Teaching Programming Principles through a Game Engine

Cristiane Camilo Hernandez, Luciano Silva, Rafael Alencar Segura,
Juliano Schimiguel, Manuel Fernández Paradela Ledón,
Luis Naito Mendes Bezerra, Ismar Frango Silveira

Universidade Cruzeiro do Sul, São Paulo, Brazil
{cristiane.camilo, luciano.silva, rafael.segura, juliano.schimiguel, manuel.ledon,
luis.naito, ismar.silveira}@cruzeirodosul.edu.br

Abstract

Teaching fundamental programming to freshmen is a hard task to be performed, given the high level of abstraction and logical reasoning that are required for these students to develop in a very early stage of their course. This paper presents a discussion about this topic, along with a case study where computer games are meant to be developed by students in a first programming course, through the use of a game engine which allow students to develop fundamental programming skills without having to learn syntax and idiosyncrasies of any programming language.

Keywords: Teaching programming, Games, Game engine, Logic reasoning skills

1 Introduction

There is a large variety of factors that could be pointed out to justify the high rates of dropouts that are often observed in CS (Computer Sciences) and IT (Information Technology) courses' initial years. Some of these factors are related to freshmen's expectations about the course itself, which are far from CS courses' reality that demand, for instance, a solid Math background. However, another dropout cause could be related to those courses that are meant to teach the basis of programming logic, which demand from students higher abstraction levels than they are used to having, besides considerable skills for general problem solving. Tasks proposed by these courses normally are planned to make students develop cognitive abilities that normally had not been developed before. Such development often contrasts with the strictly curriculum-based macrostructure of K-12 and High School, often applied in undergraduate courses.

Many authors agree with these affirmations, stating that the process of learning programming is a slow and gradual task to be performed [6], besides the fact that today's programming languages - because of their complexity - do not seem to be adequate tools to teach programming foundations [14]. Many programming languages, such as Pascal - which was developed having as a main goal the teaching of programming, thus being simple structured, have evolved to languages that are intrinsically tied to more complex environments - such as Delphi. More than this, Jenkins [12] affirms that such languages usually have adequate syntax constructions for professional programmers, but not for initial learners. Pereira Jr. & Rapkiewickz [17] [18] present the difficulties students usually encounter on the assimilation of the abstract process of elaborating an algorithm, which frequently leads to many students' failures and dropouts, sometimes in a very early stage of an undergraduate course. The same authors point out that programming-related courses are responsible for the highest failure rates in Brazilian Universities within IT and CS-related courses, which demands a reflection and a paradigmatic change in didactical practice on these courses. Gomes et al. [10] shows a more detailed discussion about this issue.

There is another key problem to be addressed, which is the global phenomenon of the oscillating rate of enrollment in CS-related courses. This fact could be observed in following Figure 1 and 2.

Figure 1. Oscillating enrollment in CS-related courses at UCLA, with two major peaks [25].

Fig. 1 shows an evolutionary graphic of enrollment in CS-related courses at UCLA (University of California at Los Angeles), from 1971 to 2004, having two peaks clearly motivated by the advents of microcomputers (at the beginning of 80's) and spreading of Internet in the USA (end of 90's). It is reported that around 2004, less than 2% of UCLA's freshmen applied to Computer Sciences as their majors.

On the other hand, Fig. 2, extracted from [5] shows that US schools recently had recorded a slight increase of enrollment in their CS undergraduate courses, whose majors rose 8.1%, thus recording its first increase since 2000. This positive oscillation, however, has not yet been reflected in Brazil: Fig. 3, based on data presented by Simon [23], shows a notable reduction of applicant to vacancy ratio in Brazilian's FUVES (Fundação Universitária para o Vestibular – University Foundation for Entrance Exam) entrance exam for Computer Sciences at IME-USP (Instituto de Matemática e Estatística da Universidade de São Paulo – Math and Statistics Institute at São Paulo University), registering from 2001 to 2008 a decrease of 45.9 points.

Figure 2. Newly Declared CS/CE Undergraduate Majors in US universities

Figure 3. Reduction on applicant to vacancy ratio for CS' entrance exam at IME-USP (Brazil)

Regarding the whole scenario, Fig. 4, extracted from [15], shows freshmen enrollment in CS/IT-related courses in Brazil. In spite of the global growth of enrollments, it must be noted that enrollment in CS and CE (Computer

Engineering) courses oscillated slightly from 2001 and 2008; IS (Information Systems) undergraduate courses had a perceptible growth from 2001 and 2004, tending to face a stabilization from 2005 to 2008.

Figure 4. Freshmen enrollment in CS and IT-related courses in Brazil, extracted from [15].

The other courses mentioned in Fig. 4 are TDC (Teaching Degree in Computers), SC (Sequential Courses), and others. Still according to Fig. 4, growth in CS/IT area has been noted only in Short-Term Courses (STC), normally four or five semesters long. The number of enrollments is proportional to the total number of students graduating from this field, as seen in Fig. 5, recently increasing only in STCs.

Figure 5. Graduated students in CS and IT-related courses in Brazil, extracted from [15].

Fig. 6, based on [3], shows a graphic where it can be observed that, even though having more graduating students, this number is not even near to the forecasted demand for jobs in this area.

Figure 6. Graduated students from 2006 to 2011 (forecast) – careers' original names were kept from [3]: CS (Computer Sciences), IP (Information Processing) and EA (Electronics and Automation).

Besides the freshmen's decreasing interest to enroll in CS-related courses, the number of students actually enrolled is not enough to supply the abovementioned forecasted demand. Regarding Brazilian reality, according to [3,2] such demand varies from 100,000 to 130,000 new professionals per year.

Considering this scenario of oscillating – often decreasing – students' enrollment, high dropout rates and an insufficient number of graduated students, there is an urgent need for reflection about strategies to attract and retain students in CS-related undergraduate courses.

By using conventional didactic strategies for a first programming course, for instance, barely a “tangible product” would be obtained at the end of this course – often, the result of these courses is a raw program written in C or Java, very unrelated to the systems being used by students daily. There must be considered, in this context the new students' profile that universities have been facing in these first years of the 21st century: they are the first generation of “digital natives”, using the term coined by Prensky [20]. According to this author, the simple fact that these students have been educated in a highly digital context, immerse into technology since their childhood, influences the formation of different cognitive processes. Tapscott [24] names such generation as n-gen, or “net generation”, meaning those people that grew up using technological resources – mainly the Internet, in a largely natural manner, which changed the way these people deal with situations and with other people, making huge use of interaction aspects and collaborative work. Thus, the perception of these students' new profile leads to a reflection about new teaching strategies, more adequate to this new reality.

In this sense, there is a large variety of experiences using non-commercial environments and tools to teach programming [19], or even different organizational approaches [4]. Many of these strategies have a common point the effort to deliver to students quotidian situations, which could reflect students' realities, according to the principles of Meaningful Learning Theory [1]. In this way, this paper has as its main goal to present a game-based methodological strategy to teach programming by using a game engine. This sort of tool allows teachers and students to deal with programming logic fundamentals at a more concrete level, bringing students to an environment which could seem more familiar to them. This strategy could have some positive effects in diminishing students' dropout rates, at the same time that it is believed that such a strategy could help to motivate new students to enroll in CS-related careers.

2 Teaching Programming in CS and IT Courses: Previous Works

The teaching of programming skills is a fundamental part of CS and IT curricula. [13]. Programming, as an activity, permeates the other aspects of these courses and it is often a prerequisite for other disciplines.

Traditionally, the main goal of teaching computer programming is to promote the learning of a programming language according to the following model: teacher has a computer lab available for practical classes. However, according to Fontes & Silva [8], introductory programming courses usually follow pedagogical strategies since the 1990's, which are based in books or handouts that have the initial concepts in some programming language, when normally students learn how to make the classical program “Hello World” that shows the message “Hello World” on the computer screen. Programming languages make use of keywords in English, which often are not part of the students' quotidian, unless they had a previous formation in a technical or extension course. Many of them even confuse the “Hello World” message with the commands used to print the message itself – which denotes one of the difficulties related to algorithm and programming teaching [22].

Recently, a case study was developed for data structure and programming logic teaching in high school, as a way of motivating young students' interests in CS and IT [9]. This study was performed by presenting logic problems to students, who proposed algorithms in pseudo-language to solve them. After that, they implemented in C language.

Using student's quotidian elements in teaching and learning processes is important [11], according to Ausubel principles. In fact, the use of games as motivating elements in programming teaching has been proved to be an effective strategy, as is shown in [7],[21] and [16]. In this way, the next item presents the case study that was developed during the year 2009 in Cruzeiro do Sul University, located in São Paulo, Brazil.

3 Case study: a strategy for teaching basic programming concepts

The environment presented in this article involves the use of a game engine in the discipline of Computer Programming, which is a common introductory course for freshmen in the CSTs (Colleges of Technology) in Systems Analysis and Development, Internet Systems, Computer Networks, Game Development and Bachelor of Computer Science. The plan of teaching the subject in question provides as general objective to discuss and present

to students the concepts of: (i) fundamentals of computer logic and thinking, (ii) programming stages, (iii) language classification, (iv) programming paradigms and (v) visual programming. Initially, the proposal was not focused on any particular language, but the aim was to present programming techniques applied in various languages belonging to different paradigms such as Java, C, JavaScript, Prolog, Logo and Scheme. Then, several programming principles were put into practice by using the GameMaker engine. The GameMaker was chosen because it is a visual programming tool that allows us to create applications in a simple and efficient manner. Moreover, it is an engine geared to two-dimensional games that have elements of simpler interaction, appropriate to the profile of the students in question: this is the main reason that three-dimensional engines were not considered for this experiment. When working with Gamemaker, the student comes into contact with concepts and elements of programming, without having to code them into a programming language itself, simply by using interface elements.

The course was developed having an eminently practical approach, with all classes taught in laboratories. In introductory classes, GameMaker company's Web site developer (YoYo Games: <http://www.yoyogames.com>) was presented, where students could download the free lite version. The environment, basic concepts, the toolbar and some examples of games were presented, as shown in Fig 7, so that students are aware of the possibilities provided by the engine while being motivated through practical examples.

Figure 7: A basic game: Scrolling Shooter [16].

During the course, we presented new elements of the tool, using the game design document, which contains essential items for the process of level design as: objects, sounds, flow and level. From the knowledge of these elements, the student is able to build logical structures to control games without necessarily manipulating a programming language, or even having prior contact with the fundamentals of programming logic presented in the traditional manner. We used the strategy of gradual transition between the visual, always linked to games, and textual programming. The concepts treated in a first programming course included: setting and accessing variables, decision command and repetition.

The treatment of variables began with the use of predefined variables for the GameMaker environment: score, health and lives. These three variables define, respectively, the total points a player, a measure of living/damage and quantity of life. In order to introduce the concept of variable, we used the game The Scrolling Shooter [16], where experiments initially performed visual changes of these variables. As students gained experience in visually changing these variables, visual commands were gradually replaced by scripting commands, as shown in Figure 8:

Figure 8: Transformation of visual programming (change of a variable) into textual programming.

Then, we introduced the concept of declaration of variables. The environment GameMaker allows programmers to declare variables in the environment both visually and in text format. Typically, the use of the visual environment provides some initial facilities, in the sense that it is not necessary to specify the type in the declaration. In the textual environment, we can use the notion of type, syntax that resembles the C Language. Figure 9 illustrates an example of variable declaration in the visual and textual environments:

Figure 9: Declaration in both visual and textual environments.

Continuing the basic commands, we dealt with the conditional statement. This important concept of imperative programming can be tested with so much fun in a gaming environment: collisions between objects, winner of a game and the existence of a new phase are some examples presented in a systematic way to students. Normally, in other programming languages, these examples cannot easily be constructed because they depend on knowledge of graphics programming. Figure 10 shows an example of construction of the conditional statement, where you can notice the presence of blocks:

Figure 10: Example of a conditional statement.

Finally, the loop statement was covered. The GameMaker environment allows visual construction of repetitions in the style of the iteration statement, where is not necessary to define a counter, but only the number of times a particular block should be performed. In the final topic, it was adopted the same strategy above: gradual change of the visual environment to text.

Compared to previous semesters, the use of the strategy of gradual change of the visual programming environment to the text allowed students to develop very early, complex programming logic within a game without relying on the constant challenges imposed by textual programming languages. The view offered by the gaming environment has also allowed them to acquire skills to find logic errors based solely on visual inspection of game operation.

4 Final considerations

This experience allowed the initial presentation of the concepts of visual programming, event-oriented programming and object-oriented programming, without formalizations or specifications. Students start to build these concepts themselves – or at least they begin the knowledge construction about these subjects - through teacher-oriented controlled experimentation, they practiced in a simple development environment and in collaboration with their classmates. It is possible to conclude that the use of a simple, interactive and intuitive development tool like the GameMaker engine allows teachers to introduce to freshmen the basic principles of programming logic, without dealing with paradigms' idiosyncrasies or in programming languages' details of syntax.

The informal presentation of some essential concepts for future programming activities to freshmen – event-based programming, human-computer interaction and even object-oriented programming is particularly interesting. The proposal of using ludic elements in games created, designed and developed by students themselves, based on the exhibition of the previously mentioned theoretical elements, opens interesting possibilities regarding creativity, the practice of logical reasoning and the commitment of appropriate use of a language. Thus, based on this analysis, students' performance improvement in a posterior course on programming can be verified in the class of 2009, compared to 2008, when students did not have this course before having contact with real-world programming (Figure 11).

Figure 11: Performance graphic of the students in the years 2008 and 2009.

In the first semester of 2008, teachers used traditional methods- in other words, first programming concepts were presented to students through a programming language, namely Java. In the first semester of 2009 they started to use GameMaker to present these concepts in a different way. It is noticeable that in both aspects evaluated (percentage of delivered exercises and percentage of approved students) the performance of students was better in the class of 2009 than 2008. It can be inferred that this performance improvement could have been influenced by such methodological change. Nonetheless, a more detailed analysis is necessary to corroborate this hypothesis in future works.

References

- [1] Ausubel, D.P.: Educational Psychology: A Cognitive View. New York, Holt, Rinehart and Winston. (1968)
- [2] BRASSCOM: Brasil precisa capacitar 520 mil profissionais de TI até 2010. (2008). Available on the internet at <http://www.brasscom.com.br/brasscom/content/view/full/2375>. Accessed May 7, 2009.
- [3] BRASSCOM: Projeção de Alunos Concluintes nos cursos relacionados à área de TI. (2007). Available on the internet at <http://www.brasscom.org.br/brasscom/content/download/>. Accessed May 7, 2009.
- [4] Castro, T. H. C. et al.: Melhorando o Entendimento de Programação usando Esquemas Conceituais em Cursos Introdutórios. Anais do XXII CIESC – Congresso Iberoamericano de Educación Superior en Computación. Arequipa, Peru. (2004)
- [5] Computer Research Association. 2007-2008 Taulbee Survey. Available on the internet at: <http://archive.cra.org/statistics/survey/0708.pdf>. Accessed April, 5, 2010.
- [6] Dijkstra, E. W.: On the Cruelty of Really Teaching Computer Science. Manuscript (1988). Disponível em: <http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF>. Accessed in May 6, 2009.
- [7] Eagle, M.; Barnes, T.: Wu's Castle: Teaching Arrays and Loops in a Game. Proceedings of the 13th annual conference on Innovation and technology in computer science education. Madri, Espanha, p. 245-249. (2008)
- [8] Fontes, C.R.; Silva, F.W.O. da: O ensino da disciplina linguagem de programação em escolas técnicas. Ciências & Cognição 2008; Vol 13 (2): 84-98. Available on the internet at <http://www.cienciasecognicao.org>. Págs 84-98. (2008) Accessed May 9, 2009.
- [9] Garcia, R. E.; Correia, R. C. M.; Shimabukuro, M. H.: Ensino de Lógica de Programação e Estruturas de Dados para Alunos do Ensino Médio. Anais do WEI 2008, Belém. p. 246-248. (2008)
- [10] Gomes, A., Henriques, J., Mendes, A. J.: Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. Educação, Formação & Tecnologias; vol.1, n. 1, p. 93-103. (2005)
- [11] Guzdial, M.; Soloway, M.: Teaching the Nintendo Generation to Program. Communications of the ACM, vol. 45, n. 4, p. 17-21. (2002)
- [12] Jenkins, T.: On the difficulty of learning to program. Proceedings of 3rd Annual LTSN_ICS Conference. United Kingdom. The Higher Education Academy, p.53-58. (2002).
- [13] Ledón, M. F. P. ; Silveira, I. F. ; Bezerra, L. N. M.: Práticas multidisciplinares nas disciplinas da linha de programação no curso de Ciência da Computação. In: Jarmendia, A. M; Silveira, I. F.; Farias, L. A; Sparano, M.; Di Iório, P. L.; Domingues, S. F. S.. (Org.). Aprender na Prática - Experiências de Ensino e Aprendizagem. 1 ed. São Paulo: Edições Inteligentes, v. 1, p. 155-161. (2007)
- [14] Motil, J. e Epstein, D. JJ: A Language Designed for Beginners (Less is More).
- [15] Nunes, D. J. Estatísticas da Educação Superior em Cursos de computação – Censo 2008. Available on the internet at <http://www.sbc.org.br/index.php?language=1&subject=552&content=downloads&id=454>. Accessed April 9, 2010.
- [16] Pedroni, M.; Meyer, Bertrand.: The Inverted Curriculum in Practice. Proceedings of the 37th SIGCSE technical symposium on Computer science education. Houston, Texas, EUA. p. 481-485. (2006)
- [17] Pereira Jr., J. C. R. Rapkiewicz, C. E.: O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma Visão Crítica da Pesquisa no Brasil. Anais do WEI 2004 – Workshop de Educação em Computação – Congresso da Sociedade Brasileira de Computação. Rio das Ostras. (2004)
- [18] Pereira Jr., J. C. R. Rapkiewicz, C. E.: Um Ambiente Virtual para apoio a uma Metodologia para Ensino de Algoritmos e Programação. Revista Novas Tecnologias na Educação, vol. 3, n. 2, p. 1-7. (2005)
- [19] Pinheiro, M. C.: Uma Experiência no Ensino de Lógica de Programação para Cursos de Engenharia Usando o Pascal e o Logo. Anais do WEIMIG 2003 – Workshop de Educação em Informática de Minas Gerais. Poços de Caldas. (2003)
- [20] Prensky, M.: Digital natives, Digital immigrants. On the Horizon, v. 9, n. 5. (2001)
- [21] Rankin, Y.; Gooch, A.; Gooch, B.: The Impact of Game Design on Students' Interest in CS. Proceedings of the 3rd international conference on Game development in computer science education. Miami, Florida, EUA, p. 31-35. (2008)
- [22] Rapkiewicz, C. E.; Falkembach, G.; Seixas, L.; Rosa, N. S.; Cunha, V. V.; Klemann, M. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. RENOTE – Revista Novas Tecnologias Educação, v.4, n. 2, dezembro, Porto Alegre. (2006)
- [23] Simon, I. : A Relevância Social e Profissional da Computação no Início do Século 21 - Uma Reflexão Pessoal. Curso de Qualidade 2008 – SBC. (2008). Available on the internet at <http://www.ime.usp.br/~is/aula/belem-2008/rspc.pdf>. Accessed May 7, 2009.
- [24] Tapscott, D.: Growing Up Digital. The Rise of the Net Generation. Education and Information Technologies, v. 4, n. 2, p. 203-205. (1998)
- [25] Vegso, J.: Interest in CS as a Major Drops Among Incoming Freshmen. (2005). Disponível em <http://www.cra.org/CRN/articles/may05/vegso>. Accessed May 7, 2009.