For this experiment, such context could not be ensured at all times: students often used wireless connections in classrooms where other colleagues could be developing other (non-SL)



Bot Reyes: ehehe xD

*Figure 2: An example of a session in Second Life*

projects or even in college halls/corridors or bars; only sometimes did they work from the relative tranquillity of their homes. This led us to focus on using text-based communications for their greater freedom of physical context for educational activities. Text-based communication retains a history that can be read and reread, even if the physical context of a student/teacher causes distractions (a phone ringing, people passing by, etc.). All text-based communication that occurs through the public channel can be seen by everyone who is within 20 m of virtual world space, whereas in a private channel the communication is just between two people. The students often communicated with the teacher through a private channel.

## Findings

The project was done within a specific organization. We now turn to analytic generalization of that AR to build a practical framework linked to existent literature that explains how this virtual world can be used to teach and learn programming. A more detailed explanation of all research phases can be found in Esteves *et al* (2008, 2009). Our findings and framework are summarised in Table 2 and discussed ahead.

*Table 2: Framework for teaching/learning computer programming inside Second Life*

| *Elements* | *Procedures* |
|---|---|
| Communication | Public channel—for general explanations. |
| | Private channel—for private explanations. |
| Project | The project should be complex enough that the cooperation from all members of the group will be necessary. It must have a strong visual behaviour and should be adapted to the level of knowledge that students have. |
| Methodology | Project-based learning. |
| | Use of an outside platform (eg, Moodle or another learning management system) as a repository of learning materials. |
| | Support interaction with the SL user community. |
| Classroom (workspace) | Identify classroom areas for each group to work. |
| | Provide sample objects with simple programs for students to use as a reference. |
| Lectures | The teacher should be physically present in the first class to explain the SL interface. |
| | Teacher should prepare beforehand short phrases, ready to copy and paste when necessary. |

SL, Second Life.

### *Findings*

Our concern throughout the development of this study was to find out what the implications were for teachers teaching through SL; what problems teachers and students face, whether these problems can be fixed, whether motivation would be present. That is, not only finding if SL could be used but also how it could be used. As described in the previous sections, the preliminary exploratory research and subsequent AR cycles allowed us to better understand the problem and gather information on how to teach in SL and help students learn. We identified three important issues: communication between students and teacher, students' process of learning, and the teaching process itself.

### *Communication*

The first issue was communication: how the teacher could explain subjects and clarify students' doubts within SL. In our research, voice communication was not used, but rather text-based communication, as explained earlier. Our first approach to text-based communication was to use the public channel for explaining subjects and clearing students' doubts. However, we quickly observed that this meant all the messages appeared on the screen at the same time, in intertwined conversations. Consequently, it was difficult to follow a line of reasoning or conversation—to the point of sometimes mistaking the source of a specific statement (even though each statement was accompanied by the name of who wrote it). We then tried the approach of using a private channel to clear students' doubts and use the public channel only for the teacher to explain subjects or call attention of all students for a particular matter. The students' evaluation was that they appreciated this way of communicating because they felt they had a private teacher to whom they could present their doubts without feeling embarrassed to be lagging in relation to the others. From the teacher's perspective, the use of a private channel to explain the students' doubts was an important issue because she could develop a sense of trust and safety within the electronic community. In the absence of this trust, learners would feel uncomfortable and constrained in posting their thoughts and comments (Anderson, 2004). However, this approach originally led to delayed feedback time because the students would not be aware that the teacher was responding to someone else's queries. Research on assessment in distance education has shown that rapid feedback is important for both understanding and motivation to complete courses (Rekkedal, 1983); therefore, we needed to solve this issue. Yet there was no escaping the fact that the teacher had to provide attention to several groups at the same time. In order to give students an immediate response during the lectures, a simple solution was found: the teacher could keep several common sentences ready for copy-pasting, to provide this feedback without shuttling back-and-forth between different private communications: eg, a few sentences such as 'OK, just a moment, I'm talking to your colleague … ' or 'there is not a semi-colon at the end of the instruction … '. Hence, the students would not be waiting so long for the teacher to provide some feedback to their doubts.

### *The students' process of learning*

#### Project

The project was the starting point for the students' learning process. Hence, one of the issues that concerned the teacher was the type of project that could be more adequate for the teaching/learning process inside SL. During this research, the researcher tried two types of projects:

- Visual—involving the building of one or several 3D objects, such as a dog, a robot, a car or a motor-racing track, and the development of several scripts in a program, with the aim of achieving the behaviour that each object had to execute.
- Nonvisual—a project that consisted only in text data processing, meaning that the objects respond to text commands from avatars and reply in the same manner, or store data from in-world events in lists of strings.

Our trials confirmed the literature (Kiili, 2005; van Dam, 2005) about the benefits of visualization to certain aspects of problem-solving performance in novices, as well as to the level of engagement and motivation that students gain from constructing and presenting their own visualizations. With the visual projects, the students had an obvious feedback regarding the correctness of their program: it would suffice to look at the behaviour of the object. For example, the robot should follow its owner's orders. We verified that using a nonvisual project did not have a good impact on the students' performance because they were focused primarily on nonvisible techniques such as data structures and string processing, benefiting from the SL environment just for enhanced context and not as a source of feedback for programming behaviour. It would not be unusual for a student to assume that a script was right just because it was outputting apparently correct data, for instance, whereas erroneous robot behaviour was a cause for much more concern for students. Mostly, throughout the nonvisual project, students did not learn properly and struggled with themselves, not understanding why they had to do such a kind of project inside this environment. The teacher spent her time trying to motivate them, without results. As Duch (2001) mentioned, effective problems should engage the students' interest and motivate them to probe for deeper understanding of the concepts being introduced. Learning demands both the fun of playing with ideas and the hardness of refining and reworking these ideas, and that both complementary parts are needed for learning (Barrett, 2005). The fun transpired to be what Papert (1996) termed 'hard fun', in that it is both challenging and interesting, and this implies 'hard'.

### Students' difficulties

A particular important aspect in the learning of programming is the students' reaction to the compilation and execution errors (Esteves & Mendes, 2004; Lahtinen *et al*, 2005), which are inevitable in the learning process. In relation to the compilation errors, the methodology used by the teacher was writing comments in the students' code, explaining how and why the errors occurred, which helped the students to avoid the mistakes and understand them.

In the execution errors, we observed two types of situations: students who did not understand the project and the others whose main difficulty was understand the predefined function of LSL language. In the first case, they had difficulty in structuring their thought because they did not understand what was asked of them—and consequently, execution errors occurred because students had not implemented a correct algorithm. In this study, the students had to submit to the teacher the algorithm for the project in the second week of work, and from this point they developed their project. The use of problem-based learning (PBL) methodology helped students overcome these difficulties; once within PBL, the problem acts as the catalyst that initiates the learning process (Duch, 2001). Furthermore, the students had to elaborate on their initial ideas and critically evaluate what they knew and did not know. Finally, they had to formulate their learning issues for self-directed study. After about 1 week, the students met again to report and synthesize their findings in relation to the problem. The contribution of the teacher as a tutor in this discussion is important because she challenged the students to clarify their own ideas, inciting students to elaborate on the subject matter, questioning ideas, looking for inconsistencies and considering alternatives. In the second one, we concluded, after several interactions, that the causes for the errors were the difficulty in understanding the English language used in the reference sources. For that reason, we decided to translate into Portuguese the available online references for the main functions, and include examples.

However, some of the students of Group B1, who already had contact with programming using the C language, presented many faults in understanding basic programming concepts, namely cycle instructions (ie, when to use them and how) and using functions (ie, if given a function definition, they did not know how to use it—ie, how to call a function). A point to stress is that these students did not recognise their faults. They said 'I know the C language, I do not have difficulties in C', but then they were not able to put that knowledge into practice when developing the project. In relation to these difficulties, Winslow (1996) mentioned that novice programmers neglect strategies, are limited to surface knowledge of the subject, and that knowledge is fragile. Fragile knowledge is described as something that a student knows but fails to use when necessary. According to O'Kelly and Gibson (2006), the use of the PBL methodology encourages a deep understanding of the material, rather than surface learning, because it is the students who are actively 'doing'.

### Students' motivation

The majority of students who participated in this study already had a previous bad experience in learning programming. Some of them were in the third curricular year and had failed to obtain approval in programming courses; others did not like to learn it and because of that wanted to try a new form of learning. This reveals some dissatisfaction in the way that students learn how to program. Lethbridge, Diaz-Hererra, LeBlanc & Thompson (2007) refers that the number of students in computing disciplines have decreased in the USA since 2000, and points out some reasons for that, such as young people are so immersed in computers that they do not see the excitement to them anymore; and the stereotype of the 'nerd' coding all night with no social contact, making the students avoid these areas. With SL, we observed the opposite: students are not without social contact when programming and they are excited. Two events had impact on students' engagement: we had a student that received a proposal by an avatar in SL to buy his programming assignment, not as a violation of the student's conduct but as a sanctioned exchange of virtual commodities; another received a professional proposal to provide SL programming services for a company.

*The teaching process*
It is clear that the functions of a teacher are multifaceted as the teacher performs several functions when he or she teaches in the SL environment. His or her functions begin before the lectures commence as the teacher acts as an instructional designer, plans and prepares the lectures, facilitates the discourse and provides direct instruction when required. The teacher's function includes nontraditional activities such as preparing the virtual classroom or space, ie, defining areas in the classroom for each group of students to help the teacher identify what each group is doing on the project. Also, he or she has to prepare the classes' supplies and some visual materials, and program objects' behaviour so that novice students can observe and change these same objects. Thus, novice students may understand better the concepts being introduced. According to Miliszewska and Tan (2007), learning by examples is an excellent way for novices to overcome their difficulties. The teacher's work was also more intense and stressful than in a traditional class because of the need to include fine details in his or her advance preparation by writing out everything he or she wants to teach/say. Hence, the teacher had not only to predict the students' potential difficulties and the possible questions that could arise but also to prepare in advance the text of his or her response to those difficulties/questions in order to be able to provide quick feedback to the students.

In our research work, we concluded that the teacher's physical presence, in the first class, is important for the students to understand the SL interface as it facilitates the students' handling of the SL environment and program editor. Obviously, this was a result of the students' lack of familiarization with the SL interface, and possibly, may not be all that relevant if students have previous SL experience.

In this research, we also found that it would be useful to have a mechanism that could inform the teacher, by email or another outside system, about what students had done throughout the week, the difficulties they had felt and attempts to overcome them. Discovering the students' difficulties during the self-study would allow a better guidance from the teacher. A tight integration of SL-based activities with a learning management system (LMS) would be a possible path to this (eg, as proposed by Antunes, Morgado, Martins & Fonseca, 2008).

The overall framework for teaching and learning computer programming inside SL, based on the research summarized earlier, is presented in Table 2.

## Implications for learning and teaching

The results of this research have implications for both learning and teaching processes. The first implication is related to the benefits of using a proper project to motivate the students. It is well accepted that effective problems engage the students' interest and motivate them to probe for deeper understanding of the concepts being introduced (O'Kelly & Gibson, 2006). It is also known that the students' initial reactions to a subject or topic are critical to them gaining an interest (O'Kelly & Gibson). Consequently, it is important that the novice students are introduced to programming inside SL in an appropriate manner. The model developed in this research suggests the use of a project with a strong visual behaviour and adapted to students' prior level of knowledge. According to Poikela (2004), the nature of knowledge is contextual, as a resource and catalyst of learning. It is not only a conceptual, symbolic or formal fact, but it is embedded as potential in objects, artefacts, human activity or in the structure of an organisation (Poikela).

The model also highlights the use of PBL methodology. PBL gathers and integrates many elements regarded as essential in effective, high-quality learning, such as self-directed or autonomous learning, critical and reflective thinking skills, and the integration of disciplines (Poikela, 2004). Within PBL, the focus is shifted from teaching to learning, and this shift, in conjunction with a good project, provides each student with the freedom to think for themselves, activate their prior knowledge and acquire new knowledge in an explorative and creative way. The model further emphasizes the importance of teacher support to students' doubts. Thus, the teacher must be aware of those issues and provide a supportive feedback in writing proper comments on the students' code. It also encourages the students to explore the solution to their problems by themselves, engage in self-reflection and group reflection activities, and collaborate with their colleagues. As Dahlgren, Hult, Dahlgren, Segertad and Johansson (2005) refer, the interaction with peers is important for learning as one of the features put forward in many self-called student-centred pedagogical approaches within higher education. Furthermore, the use of small tutorial groups as the basic working form stresses the importance of interaction and communication for the learning process.

Anderson, Rourke, Archer and Garrison (2001) delineated three critical roles that a teacher performs in the process of creating an effective teaching presence in online learning. The first of these roles is the design and organisation of the learning experience that takes place both before the establishment of the learning community and during its operation. Second, teaching involves devising and implementing activities to encourage discourse between and among students, between the teacher and the student, and between individual students and groups of students and content resources (Anderson, 2002). The third, the teaching role goes beyond that of moderating the learning experiences when the teacher adds subject matter expertise through a variety of forms of direct instruction.

This research showed the importance for students' motivation the integration in a community which recognises their work. According to Wenger, Snyder and McDermott (2002), a community of practice is a good way to promote learning and good practices not only because it develops knowledge in a living and experimental way but also because it helps participants reach solutions to possible problems, with significant connections leading individuals to higher creative levels than they could reach on their own. Moreover, this study showed that the students' attitude for learning inside this environment was, in general, connected to their commitment in completing the project, making more attractive objects than their colleagues. They worked outside of the class time, collaborating for long hours with each other and with the teacher. As another important aspect, they considered the teacher as a work colleague with whom they could talk to, play and work. The most surprising and delightful aspect was observed when the students, by their own initiative, created other programs just for fun and had pride in showing what they had done. In our experience as teachers of computer programming, we have also observed this kind of behaviour in our students involved in traditional settings, but it typically is not widespread; ie, in traditional settings only a few students do this 'programming for fun', not the majority. As Twining (2009) referred, virtual worlds seem to provide the ideal vehicle for providing people with such 'lived experiences', of radically different models of education. They allow users to do things which would be difficult or impossible to do in the physical world. We acquired from this experience that it is important for learning that students study in a meaningful place for them, and where they can let their imagination fly. As Isaacson (2007) refers, '*A society's competitive advantage will come not from how well its schools teach the multiplication and periodic tables, but from how well they stimulate imagination and creativity.*'

This research provides a first step on how the teaching/learning of programming can be achieved inside SL. Thus, it contributes to the theory-deficient area of the use of virtual worlds (and SL in particular) in computer science higher education, and provides the first framework (see Table 2) for such use.

## Final thoughts and further research

A thought we expressed earlier was the importance of integrating virtual world activities with an LMS. We believe this integration could be even further than what we expressed earlier by allowing the teacher to follow the students' progresses/efforts and help them in a more effective and direct way, even when not inside the virtual world. To achieve this, it would be necessary to develop automatic mechanisms to track the students' progress within the activities taking place in SL, when the teacher is not present, possibly respond automatically to some queries (for instance, queries for further material for an assignment that is dependent on completing a previous step), and provide contact with the teacher through various media (for instance, using a system like the one described by Valério *et al*, 2009). However, that is a different development and research path. From the knowledge acquired with this research, we believe that the most relevant issue is that it is now possible to plan a research approach to find out if novice students learn computer programming better inside SL than through traditional classes. The reason for this is that it is now possible to base such a comparative research effort on specific teaching/learning environment—the one provided by the framework herein. This is not to say that it should be the only framework, not at all: we simply mean to state that a framework of coherent teaching/learning model is necessary for a comparison to be able to know beforehand what it is comparing. Specifically, it would be interesting to conduct such comparisons to verify not only the domain of programming techniques but also whether the acquired understanding is deeper, particularly by analysing if students can transfer their knowledge to other situations. Results from such research would also be significant to revise and improve this framework or devise novel frameworks for teaching/learning computer programming within virtual worlds. We intend to pursue such research efforts, and also to analyse other variables, such as the impact of learning in such environments on the students' motivation and—hopefully—improve the retention rates of first-year students in computer programming courses.

## References

Anderson, T. (2002). *Getting the mix right: an updated and theoretical rationale for interaction*. ITFORUM, Paper #63. Retrieved March 5, 2009, from http://it.coe.uga.edu/itforum/paper63/paper63.htm

Anderson, T. (2004). Teaching in an online learning context. In T. Anderson & F. Elloumi (Eds), *Theory and practice of online learning* (pp. 1–14). Athabasca: Athabasca University.

Anderson, T., Rourke, L., Archer, W. & Garrison, R. (2001). Assessing teaching presence in computer conferencing transcripts. *Journal of the Asynchronous Learning Network*, 5, 2. Retrieved March 5, 2009, from http://www.aln.org/publications/jaln/v5n2/v5n2_anderson.asp

Antunes, R., Morgado, L., Martins, P. & Fonseca, B. (2008). Managing 3D virtual classrooms. *Learning Technology*, *10*, 1, 3–5.

Barrett, T. (2005). Who said learning couldn't be enjoyable, playful and fun?—The voice of PBL students. In E. Poikela & S. Poikela (Eds), *PBL in context—bridging work and education* (pp. 159–175). Filand: Tampere University Press.

Bauman, Z. (1991). *Modernity and ambivalence*. Oxford: Blackwell Publishers Ltd.

Ben-Bassat Levy, R., Ben-Ari, M. & Uronen, P. (2003). The Jeliot 2000 program animation system. *Computers & Education*, *40*, 1–15.

Bergin, J., Stehlik, M., Roberts, J. & Pattis, R. (1997). *Karel++, a gentle introduction to the art of object-oriented programming*. New York: John Wiley & Sons.

Carlisle, M. C., Wilson, T. A., Humphries, J. W. & Hadfield, S. M. (2005). RAPTOR: a visual programming environment for teaching algorithmic problem solving. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, MO, February 23–27, 2005). SIGCSE '05 (pp. 176–180). New York: ACM.

Cooper, S., Dann, W. & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. In J. G. Meinke (Ed.), *Proceedings of the Fifth Annual CCSC Northeastern Conference on the Journal of Computing in Small Colleges*, Ramapo College of New Jersey, Mahwah, NJ (pp. 107–116). New York, NY, USA: Consortium for Computing Sciences in Colleges.

Dahlgren, M., Hult, H., Dahlgren, L., Segertad, H. & Johansson, K. (2005). The transition from higher education to work life, the outcomes of a PBL programme and a conventional programme. In E. Poikela & S. Poikela (Eds), *PBL in context—bridging work and education* (pp. 23–44). Filand: Tampere University Press.

van Dam, A. (2005). Visualization research problems in next-generation educational software. *IEEE Computer Graphics and Applications*, *25*, 5, 88–92.

Dann, W., Cooper, S. & Pausch, R. (2000). Making the connection: programming with animated small world. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE conference on Innovation and Technology in Computer Science Education*. Helsinki, Finland, July 11–13, 2000. ITiCSE '00 (pp. 41–44). New York: ACM Press.

Dann, W., Cooper, S. & Pausch, R. (2001). Using visualization to teach novices recursion. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*. SIGCSE Bull. *33*, 3 (Sep. 2001), (pp. 109–112). Canterbury: ACM. DOI=http://doi.acm.org/10.1145/507758.377507

Dershem, H. L. & Brummund, P. (1998). Tools for Web-based sorting animation. In *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*. Atlanta, GA, February 26–March 1, 1998.

Dick, B. (1999). *Rigour without numbers: the potential of dialectical processes as qualitative research tools* (3rd ed.). Brisbane, QLD: Interchange.

Dickey, M. D. (2003). Teaching in 3D: pedagogical affordances and constraints of 3D virtual worlds for synchronous distance learning. *Distance Education*, *24*, 1, 105–121.

Dillenbourg, P. (2000). *Learning in the new millennium: building new education strategies for schools*. Workshop on Virtual Learning Environments. Retrieved 27-Jun-2000, from http://tecfa.unige.ch/tecfa/publicat/dil-papers-2/Dil.7.5.18.pdf

Duch, B. (2001). Writing problems for deeper understanding. In B. Duch, S. E. Groh & D. E. Allen (Eds), *The power of problem-based learning: a practical 'how to' for teaching undergraduate courses in any discipline* (pp. 47–53). Sterling, VA: Stylus Publishing.

Esteves, M., Fonseca, B., Morgado, L. & Martins, P. (2008). Contextualization of programming learning: a virtual environment study. In *Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference*, October 22–25, 2008, Saratoga Springs, NY (pp. 17–22). Washington, DC: IEEE.

Esteves, M., Fonseca, B., Morgado, L. & Martins, P. (2009). Using Second Life for problem based learning in computer science programming. *Journal of Virtual Worlds Research*, *2*, 1. Retrieved 2009-04-08, from https://journals.tdl.org/jvwr/article/view/419/462

Esteves, M. & Mendes, A. (2004). A simulation tool to help learning of object oriented programming basics. In *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference* (pp. 20–23). Savannah, GA, October 2004.

Fleury, M. & Oliveira Junior, M. (2001). *Gestão do Conhecimento Estratégico—Integrando Aprendizagem*. São Paulo: Conhecimento e Competências. Editora Atlas.

de Freitas, S. & Neumann, T. (2009). The use of 'exploratory learning' for supporting immersive learning. *Computers & Education*, *52*, 2, 343–345.

Gomes, A., Areias, C. M., Henriques, J. & Mendes, A. (2008). Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa De Pedagogia*, *42*, 2, 161–179.

Isaacson, W. (2007). *Einstein: His Life and Universe*. Walter Isaacson (Edt.). Simon & Schuster Paperbacks, Rockefeller Center. New York.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of 3rd Annual LTSN_ICS Conference*, Loughborough University, UK, August 27–29, 2002 (pp. 53–58). York: The Higher Education Academy.

Kiili, K. (2005). Digital game-based learning: towards an experiential gaming model. *Internet and Higher Education*, *8*, 13–24.

Kölling, M., Quig, B., Patterson, A. & Rosenberg, J. (2003). The Blue J system and its pedagogy. *Journal of Computer Science Education*, *13*, 249–269.

Lahtinen, E., Mutka, K. A. & Jarvinen, H. M. (2005). A study of the difficulties of novice programmers. In *Proceedings of the 10th Annual SIGSCE Conference on Innovation and Technology in Computer Science Education (ITICSE 2005)*. Monte da Caparica, Portugal, June 27–29, 2005 (pp. 14–18). New York: ACM Press.

Lessard-Hébert, M., Goyette, G. & Boutin, G. (1990). *Recherche Qualitative: Fondements et Pratiques*. Montréal: Agence d'ARC.

Lethbridge, T. C., Diaz-Herrera, J., LeBlanc, R. J. & Thompson, J. B. (2007). Improving software practice through education: Challenges and future trends. In *2007 Future of Software Engineering*, May 23–25, 2007 (pp. 12–28). International Conference on Software Engineering. Washington, DC: IEEE Computer Society.

Michail, A. (1996). Teaching Binary Tree Algorithms through Visual Programming. In *Proceedings of the 1996 IEEE Symposium on Visual Languages (Vl'96)* (pp. 38–45). (September 03–06, 1996). IEEE Computer Society, Washington, DC, USA.

Miliszewska, I. & Tan, G. (2007). Befriending computer programming: a proposed approach to teaching introductory programming. *Journal of Issues in Informing Science & Information Technology*, *4*, 277–289.

Motil, J. & Epstein, D. (2000). *JJ: a language designed for beginners (less is more)*. Retrieved July 16, 2008, from http://www.ecs.csun.edu/jmotil/TeachingWithJJ.pdf

Newman, D., Goldman, S. V., Brienne, D., Jackson, I. & Magzamen, S. (1989). Computer mediation of collaborative science investigations. *Journal of Educational Computing Research*, 5, 2, 151–166.

O'Kelly, J. & Gibson, J. P. (2006). RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy, June 26–28, 2006) (pp. 217–221). ITICSE '06. New York: ACM.

Olapiriyakul, K. & Scher, J. M. (2006). A guide to establishing hybrid learning courses: employing information technology to create a new learning experience, and a case study. *The Internet and Higher Education*, *9*, 287–301.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1996). *The connected family: bridging the digital generation gap*. Atlanta, GA: Longstreet Press.

Pattis, R. (1981). *Karel the robot*. New York: Wiley.

Poikela, E. (2004). Developing criteria for knowing and learning at work: towards context-based assessment. *Journal of Workplace Learning*, *16*, 5, 267–274.

Rekkedal, T. (1983). The written assignments in correspondence education. Effects of reducing turn-around time. *Distance Education*, *4*, 231–250.

Robins, A., Rountree, J. & Rountreen, N. (2003). Learning and teaching programming: a review and discussion. *Computer Science Education*, *13*, 2, 137–172.

Salmon, G. (2009). The future of Second Life and learning. *British Journal of Educational Technology*, *40*, 3, 526–538.

Schulte, C. & Bennedsen, J. (2006). What do teachers teach in introductory programming? In *Proceedings of the Second International Workshop on Computing Education Research*, Canterbury, UK, September 9–10, 2006 (pp. 17–28). ICER '06. New York: ACM.

Soloway, E. M. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, *29*, 850–858.

Stasko, J. T., Domingue, J., Brown, M. & Price, B. (Eds) (1998). *Software visualization, programming as a multimedia experience*. Cambridge, MA: MIT Press.

Twining, P. (2009). Exploring the educational potential of virtual worlds—some reflections from the SPP. *British Journal of Educational Technology*, *40*, 3, 496–514.

Valério, S., Pereira, J., Morgado, L., Mestre, P., Serôdio, C. & Carvalho, F. (2009). Second Life information desk system using instant messaging and short messaging service technologies. In G. Rebolledo-Mendez, F. Liarokapis & S. Freitas (Eds), *IEEE First International Conference—Games and Virtual Worlds for Serious Applications*, Coventry, UK, March 23–24, 2009 (pp. 125–132). Los Alamitos, CA: IEEE Computer Society.

Warburton, S. (2009). Second Life in higher education: assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. *British Journal of Educational Technology*, *40*, 3, 414–426.

Wenger, E. (1998). *Communities of practice. Learning, meaning and identity*. Cambridge, UK: Cambridge University Press.

Wenger, E. C., Snyder, W. M. & McDermott, R. (2002). *Cultivating communities of practice: a practitioner's guide to building knowledge organizations*. Cambridge, Massachusetts, USA: Harvard Business School Press.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *SIGCSE Bulletin*, *28*, 17–22.

Zuber-Skerritt, O. (2002). A model for designing action learning and action research programs. *The Learning Organization*, *9*, 4, 143–149.