Teaching Computer Programming : A Game Driven Approach

Paulo Rogério da Motta Junior Hamilton Athanazio de Lima Junior

EDS do Brasil, Rio Application Delivery Unit, Brazil

Abstract

On this paper we present two successful stories on applying game-driven programming as a tool to facilitate the teaching of computer programming. Given that computer programming is a complex task on its own, student motivation on the subject tends to decrease throughout the course. When using a game project as the final outcome of the discipline, users became motivated to work together as teams or individually to overcome difficulties in order to achieve the final goal of a complete and playable computer game. The study cases are presented together with our thoughts on how to proceed with this research in order to quantify the benefits that can be derived from this approach.

Keywords: oop, programming teaching, games, robocode, mobile games

Authors' contact:
{paulo.mottajunior,hamilton.limajunior}
@eds.com

1. Introduction

Computer programming is a challenging task on its own [Ulloa 1980]. Based upon the premise that an individual must study and practice different subjects to attain great skill and ensure a strong foundation in the programming profession, we can directly infer that *teaching* computer programming is not a simple and easy task.

Preparing students to become professional developers demands a great deal of hard work and adaptability from the instructor. When teaching specific programming languages there is a tendency to focus only on the language features instead of on the programming skills [Ulloa 1980]. Furthermore, we end up in need of a specific course so that students may be exposed to the trends of project reality.

Although offering a specific course on software development sounds perfectly reasonable, rarely, if ever, does this occur. Our best alternative is to use the programming language courses and develop small projects that can make the student experience more valuable.

Which leads to the question: What kind of projects should we choose to work on, that drives the students' attention and makes the experience of project development more interesting? The one thing that came to our attention is that, programming students, regardless of age, play and enjoy video games.

Our invitation then is to lead them through an incremental computer game development so that, at the end of the course, the student will have a fully operational computer game that is usable by others, instead of an abstract piece that creates a gap of understanding [deLaet 2005].

The remainder of this paper is organized as follows: Section 2, Related work that guided the organization of our own experiences; Section 3 The Motivation for the Research; Sections 4 and 5 Two successful stories on Applying Game Development as a Educational Tool; Section 6 Future Development; and finally on Section 7 Conclusions.

2. Related Work

We present two types of related work for this paper, the first is the inherent challenges associated with teaching programming languages, and the second is about the use of games for educational purposes.

Regarding the teaching of programming languages topic, we can see at [Lemos 1979] and [Ulloa 1980] the classification of the most common problems faced when teaching programming to students with little or no experience on the subject. Although now when we apply some of the techniques as described by the authors above we still face the same problems.

Regarding the use of computer games as a tool for increasing the students experience we can find at [deLaet 2005] a first approach for using games in Computer Science education.

3. Motivation

3.1 Student driven approach

Once students understand that they are part of the project and are consulted on the course's evolution, the synergy grows stronger and the teacher-student relationship evolves into a partnership.

3.2 Student's motivation

Although we have found some related work on the use of game construction as an aid for, teaching programming, there is still little experience on the topic

in Brazil. We find different courses on game development (for students that already do program), educational game development (on how to develop games that will help to teach other areas), and on the other hand game-aided courses that are not related to computer programming.

Following we describe two successful attempts to use the game programming approach. The first teaches students that already had an introductory course on programming how to program in the Java language and the second, teaches students with Java programming experience how to move to the mobile environment. On both cases the game took the place of the traditional information system application that is commonly used.

4. Teaching OOP Using Games

4.1 Programming Game Concept

There is a common perception that games are "toys" [Lee 2004], but when we begin to touch upon this arena, we found that we are talking about a huge industry [GameIndustry 2006]. Defining a game as the final product of the OOP teaching is one of the strategies but an alternative is to use a programming game, essentially it is a game that is not playable by itself.

The main goal of the programming games is to invite the student to be a part of the game by adding a piece of code in it that enables the game to be played.

The used terminology of "programming game", could be easily interchanged with the term "environment", however, this interchange could create a misunderstanding by identifying the programming games as *regular educational games*, whereby the user interacts only with the *existent* game elements and interfaces, and is not required to implement the playable piece of the game.

Game Engines are the other elements in this game scenario that can used to avoid the implementation of common game coding routines, as collision detection, map creation, sound manipulation, etc.

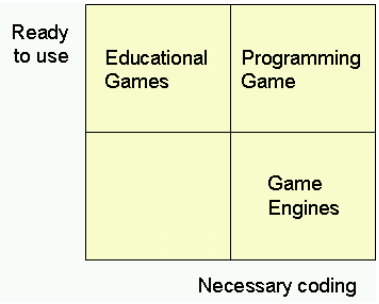

Figure 1: game creation resources

4.2 The Programming Game Used – Robocode

The Robocode offers an API to create virtual robots, an environment to create an arena where the robots fight against each other, which also offers a very simple IDE to create the Robots using the Java language.

The API offers the class Robot where all of the commands are executed in a blocked mode – the first command to the robot must wait until the first one finishes and the Advanced Robot class where the robot implements a pool of commands which run in a non-blocked mode.

Soft curve movements are only available in the Advanced Robot class where the "turn tank" and "go ahead" movements can be combined in a curve. However, Simple movements such as "turn left" , "go ahead" and "stairclimbing-like" movements are offered in the Robot class.

More than just movement commands, there are also event handling constructs, including onGetHitByBullet that is triggered when the robot is hit by a bullet and onScannedRobot which is triggered when the radar identifies other robots in its field of vision.

4.3 The Experiment

The choice for the Robocode to be used with the students, happened in an empirical method. After some use with the students, the "balance of challenge" [Carswell 1996] seems to be correct, as observed students creating very rudimentary robots as well as some sophisticated robots.

Since the first semester of 2003 at the Unicarioca University (Rio de Janeiro-Brazil) during the Java Programming discipline, the Robocode approach has been used with the students.

The main purpose is to increase the self-confidence of the students by adding an unknown API with a short training session[1] to demonstrate their ability to make it work and enforce that most of the programming activities are approached by applying the concept of "reuse" of an unknown API.

Some competitions occurred during the weekend while others happened during a class day, depending on the quorum. The robots were made individually or sometimes by team participation. To increase the challenge and promote the value of code sharing, the sessions were divided into two acts: The first act, where the teams code their robots (with around 60

[1] The students receive 30 minutes of instruction one week before the competition and are asked to study by themselves before the competition

minute time limit) and then submit the 'ready-to-use' robot to a server where the others competitors had access to it. In the second act, the teams are challenged to test their own robots against the other participants' robots and make the final adjustments needed on their own robots. After the test comparison, the students submit their final version. We make a simple competition by adding four robots to the arena then taking the first and the second robot to the next phase and repeating until we have only one robot which is the winner of the competition.

Sometimes at the end of the competition, which has *always* resulted in a fun environment, we created a "jam session" whereby all of the robots were inserted into the arena.

5. Mobile Games for Java Students

Given the growing interest for mobile computing, many Computer Science courses are adding some type of mobility interaction as optional disciplines. Some of these disciplines have time limitations of approximately 32-hours. Time constraints brings to the table the question: How can we overcome the inherent complexity of the programming paradigm shifting to a time-restricted course? The answer came as the use of game development could help students to focus on an interesting task and bringing them together to work as teams.

Since the J2ME course was included on the Technical Computing Course of Fundação Bradesco[2] (Rio de Janeiro – Brazil), and students were already familiar with simple programming projects in the Java programming language, bringing the students together became the challenge so that we could focus on the task at hand and develop the course content.

The approach was to divide the class into seven teams of five students each. Each student had a well-defined role either as a programmer, designer, tester, manager or documentation writer. The projects were proposed by the teams and all communication would only take place through the manager with the teacher. Every team member had to participate on design and programming activities and to their other assigned roles, as well. Using this structure, we could exercise a real-life project in a small controlled environment as in [Jones 2000].

The first task was to present the project to the teacher to obtain sponsorship. Students were asked the question, "Would you buy this project?" Which then led the students to reformulate the project definitions and presentations. Besides the project implementation, theoretical concepts were presented in class both on J2ME and game development. Since the students were already familiar with programming, the approach used was the "Game Library", as described earlier in this paper, where they received a base working game with a simple library that allowed collision detection and game timing through a simple thread mechanism.

The students had little or no experience with threading and this was revealed as a critical path element because it was not specified as a prerequisite or included as part of the scope.

At some point in time, near the middle of the course, all the basic topics were covered and the remaining time was devoted towards implementing the projects. This was a major change on the student's perception since until that time they were guided to implement only small programs. For the first time they were confronted with a real project and with the associated doubts and uncertainties. At this point, the manager-only communication was dropped and communication started to take place mostly with lead programmers in order to check the code. The teacher's role changed from one of sponsorship to technical leader, basically helping the programmers find the solutions for their problems.

One clear observation was that as soon as the concepts were finished and only the coding was left the students started working diligently. The team synergy exhibited during the meetings was a new ingredient that wasn't evident before. All team members were working together to achieve the defined project- goals.

The results observed are enumerated as follows:

1. Motivation was improved – using a subject that is current in their life helped to decrease the distance between teacher and students. Most of the time we try to bring them out of their world and ask them to think on Information Systems of which they have little or no knowledge.

2. Self esteem and confidence developed – facing a real life challenge helped to increase the student's morale by giving them the tools necessary to develop software that could be shared and explained even to non-computer friends [deLaet 2005].

3. Programming skills improved – the use of new and challenging features like threading and facing a real project lifecycle exposed the students to a group of situations demanded them to improve their programming skills. Although we observed a great difference in the complexity of the games, all of the groups programming skills improved.

As a final note the games ranged from quiz-based adventures to action shooting games. Since the games weren't intended for sale, we did not focus on

[2] Fundação Bradesco gives a post high school computing course at technical level.

performance and size metrics. We believe that these metrics should be studied in a game development discipline that focuses, specifically, on the issues that arise in the game industry.

6. Future Development

Since this is a work in progress, we are planning next steps on how to measure the results in a more pragmatic way. These first attempts prove that 1) this is a path that must be explored and 2) the use of game developing does improve students' learning experience. We are now facing an issue as to how and what to measure in order to quantify the benefit of using this approach on a regular basis.

Specifically, for the Robocode approach, after the identification of the quality impact metrics by use of this game framework, we plan to create another game framework that deals with a non-violent subject, and establish a measure to compare the results. The proposal of this new game framework is draft version and can be reached at [JSoccer 2005].

For the mobile computing the development of a richer library that will empower students with more functionality could be a good approach, but it is not quite clear on how to measure the students evolution since this is a much more flexible environment.

7. Conclusion

We conclude the use of games as tools for motivating students on a complex subject like computer programming has many benefits. Not only in the development of programming skills, but also by bringing together many different aspects that are learned, isolated , throughout the Computer Science program of Universities [Jones 2000]. When motivating students, classes become easier to conduct and the content is better understood.

Another important point is that it is not necessary to be a part of the game industry to guide students through the development of good and appreciable games (not necessarily *simple*) the level of sophistication depends mostly on the effort employed by students in learning more sophisticated programming techniques for their games.

References

CARSWELL, L. and Benyon, D. 1996. An adventure game approach to multimedia distance education. In *Proceedings of the 1st Conference on integrating Technology into Computer Science Education* (Barcelona, Spain, June 02 - 06, 1996). ITiCSE '96. ACM Press, New York, NY, 122-124.

DELAET, M., Sweedyk, E., Slattery, M. C., and Kuffner, J. 2005. Computer games and CS education: why and how. *SIGCSE Bull.* 37, 1 (Feb. 2005), 256-257.

GAMEINDUSTRY, available at http://www.gamesindustry.biz, August 2006

HOLLIDAY, M. A. 1995. Incremental game development in an introductory programming course. In *Proceedings of the 33rd Annual on Southeast Regional Conference* (Clemson, South Carolina, March 17 - 18, 1995). ACM-SE 33. ACM Press, New York, NY, 170-175.

JONES, R. M. 2000. Design and implementation of computer games: a capstone course for undergraduate computer science education. In *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education* (Austin, Texas, United States, March 07 - 12, 2000). S. Haller, Ed. SIGCSE '00. ACM Press, New York, NY, 260-264.

JSOCCER, available at http://jsoccer.dev.java.net, August 2005

LEE, J., Luchini, K., Michael, B., Norris, C., and Soloway, E. 2004. More than just fun and games: assessing the value of educational video games in the classroom. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM Press, New York, NY, 1375-1378.

LEMOS, R. S. 1979. Teaching programming languages: A survey of approaches. In *Proceedings of the Tenth SIGCSE Technical Symposium on Computer Science Education* SIGCSE '79. ACM Press, New York, NY, 174-181.

ROBOCODE, available at http://robocode.sourceforge.net/, August 2006

ULLOA, M. 1980. Teaching and learning computer programming: a survey of student problems, teaching methods, and automated instructional tools. *SIGCSE Bull.* 12, 2 (Jul. 1980), 48-64.