Enhancing Computer Engineering Courses with Game Development

Artur H. Alves*, Carlos A. Ynoguti*, Daniel S. Neves*, Pedro J. Matuck*

* Computer Graphics, Games and Mobile Apps Lab, National Telecommunications Institute, Santa Rita do Sapucaí, Minas Gerais, Brazil

Emails: artur.alves@gec.inatel.br; ynoguti@inatel.br; danielsader@gec.inatel.br; pjmatuck@inatel.br

Abstract

The course "Introduction to Game Development using Construct 2" has as its main goal offering an opportunity of stepping into the wide area of game development, with a hands-on approach to game design, logic and arts. It also gives the Computer Engineering student the chance of seeing real applications of diverse courses they will take in college, e.g. algorithms, multimedia systems and computer graphics. In a first experience, this environment showed to be very attracting for both students and professors, and this paper describes in depth the implantation, the execution and the first results of this course.

Keywords: game development; project-based learning; engineering education.

1 Introduction

Computer games are a very attractive recreational activity, especially among young people (Overmars, 2004). Also, many of the features of the generation that is entering university today had origins in the way people interact with these games: need for immediate feedback, anxiety, multitask behaviour, etc. (Oliveira, 2011).

Therefore, it is natural to think that games can be used within higher education in various ways to promote student participation, enable variation in how lectures are taught, and improve student interest (Wang & Wu, 2009) (Papastergiou, 2009).

Actually, they were and are being used to make students learn in a more attractive way. Some initiatives include game based learning (Jong et al, 2013), the use of physical joysticks to implement I/O subsystems (Larraza-Mendiluze et al, 2013) and the use of selected video games for student motivation (Gonzalez et al, 2013), among many others.

A different approach would be to use the computer game development scenario as a pedagogical activity to motivate and engage students in curriculum-related activities (Ownston et al, 2009). Two points in favour of this approach are the introduction of "fun factor" needed to engage students, and the practical realism of a real engineering project (Kajal, 2005).

This activity can include topics in algorithms, computer graphics, artificial intelligence, human-computer interaction, security, distributed programming, simulation, software engineering, among others (Overmars, 2004), and improve students' abilities in four important areas: (1) problem solving; (2) the application of previously learned knowledge; (3) the use of independent learning; and (4) learning by doing (Cagiltay, 2007).

With these goals in mind, in this article, a freshmen-directed, Project Based Learning, Game Development course is described. Its main goals, in addition to the above ones, are to engage Computer Engineering students, to reduce truancy and to integrate in a significant way the contents learned so far. The results show that this could be a possible means to make them learn some concepts and acquire some soft skills needed for their future professional life.

The remaining of this article is organized as follows: in section 2, the motivations and goals for this work are given. Some basic information about game development are given in section 3. Section 4 describes the course activities in detail, and finally, the conclusions are presented in section 5.

2 Motivations and Goals

The main reason to implement this course is to motivate our Computer Engineering students. Besides, there are some more motivations listed below.

2.1 Interdisciplinarity

Engineering courses, by definition, focus heavily in Math courses such as Calculus, Algebra and Physics. Although one can correlate them, the traditional education methods treat these topics as isolated steps of graduation, with little connection between them. One of the motivations for this course is to stimulate the integration between these subjects and use this integrated knowledge to plan and build a game project from scratch.

We observed this integration mostly when explaining the Physics component of the software. To launch a projectile, for instance, one first needs to create the projectile's sprite (a visual representation of the object), then set the object's physical properties (static drag, air drag, elasticity) and then finally add the force vector needed to make it fly. This simple activity demands knowledge on Physics ("What is drag?", "How does it affect the object's movement?"), Algebra ("What is a vector?", "How do I represent a vector's components?") and Trigonometry ("How far does the projectile go when the angle is 45º?", "What happens to the projectile's range if the angle is larger/smaller than 45º?").

2.2 Better understanding of common core concepts

Another problem faced by students when learning these subjects is the difficulty to visualize some concepts, like vectors and wave properties (frequency, period, amplitude) in a simple and concise way. It is hard to represent concepts like wave movement or the change of speed of an object with traditional, static drawings. The software used allows the students to see dynamic graphical representations of these topics and freely modify the properties of the objects, giving almost instant feedback on the changes made. For instance, the variation of the speed of a body can be represented as a car braking, where the parameters (weight, initial velocity, dynamic drag) can be easily modified. When the game is executed, it automatically reflects these new parameters with the car behaving differently according to their values. This fact was also observed by (Leutenegger and Edgington, 2007): "the visual component allows students to see mistakes in their code as manifested in the resultant graphics."

2.3 Game development as a goal

Throughout a few months of working at the Computer Graphics, Games and Mobile Applications Lab at Inatel, we detected a widespread but not stimulated interest in game development in other students. Most of them did not know where to begin, or what abilities they should improve. Since we were also studying game development with various tools, we thought it to be a good idea to spread the knowledge we were acquiring.

2.4 Uninterested students

A problem often perceived by professors is that students cannot understand why they should study what they study. Sometimes, the traditional way of teaching does not make it clear. For Computer Engineering, one way to show students how they might use some of the knowledge they are given during the graduation is to teach them how to make games. Game development is not only programming; it can include calculus, physics, algebra, CG, AI, etc. Toghether with the perceived interest of students in game development, the idea of a game development course started to gain importance. Considering the problem of uninterested students, we needed to find a more dynamic way of teaching.

3 Game Development

Game development is a multidisciplinary area intended to yield fun and information using computing concepts to create an immersive environment, combining programming, arts and design, the pillars of game development (Rabin, 2012). Each pillar is an extensive area of study by itself.

A game can have many definitions depending on the context in which it is applied. In computing, the definition of games is the combination of multimedia resources that seeks to establish a meaning based on rules, goals and rewards (Rabin, 2012).

In most cases, the resources used in game creation require mathematical, visual and physical representations of their behaviour. Therefore, the developer must know concepts of algebra, geometry, physics, algorithms, and game design to create a coherent result.

During the development of a game, the creator needs to think about various issues, mainly how the player will react to one or more objects on the screen and if this reaction will lead him or not to the proposed goal. This environment is composed primarily of a scenario, where the rules are applied; characters, including the player themselves; elements that assist or hinder the player in the pursuit of goals; and objects, elements that compose the scenario requiring or not interaction with characters (Rabin, 2012). In order to create a meaningful result, these components must be tightly integrated, which usually requires the establishment of a workflow.

The result is presented to the player through input and output interfaces: a controller, the way that input commands are sent to the game, and a display, where the game reacts to the inputs. The interfaces are closely related to the platform receiving the game (Rabin, 2012). There are several digital platforms for games, such as consoles, web and mobile devices. Each platform provides different interface, allowing different uses and combinations of resources.

4 Course Description

4.1 Classes

The course was comprised of 100-minute classes, happening once a week, for 14 weeks. Each class was focused on a single topic related to game development. Short presentations were prepared for each of those topics, and presented and explained at the start of each class. Those covered the basics of the current theme: what it is, how it is developed, how it ties in with the engineering course and how it is applied in the software.

Specific details were deliberately omitted to encourage students to research about them. One example was the Particle System object. A particle system is a graphic technique used to simulate gaseous substances like smoke, water and fire. This technique comprises of using a large number of small images (called Particles) to simulate those behaviours. The basic instructions given by the instructors showed the students the bare minimum needed to have a working particle system on the screen, but all the optional parameters were left for the students to explore by themselves.

Sometimes, entire functions were not explained, like the Sine component, which can be used to change the position/scale/rotation of an element on the screen based on a sinusoidal function. Links and documentation about the subject were given to the students, to help them on their research, but they were expected to learn by themselves. Many of the student-created projects relied on this behaviour, indicating that the students learned and mastered that function without the teachers' direct intervention.

The practical section of the classes involved a very short base game with some functionalities missing, e.g. a spaceship game without the shooting logic, or a race game without the car controls. The students were asked to fill in the missing functionality using the concepts learned at the current and previous classes. The instructors were available at this time for any eventual questions or problems.

The students were also granted creative freedom, being free to use any images/sounds they wanted, as long as they respected all, if any, copyright licenses. Most of the art assets used on the base games were licensed with the Creative Commons (Creative Commons, 2014) license, which allows the users to edit, adapt and create using those assets, as long as they credit the original creator of the content on the game's credits. The students were warned any time they used copyrighted content, and asked to replace the infringing content with an appropriately licensed one.

4.2 Software

Scirra's Construct 2 (Construct 2, 2014) is a game development tool mostly known by its ease-of-use, mainly for simplifying the programming side of game development by abstracting the use of a programming language. Instead, users must develop the game logic by putting together logical blocks. This makes this tool a great asset for teaching freshmen not only game development but also programming logic.

4.3 Assessment

One of the most important challenges in this activity is the elaboration of a simple yet broad assessment method. It had to offer specific criteria for each of the topics approached in each lesson, while being easy to grade and to understand. Therefore, we settled for a three-grade rubric to be adapted for each lesson with all subjects relative to it. Figure 1 shows an example of a rubric, used in an activity where students were asked to research about the use of Physics and Algebra in games, create a one-page report on the research and, as a bonus, to create an example in Construct showing concepts they had written about. Notice that an A grade corresponds to 100 points, a B grade to 50 points and a C grade to 0 points.

	Grades			FINAL GRADE
Subject	C = 0	B = 50	A = 100	
Algebra				
Physics				
Authenticity				
Grammar and Ortography				
BONUS: example in Construct				

Legend	Observations
A **Reached/exceeded expectations**	
B **Reasonably developed on the subject**	
C **Unsufficiently performed**	

Figure 1: Three-grade assessment spreadsheet

The next challenge was to decide how to describe the topics from each lesson in an objective way, so we could evaluate them as correctly as possible and so the students would always know how we were testing their learning. That is far from easy in game development, because one must account not only for if the game works or not, but also for the creativity and effort applied to the project. We thought the best way was to split the technical topics in subjects related to some of the graduation courses, so that the students could better relate their work with each subject. Likewise, it would be more natural for us to evaluate these points because of our familiarity with them, from our own experience in graduation.

That still left some subjective topics, though, like the aforementioned creativity. To mitigate this problem, one approach we thought interesting was to offer them some artistic assets from public domain resources like Opengameart (Opengameart, 2014) and Freesound (Freesound, 2014). Even though, these topics are subjective; finally, we decided to give them a smaller weight in the final assessment process, so we would still stimulate the students to improve their work without turning the assessment process into a subjective effort.

One major communication channel between instructors and students was Dropbox (Dropbox, 2014), a file-sharing platform with collaborative tools, available for free download. At every class that would have either a slide presentation, a specific resource or an assessment spreadsheet, we shared these files with all the students, so they could fully participate in the activities while already knowing how we would evaluate them.

Though our main assessment tool was the rubrics, not all activities were fit for it. Therefore, we split them into three main assessment levels:

- **Participation-only:** Some of the simpler activities, such as ones developed with the instructors, were assessed on a participation level: the student would guarantee their grade for that day by interacting with other students and the instructors, asking questions and developing whatever activity they received in class. These activities usually applied the concepts taught in class in a low level, asking the students to apply them in new ways.
- **Normal activities:** More complex activities were commonly left as challenges for the students to develop outside class. These activities were to be turned in by next class, when they would be assessed using a specific grade rubrics previously available to them. These activities asked the students not only to apply the learned concepts in new ways but also to research new tools and to understand a presented problem.
- **Main projects:** At the middle and at the end of the course, the students were presented a wider problem: they were asked to develop a game following some basic characteristics, like perspective and genre. What concepts, tools and resources they would use was their choice. This, of course, leads to a bigger challenge in assessment, so we asked them, through a rubric, to present some specific results (e.g. the game must use at least two physics concepts taught throughout the course).

4.4 Results

We offered 24 places to students in our college that had already had the introductory Algorithms course. Fifty students showed interest, but we had to choose the first 24 to attend. At the end of the course, two students had formally quit and another ten had given up at various points. The two who formally quit presented us with the same reason: they had more important appointments in college, and were not feeling able to cope with everything. Another three students did not present the last project because of college exams, while the other seven did not give us any reason. Speculating on this, we think some of the most probable reasons include lack of interest in the course after seeing how it approached the proposed subjects and unwillingness to commit to a new responsibility.

On the last day of the course, the students were asked to present and explain their game to the teachers and, optionally, to the whole class. There was a large variety of games, ranging from remakes of classic games to new concepts. While most of the games used free/open-source assets, others had completely original assets, with all the art created by the students. Because the games were exported to HTML5, they could run on any device equipped with a modern web browser, such as mobile phones and tablets, as well as desktops and notebooks. After the course, some students even started creating new projects with their own original ideas.

By then we could see that they were applying basic Physics and Algebra concepts on their games without difficulties. The students applied gravity, sinusoidal movement, impulse, angular movement and other things naturally to the projects, as they saw fit.

Considering the twelve finishing students, their mean grade was 76 out of 100. The final grade weights the participation-only and normal activities at 40% and the main projects at 60%. We noticed that the students usually have better performance in practical activities like the main projects, while only a few of them show effort on theoretical works, such as researches. In our first normal activity, we asked them to research about Physics and Algebra in games; only five of the then 24 students presented original results, with the others simply copying and pasting from the Internet. Because of this, we switched our efforts to practical activities for the rest of the course.

After the course, we asked the students to rate a few sentences about what they experienced, from 1 ('completely agree') to 5 ('completely disagree'):

- The activity enhanced my programming skills. (2.5)
- The activity helped me comprehend the connection between Engineering courses. (1.75)
- The activity motivated me to continue my Engineering studies. (1.625)
- The activity stimulated me to study more about game development. (1.25)

We expected the results about programming skills to be lower, since the software used does not demand much programming expertise. In a next iteration, we may choose another software suite with a steeper programming skill curve, to better accommodate the Computer Engineering skill set.

4.4.1 A comment about generalization

On an isolated day, a technical problem made the Construct 2 software unavailable for class. The only other available game development software installed on the computers was Microsoft's XNA Game Studio (XNA, 2014), a more advanced and powerful tool for game development that utilizes the C# (Visual C#, 2014) programming language as its main language. Thus, the teachers had to adapt the class of that day to the XNA Framework workflow. Even though XNA is much less intuitive than Construct 2, because they share the same basic mathematical concepts the students were able to use most of the Construct 2 logic created on the earlier classes in the new tool. Although the game created that day was simpler than the previous ones, they were fully functional.

5 Conclusions

In this paper an optional, one semester computer game design course for freshmen Computer Engineering students is proposed. In this course, they would not only learn the basics of game design and programming logic but also see some of the concepts they were supposed to learn in traditional graduation courses in real-life situations. This was possible by the use of project-based learning techniques and an easy to understand assessment tool.

The course was given throughout almost four months to a starting class of 24 students. At the end, 12 students successfully completed all the proposed activities and projects. The general response to the course was positive; all students who finished it showed interest in the next one, even though we had not even hinted at a continuation. At the end, they had developed two major projects that applied some key concepts learned at basic college courses, including Physics and Algebra, by themselves.

We stimulated students to dedicate their time and effort to learn contents important to their Engineering course while developing other skills that are interesting to them and may be of great importance in their professional career. This analysis also helped us identify some major points that need development in our next course, while giving us confidence in our results.

The most significant difficulties we identified while planning and executing this experiment are stimulating and rewarding students for subjective topics as creativity and artistic prowess and reaching out to students in a more dynamic way. For our next course, we propose some improvements:

- **Four-grade assessment spreadsheets:** the three-grade system used in this experiment gives us a reasonable overview of the students' progress and interest, but it does not perform well in depth, especially in the lower spectrum. Therefore, we will use a four-grade system to give us more details on student performance while maintaining the ease of understanding we were aiming from the beginning.
- **Definite weights for each assessment:** as this was our first experiment in such a course, we felt the need to adapt our assessment weights during the semester. A downside to that was that students not always knew how their performance in one activity would affect their final grade. Now, with these results in hand, we can define these weights more confidently; we expect this to better stimulate our next students.
- **Previous alignment with professors:** we focused on the common core topics that matter the most for game development. To seize better this opportunity, however, we may try asking professors correlated to the areas what are the most important subjects that students show difficulties to learn, so we can focus our efforts in developing more effective activities for future lessons.
- **Bigger focus on PBL:** even though the projects were developed in pairs, we want to plan the next course to stimulate group effort and critical thinking throughout all lessons. Some ways we already

thought of to achieve this goal include increasing the number of main projects (from two to three, at first) as well as connecting the activities developed throughout various topics.

- **Better integration of theoretical and practical content:** we noted a resilience by the students towards theoretical activities, such as researches. For our integration with common core subjects and Engineering in general, though, we need to help the students get used to researching. Therefore, we want to plan our home activities in a way that they need to research more than they had to in this first experiment, while maintaining the entertaining factor of game development. This integrates heavily with our desire to increase the use of PBL, as one way to instigate them to research is to give them more elaborate projects to develop.

The results from the final poll show us that, in average, the students felt really stimulated to continue making games and, most importantly, that they understood the connection between different Engineering topics in a practical project. This further comprehension of interdisciplinarity is positive not only to their possible future as game developers, but also as Engineering students and professionals in general.

References

Cagiltay, Nergiz Ercil. (2007) Teaching software engineering by means of computer-game development: Challenges and opportunities. British Journal of Educational Technology, 38(3), 1467-8535.

Creative Commons (n.d.). Retrieved March 1, 2014, from http://commonly.cc/

Gonzalez, J.; Pomares, P.; Damas, M.; Garcia-Sanchez, P.; Rodriguez-Alvarez, M.; Palomares, J.M., (2013). The Use of Video-Gaming Devices as a Motivation for Learning Embedded Systems Programming. IEEE Transactions on Education, 56(2), 199-207.

Jong, Bin-Shyan; Lai, Chien-Hung; Hsia, Yen-Teh; Lin, Tsong-Wuu; Lu, Cheng-Yu, (2013). Using Game-Based Cooperative Learning to Improve Learning Motivation: A Study of Online Game Use in an Operating Systems Course. IEEE Transactions on Education, 56(2), 183-190.

Claypool, Kajal and Claypool, Mark. (2005). Teaching software engineering through game design. In Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05). ACM, New York, NY, USA, 123-127.

Construct 2 (n.d.). Retrieved March 1, 2014, from https://www.scirra.com/construct2

Dropbox (n.d.). Retrieved March 1, 2014, from http://www.dropbox.com/

FreeSound (n.d.). Retrieved March 1, 2014, from https://www.freesound.org/

Larraza-Mendiluze, E.; Garay-Vitoria, N.; Martin, J.I.; Muguerza, J.; Ruiz-Vazquez, T.; Soraluze, I.; Lukas, J.F.; Santiago, K. (2013). Game-Console-Based Projects for Learning the Computer Input/Output Subsystem. IEEE Transactions on Education, 56(4), 453-458.

Leutenegger, Scott and Edgington, Jeffrey (2007). A games first approach to teaching introductory programming. In Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07). ACM, New York, NY, USA, 115-118.

Rabin, Steve (2012). Introdução ao Desenvolvimento de Games vol. 1 – Entendendo o universo dos jogos. São Paulo: Cengage Learning.

Oliveira, Sidney (2010). Geração Y - O Nascimento de uma Nova Versão de Líderes. São Paulo: Integrare.

OpenGameArt (n.d.). Retrieved March 1, 2014, from http://opengameart.org/

Overmars, M., (2004) Teaching computer science through game design, Computer, 37(4), 81-83.

Owston, Ron, Wideman, Herb, Ronda, Natalia Sinitskaya, Brown, Christine (2009). Computer game development as a literacy activity. Computers & Education, 53(3), 977-989.

Papastergiou, Marina (2009). Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation. Computers & Education, 52(1), 1-12.

Visual C# Resources (n.d.). Retrieved March 1, 2014, from http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx

Wang, Alf Inge and Wu, Bian. (2009). An application of a game development framework in higher education. Int. J. Comput. Games Technol.

XNA Developer Center (n.d.). Retrieved March 1, 2014, from http://msdn.microsoft.com/en-us/aa937791.aspx