A Video Games Technologies Course: Teaching, Learning, and Research

G. Amador[1,2] A. Gomes[1,2]

[1]Universidade da Beira Interior, Covilhã, Portugal.
[2]Instituto de Telecomunicações, Portugal.

Abstract

In the last decade, several higher education institutions began to provide courses and/or degrees in games content creation, games design, and games development, largely because of the astonishing growth of games as one of the most powerful industries worldwide. This paper presents the course entitled "Video Games Technologies", including its history, goals and methodology, as part of a MSc degree in Computer Science and Engineering. The focus is on the technologies, techniques, algorithms, data structures, and mathematics behind the design and development of game engines, instead of games themselves.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computers and Education]: Computer and Information Science Education—Computer Science Education; K.8.0 [Personal Computing]: General—Games

1. Introduction

In order to deal with the needs of the game industry [Ip12], higher education institutions worldwide have introduced novel courses and/or degrees in the last two decades [McG12]. In general, we have two sorts of degrees in games. The first is more design and arts-oriented, so that they essentially are bachelor/master degrees in game design [CMA10]. The second sustains on computer science, so that they are bachelor/master degrees in game development and programming [Sun09]. We also find game-related courses in many degree course structures in arts, design, and computer science.

The course briefly described in this paper, called Video Games Technologies, is part of the MSc in Computer Science and Engineering at the Universidade da Beira Interior, Covilhã, Portugal. The focus of this course is not on the design, prototyping, and development (including programming) of video games. Instead, the leading idea of this course is to highlight the mathematics, data structures, and algorithms that sustain the design and development of game engines (e.g., Unity3D [Dic15]). In other words, the idea is not to build up a game, but to master the technologies behind game engines.

The Video Games Technologies course is one of the optative courses of the MSc in Computer Science and Engineering at the University of Beira Interior. Its focus on game engine technologies makes it different from the common game programming and development courses, as those we find in standard curricula as argued by Ritzhaupt [Rit09]. Furthermore, in order to develop the student motivation and curiosity, the course promotes the learning by example, being the students led to tackle open issues and challenges at the research level. So, students end up learning how to extend a game engine architecture by integrating novel features (e.g., data structures and lagorithms) in either game sub-engine, not matter it is a graphics engine, physics engine, artificial intelligence engine, or else.

The rest of the paper is organized as follows. Section 2 reviews previous work related with lecturing and learning video games technologies. Section 3 overviews the video games technologies course history. Sections 4 to 7 detail the course: objectives; prerequisites; syllabus; methodology, i.e., students assessment and teaching methodology vs learning objectives. Sections 8 and 9 provide an preliminary assessment of the course methodology and a critical discussion on the proposed course syllabus and methodology. Finally, Section 10 draws relevant conclusions.

2. Related work

In higher education courses, game engines are used as a basis upon which students rapidly develop a game prototype, simply because constructing a game from scratch (i.e., without the support of a game engine) is tiresome and takes so long time.

On the other hand, some authors argue that simplified game engine tools such as, for example, the XNA framework [LS08], or smaller libraries as Slick or the LightWeight Java Gaming Library (LWJGL) [GHTS09], are more suited for teaching game development than mainstream industry game engines (e.g., Unity [Dic15]). The rationale behind this is threefold: mainstream game engine's

DOI: 10.2312/eged.20161027

learning curve is too long; mainstream game engines provide often excessive features for a learning environment; mainstream open source game engines are monolithic. In true, the only mainstream open source game engine available is the Unreal Engine 4, but it is monolithic, i.e., it is not modular [Gre14].

In a way, this explains why some authors have endeavored to develop didactic game engines, namely: Gedi [CRG05], SAGE [PKR06], G3D [Gre14], Minueto [Den05], enJine [NBJT06], and Mammoth [KVK*09]. In fact, these authors usually point out that mainstream game engines lack modularity, and are not accompanied by any open-source version. Nowadays, this scenario is seemingly changing as game engine companies are making efforts to also adapt their products to teaching. Regardless, these engines are for industrial game development first and for teaching afterwards.

In regard to Video Game Technologies course, which focuses on the mathematics, data structures, and architecture of game engines, a game engine must satisfy two important requirements: open source code availability and modularity. It happens that no current mainstream game engine satisfies these two requirements; hence, we have adopted the jMonkeyEngine 3.0. This enables students changing and extending the functionalities of the game engine in a more delimited part or module of the code, and in a more controllable manner.

3. A Brief Course History

The Video Games Technologies course came into operation for the first time in the academic year of 2008/09, after the Higher Education Reform that took place in Portugal, in 2007, in conformity with the Bologna Treaty.

The course has evolved since then into three main stages or flavors: game development, didactic game engine development, and research-oriented game engine development. The first stage took place between 2008 and 2012, during which the focus was more on the development of game prototypes, rather than algorithms or techniques for game engines, on the XNA framework [LS08], simply because XNA was not open source. Recall that Microsoft dropped its support for XNA in 2013.

The second stage lasted 2 years, between 2012 and 2014, during which we developed an open-source, didactic Java-based game engine, called NoGame Engine, to be used in practicals. However, the lack of appropriate documentation and code samples, in quality and quantity, undermined the success of this initiative. Nevertheless, this stage represents the start of a better awareness and consolidation of the teaching-learning model of the course around the design and development of game engines, from the point of view of software engineering.

The third stage started off in 2014 with the introduction of the jMonkeyEngine 3.0 in the our course, which is a game engine that builds upon Java programming language. It was chosen for the following reasons: modularity, open source availability, cross-platform availability, and thorough documentation, including tutorials, books, and small game prototypes and code snippets. Finally, we had a game engine capable of sustaining research-oriented teaching and learning, as a result of open issues, challenges, and small projects put forward to students.

4. Course Objectives

The *general objectives* of the course are:

- To endow students with skills in design, development, and innovation in game engines, and their data structures and algorithms.
- To empower students with a holistic view of computer science and engineering through video games.
- To prepare students for high-tech companies, and research at MSc and PhD levels.

Regarding *learning objectives*, also called goals or specific objectives, at the end of course the student must be able at least to:

- Make a critical analysis of an algorithm associated with a specific game technology (e.g., a collision detection algorithm).
- Sketch an innovative algorithm as a result of a critical analysis of a category of game engine algorithms.
- Implement, test and incorporate a game engine algorithm into the jMonkeyEngine 3.0 (e.g., a pathfinder).
- Build a video game prototype on the top of a game engine.

5. Course Pre-requisites

The Video Games Technologies is an optative course of the first year, second semester, of the MSc in Computer Science and Engineering. Students enrolling in this course may have a bachelor degree in computer science or not. However, in order to succeed in this course, students must master:

- Problem solving techniques.
- Object-oriented design and development of programs.
- The traditional data structures (e.g., arrays) and algorithms (e.g., graph search), as well as to be proficient in complexity analysis.
- Skills in computer graphics.
- Basics of image analysis and processing.

Therefore, in this course, it is expected that students are familiar with the broad field of computer science, because sooner or later they realize that video games technologies permeates the entire knowledge body of computer science, including computer graphics, computational geometry, and artificial intelligence.

6. Course syllabus

The main topics addressed during the video games technologies course are the following:

- *Introduction, Planning, and Project*. *History of Games* (week 1).
- *Game Genres* (week 2).
- *Game Engines* (week 3).
- *Object Data Structures* (week 4).
- *Spatial Data Structures and Algorithms* (week 5).
- *Scene Graph and Management* (week 6).
- *Culling* (week 7).
- *Terrain Generation and Modeling* (week 8).
- *Motion and Collisions* (weeks 9 and 10).
- *Game Physics* (week 11).
- *Path Finding* (weeks 12 and 13).
- *Game Networking* (week 14).

- *Public Project Presentation* (week 15).

Thus, the theoretical modules of the course are essentially organized according to the structure of the game engine into sub-engines. This course organization also applies to practicals.

7. Course methodology

7.1. Student assessment

The Video Games Technologies course amounts to 6 out of 60 ECTS required to complete the first year of MSc degree. The first year totalizes 60 ECTS equally distributed by 10 courses, with 5 courses per semester. The second year also corresponds to 60 ECTS, during which each student has to elaborate a master thesis under supervision of a professor.

Portuguese student's performance is evaluated according to the grade point scale 0-20, as follows:

- A (excellent) a grade in the interval [18,20];
- B (very good, with few errors) a grade in the interval [16,17];
- C (good, with some errors) a grade in the interval [14,15];
- D (satisfactory, with many errors) a grade in the interval [12,13];
- E (sufficient) a grade in the interval [10,11].

Note, that grades are rounded, e.g., a student with 9.5 marks passes with a 10.

Student assessment is an important part of the teaching-learning process, which is as follows:

- 1 project in games (8.0 marks).
- 1 scientific paper presentation (3.0 marks); this article works as a basis for the project.
- 3 written theoretical tests (6.0 marks).
- 6 game engine programming tests (3.0 marks).

The student projects have three distinct evaluation phases:

1. Project's architecture/prototyping hand-in (2 out of 8.0 marks).
2. Project's alpha code hand-in with preliminary report (3 out of 8.0 marks).
3. Full project hand-in with final report, followed a week latter by a assurance vivas (project defense) (3 out of 8.0 marks).

The project assessment rules are as follows:

- Project proposal goes out in the second week of the course.
- No project will be accepted beyond the deadline.
- The student must hand in the project (in a .zip file) to the instructors personally.
- The project lacks oral defense and practical demonstration in public and in the presence of the instructors.
- Non-working projects will be graded with 0.
- Students demonstrating lack of (or superficial) knowledge on the peculiarities in the development, implementation and testing of their project will be graded with 0 in their projects, i.e., students will be excluded for fraud.
- The final project must be accompanied with a report elaborated in LaTeX.

Each student develops an individual project. However, it is allowed to students to work in group. In case of the project being developed in group, each student develops a distinct algorithm. For example, if the project is on pathfinding and artificial intelligence, each student participating in the project has to design, develop (including implementation), and test a distinct pathfinder. Here, it is where the innovation comes in, because the student is encouraged to propose new solutions to the problem he/she has in hand.

7.2. Teaching-learning methodology

The Video Games Technologies course is structured into theoretical lectures and practicals of 2 hours per week each, in a total of 15 weeks. Thus, we end up having 60 in-class hours of teaching and learning by direct interaction between instructors and students. Theoretical (T) lectures focus on theoretical concepts, methods and algorithms, using the white board or projecting slides, as well as discussing ideas with students whenever necessary. Practicals (PL) are thought to develop the skills of students in programming of game engines, in designing and programming new algorithms, as well as in solving problems proposed by the instructor. Additionally, the course includes 2 office hours per week in order to tutor students, to help them in solving problems and challenges, as well as to monitor students in developing their individual projects. Students are also supposed to spend 100 hours of extra work off classes.

Summing up, the Video Games Technologies course takes place according to the following teaching-learning methodology:

- *Theoretical classes* (T). Theoretical classes in which one provokes, challenges and encourages students to present their ideas and solutions to the problems posed by the teacher. -
- *Practicals* (PL). Practicals in which students are led to develop algorithms introduced and discussed in theoretical classes. These algorithms are implemented in Java and tested on jMonkeyEngine 3.0.
- *Tutorials* (TT). Follow-up tutorial whenever students need it to develop their projects or to consolidate their skills and knowledge. The instructor helps students in solving problems during classes, and promotes weekly tutorial sessions during office hours in MediaLab (research laboratory coordinated by the instructor) and/or in the instructor's office in order to guarantee the progressive improvement of student skills.
- *Project* (P). An individual project serves above all to develop an algorithm that will be integrated into the jMonkeyEngine 3.0.

The idea is thus not only develop in students their skills in game engine programming, but also a keen interest for science and research.

8. Student ranking

In the second semester of the 2014/2015, the video games technologies course had a class of 21 students; 20 of the students successfully completed the course. Considering the Portuguese grade scale [0,20], 1 student ranked in the interval [10,11], 3 in [12,13], 8 in [14,15], 5 in [16,17], 3 ranked in [18,20], and 1 of them stopped attending classes after the first two weeks of the semester for unknown reasons.

Most of the passing students had good scores in the practical

component of the assessment, i.e., practical tests and project. Their projects were more focused on their assigned algorithms to implement than in a having a fully working game prototype. In the project component, students scored well in the interval [5,8], with more than 12 students scoring 6 or more points.

This is to say that the passing students did not present learning difficulties in general. But, as expected, each student demonstrated to be more or less prone/interested in some course subjects than others. Interestingly, those students regularly attending the theoretical classes ranked above those who sporadically attend the same classes. Recall that the instructors did not force students to attend classes, what is in line with the tradition of Portuguese Universities.

9. Discussion

The course was not intended to build games. Instead, it was thought of to teaching and learning the technologies, algorithms, and data structures underlying a game engine. Therefore, in respect to project assessment, more relevance was given to the problem solving and algorithmic design of each game prototype, in particular the integration of new extra algorithms into the game engine.

Regardless, students tend to focus more on doing a complete game than on a working prototype, giving more importance to its final visual aspect than to its underlying features. In fact, from the student point of view, a game prototype owing the features required in the beginning of the semester but lacking features such as, for example, model loading and animation, game menus, and the like, is not seen as a nice game. This motivation to do a complete game appears to be a positive aspect, as it makes students blazing for research and sometimes to pursue a master's thesis in some of the subjects addressed in the course. In fact, some students came up later after concluding this course with new ideas to pursue in their theses.

In comparison to other academic years, we found that the course attained a significant level of maturity for the first time because we used a game engine that meets the following requirements: modularity, open source availability, cross-platform availability, and extensive documentation (i.e., tutorials, books, game prototypes, and code snippets). As a consequence, students could develop all of their potential at cognitive level, as well as at innovative level, in an enthusiastic manner. Another consequence reveals itself in the indicators of the success translated at a better ranking of the students in both theoretical and practical tests, and clearly better, although more complex projects.

10. Conclusions and Future work

The Video Games Technologies course has been a success at the University of Beira Interior, as an optative course of the MSc degree in Computer Science and Engineering, since 2008/2009. Every year, more than 80% of master's students enrol in the course.

Even so, we think that the theoretical modules need some improvements in order to allow students to be more effective in their learning. In particular, they need to learn to better articulate the theoretical modules and practicals in order to boost their creativity and innovation skills. Also, stronger foundations in mathematics would certainly be a great help. So, we are considering the introduction of on-demand maths refreshers whenever needed.

References

[CMA10] COMNINOS P., MCLOUGHLIN L., ANDERSON E.: Educating technophile artists and artophile technologists: A successful experiment in higher education. *Computers & Graphics 34*, 6 (2010), 780–790. doi:http://dx.doi.org/10.1016/j.cag.2010.08.008. 1

[CRG05] COLEMAN R., ROEBKE S., GRAYSON L.: Gedi: a game engine for teaching videogame design and programming. *J. Comput. Sci. Coll. 21*, 2 (2005), 72–82. http://dl.acm.org/citation.cfm?id=1089063. 2

[Den05] DENAULT A.: *Minueto, an Undergraduate Teaching Development Framework*. Master's thesis, School Of Computer Science, McGill University, 2005. 2

[Dic15] DICKSON P. E.: Using Unity to Teach Game Development: When You've Never Written a Game. In *Proc. ITiCSE '15* (2015), pp. 75–80. doi:10.1145/2729094.2742591. 1

[GHTS09] GESTWICKI P., HADDAD A., TOOMBS A., SUN F.-S.: An Experience Report and Analysis of Java Technologies in Undergraduate Game Programming Courses. *J. Comput. Sci. Coll. 25*, 1 (2009), 102–108. http://dl.acm.org/citation.cfm?id=1619221.1619241. 1

[Gre14] GREGORY. J.: *Game Engine Architecture (2nd edition)*. A. K. Peters/CRC Press, 2014. 2

[Ip12] IP B.: Fitting the Needs of an Industry: An Examination of Games Design, Development, and Art Courses in the UK. *ACM Trans. Comput. Educ. 12*, 2 (2012), 1–35. doi:10.1145/2160547.2160549. 1

[KVK*09] KIENZLE J., VERBRUGGE C., KEMME B., DENAULT A., HAWKER M.: Mammoth: a massively multiplayer game research framework. In *Proc. FDG '09* (2009), pp. 308–315. doi:10.1145/1536513.1536566. 2

[LS08] LINHOFF J., SETTLE A.: Teaching Game Programming Using XNA. In *Proc. ITiCSE '08* (2008), vol. 40, pp. 250–254. doi:10.1145/1384271.1384338. 1, 2

[McG12] MCGILL M. M.: The Curriculum Planning Process for Undergraduate Game Degree Programs in the United Kingdom and United States. *ACM Trans. Comput. Educ. 12*, 2 (2012), 1–47. doi:10.1145/2160547.2160550. 1

[NBJT06] NAKAMURA R., BERNARDES JR J. L., TORI R.: Using a Didactic Game Engine to Teach Computer Science. In *Proc. SBGAMES '06* (2006). http://cin.ufpe.br/~sbgames/proceedings/tutorials/SBGames06TC04_enJine.pdf. 2

[PKR06] PARBERRY I., KAZEMZADEH M. B., RODEN T.: The Art and Science of Game Programming. In *Proc. SIGCSE '06* (2006), pp. 510–514. doi:10.1145/1121341.1121500. 2

[Rit09] RITZHAUPT A. D.: Creating a Game Development Course with Limited Resources: An Evaluation Study. *ACM Trans. Comput. Educ. 9*, 1 (2009), 3:1–3:16. doi:10.1145/1513593.1513596. 1

[Sun09] SUNG K.: Computer Games and Traditional CS Courses. *Commun. ACM 52*, 12 (2009), 74–78. doi:10.1145/1610252.1610273. 1