UDC 004.94:378
ORIGINAL SCIENTIFIC PAPER
RECEIVED: 22-08-2011
ACCEPTED: 15-11-2011

ACTA GRAPHICA 195

Introducing Game Development into the University Curriculum

Authors

Bojan Klemenc, Peter Peer*

Faculty of Computer and Information Science,
University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
**E-mail: peter.peer@fri.uni-lj.si*

Abstract:

Integrating computer games development into computer science curriculum is gaining acceptance. However, the question is how this should be done. In our course on computer game development we present all necessary steps that a game project has to address, from design to publication and marketing, from the theoretical to the practical point of view. The goal is that each student makes a casual game for Apple iOS platform and possibly publishes it. The games are built on our XNI framework for iOS, which is a subset of Microsoft's XNA. We take an iterative incremental approach to teaching game development, where we discuss a number of selected topics from various categories, such as gameplay design, graphics and artificial intelligence, each week. Thereafter the students receive mandatory and non-mandatory assignments that force them to add functionality to their game and, thus, steadily progress towards their goal. At the end of the course more than 20 % of all projects were ready for the Apple App Store, which, together with student pools saying that the course was one of the best executed courses they attended, confirms the viability of the suggested scheme.

Keywords:

Computer Games, Curriculum, Incremental Approach, iOS, XNI Framework

1. Introduction

The size of game market does not lag behind the music market and has already surpassed the film market (*Edery & Mollick 2009*). Nowadays, a lot more people play games compared to ten years ago. However, it is not just the numbers that have grown; the target group has expanded as well (ESA 2011). The main reason for the expansion is the appearance of casual games, which are mostly simple and appeal to broader target groups. Many can be accessed free of charge over the Internet, but due to the recent huge expansion in the usage of mobile devices, combined with increasingly more powerful hardware of mobile devices, the games have penetrated into that area as well.

The size of the game market and its opportunities make game development a very interesting career path. Moreover, games are often associated with fun and satisfaction. These factors make game development very interesting for students as it gives them a strong motivation to learn new concepts. On the other side, students can use and interconnect the already acquired knowledge from other fields and courses and demonstrate their knowledge and understanding on working examples.

If we look from the game development perspective, casual games as a special game type have the advantage of being simpler to develop and, correspondingly, having a shorter development time and lower costs. Casual games are therefore suitable for individual game developers. Individual game developers experience the whole process of developing a game and can make use of this knowledge in a greater project, where they can specialize in different areas but understand all components of the project.

At the Faculty for Computer and Information Science, University of Ljubljana, we introduced a game development course as a part of new Bologna reform curriculum. In the course entitled *Game technology and virtual reality* the students are acquainted with different fields of game development and virtual worlds, from technology and programming, to design, marketing and organizational aspects.

As a big game project would be too difficult to accomplish in one semester, the students work on individual projects. We want the students to gain experience in as many fields of computer game development as possible (from graphics, sound, artificial intelligence (AI) etc.), however they should not be overtaxed. The progress of their games should be steady and measurable, so that they do not lose their motivation. So as to accomplish this, we decided to move the focus of the students on the development of casual games for a mobile platform. Each student develops their own game, from gameplay sketch-up up, to the final goal – publishing the game.

The target platform for the games was Apple iOS, as the device hardware was sufficiently capable and the distribution network had already been established. So as to bridge the gap between the platform and the game and to avoid repeated writing of the same code, we introduced a middle layer called XNI. XNI is XNA for iOS and is a subset of the Microsoft XNA Framework in the sense that it mirrors XNA's API. Instead of DirectX, it uses OpenGL ES.

Game development is a synthesis of many technical fields, such as computer graphic, artificial intelligence, sound, physics, but also non-technical graphic design, contents creation, psychology, organization. However, the technical dimension is so strong that courses related to game development are usually offered in the curriculum of computer science.

The course is an optional winter semester course for the third year undergraduate students. The length of the course is 15 weeks; it includes 3 hours of lectures and 2 hours of practical laboratory (tutorial) work each week. Practical work which takes place each week is directly related to the material taught in lectures in the same week. In the first year we had 50 enrolled students. At the end of the semester we had ten games suitable for distribution. Eight of the students decided to publish the games on Apple App Store, where the games were approved and are now available for downloading free of charge.

Since we are taking about a completely new course in our curriculum, the exchange of experiences is important. This is the main aim of the article.

The structure of the article is as follows: In Section 2 we make a review of the inclusion of game development in university education curricula. The game development course at our university is presented in Section 3. A description of the XNI framework that we use in our course is given in Section 4. The results with conclusion are presented in Section 5.

2. Game development in university curricula

ACM curricula recommendations (2008) include the use of computer games to teach computer science. It may not necessarily be a course on its own. As computer game development is an interdisciplinary area, it can be a part of many courses – a programming course or a computer graphics course may have a simple game as an assignment, an artificial intelligence course may have a game pathfinding assignment. One of the reasons to include games in the curriculum is to increase the motivation of the students and to improve the retention of knowledge. Introduction of games as assignments has a beneficial effect when used in teaching areas such as theoretical computer science because the students enjoy a task more if it is personalised instead of predefined (*Hutchins-Korte, 2008*).

Another motivation to include game development into the curriculum is the size of the game market, which encourages the students to learn concepts required to develop computer games. While there is still a debate over the inclusion of computer games in the curriculum (*Haller et al. 2008; Henry M Walker 2003*), a lot of upper education institutions have employed game development in their curricula to a certain degree. We can divide the inclusion of game development in the curriculum into 3 types (Sung 2009): (1) inclusion of games in various computer science classes, for example as assignments; (2) a special course; (3) a whole university program curriculum. Morrison and Preston (2009) made a review of the inclusion of game development by type into (primarily) US universities: many of the reviewed institutions include it as (1) or (2) or both, but there are also institutions with a whole curriculum. They identify that the focus of the courses, especially in (3), but also in (2), can be observed in 3 dimensions: computing (graphics, algorithms, programing), game related (game design, business), art/humanities (assets creation) – different programs place different level of emphasis on each of these areas.

The past curriculum at our faculty included game development in very limited fashion as type (1); however, with the Bologna reform we introduced a standalone game development course with focus on all three areas: computing, game related, and art/humanities. As for the structure of a course, Rabin (2009) recommends for a 10-week-course that 7 to 8 weeks be spent on programming with the remainder focused on understanding games, business management and asset creation. We roughly follow that division.

Various frameworks can be used to teach game development. We use our XNI Framework for iOS, which is derived from Microsoft XNA Framework. The XNA Framework has been successfully used in classes teaching game development (*Wu et al. 2009; Denninger & Schimmel 2008; Shen 2009; Linhoff & Settle 2008; Costantini et al. 2009*). The move to mobile device target platform for teaching game development is effective for motivational reasons and time constraints (*Kurkovsky 2009*) and although the mobile devices are increasingly more powerful, they also compel the programmers to write a more optimal code.

3. Our approach to teaching game development

During the lectures the students learn the theoretical foundations of various topics of game development. The practice thus relates directly to the lectures and the students learn by programming on examples. This way they use the gained knowledge in practice immediately so the retention of knowledge is automatically longer. We also provide additional code snippets that the students can analyse, build on and modify to suit their games. Depending on the nature of their games, they have to consider different possible solutions for various problems and evaluate them. From the perspective of extended Bloom's taxonomy of learning objectives (*Anderson et al. 2000*), the lectures facilitate achievement of the lower level objectives like remembering and understanding, whereas the practice addresses the higher level of objectives like the application of knowledge, analysis, evaluation and finally creation.

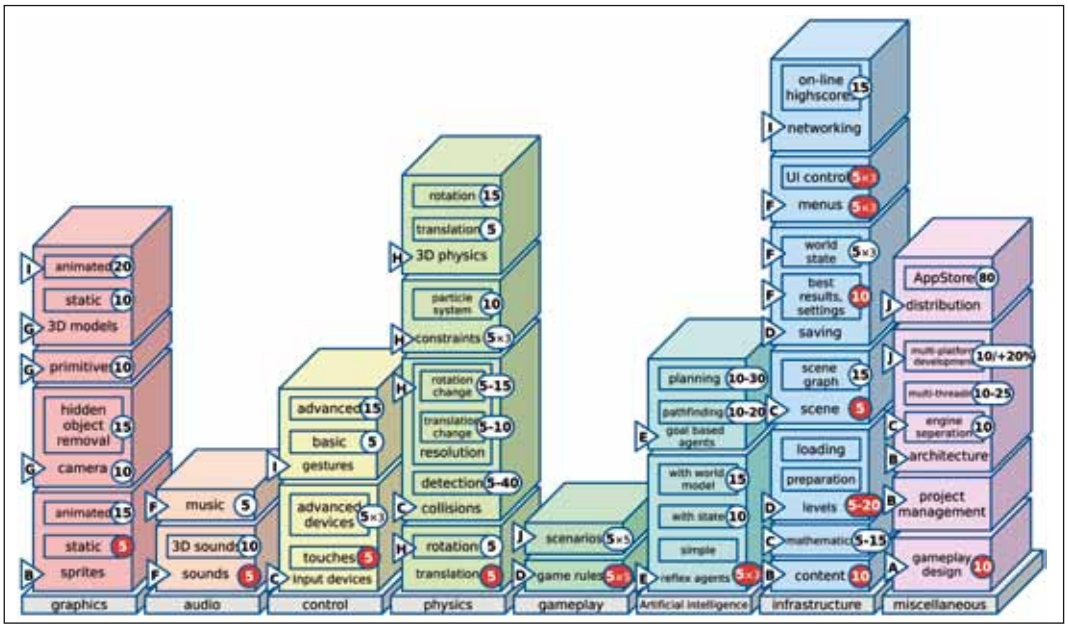

Figure 1 Division of the game development topics into eight categories for the purpose of our course. The numbers represent points for assignments, which are linked to the corresponding topic. Topics which have points with red background have mandatory assignments. The incremental approach (the order of topics) is denoted by alphabet order: the course begins with the topic of gameplay design (A) and ends with distribution (J).

Practice sessions are recorded with Adobe Connect, so the students can, if needed, review the lecture later[1].

As the target platform is iOS, we mainly use Xcode IDE, and most of the code is written in Objective-C. The students' response to using Objective-C was mixed, as the other courses mainly focus on C, Java and Python. However ACM curriculum recommends the students to be exposed to as many different languages and APIs as possible (*ACM, 2008*).

The final grade of the course is obtained mainly through weekly assignments, which function like small milestones or increments in the progress of developing their game.

DIVISION OF FIELDS AND PROGRESS THROUGHOUT TOPICS

Traditional approach to teaching discusses every field as a whole and in detail. This has the advantage that similar topics are organised and taught together. But as game development is a synthesis of many fields, such systematic approach may hamper the learning process, as some required knowledge is taught later on but is needed earlier for practical work. This is constantly slowing the speed of goal achievement. Since there is a lot of interleaving between the fields, we decided to go for the iterative incremental approach, where each field or category is divided into topics (Fig. 1), and only a few topics from a particular category are taught in a particular iteration.

For example, we do not implement the whole graphics engine at once, but we first develop a simple rendering system to get basic visual output; for testing purposes we add some basic input possibilities and add some game logic (see first image in Fig. 2). Later we return to graphics and input by implementing, for example, a camera system, adding support for 3D models and advanced input possibilities like gestures (see second image in Fig. 2). This way we iterate through all the fields and make increments by adding pieces of functionality. The sequential order of topics can be seen in Fig. 1.

[1] http://gameteam.fri.uni-lj.si/badminton/

Figure 2 A student racing game in different iterations: basic sprite render and input in 3rd week of the course (left image); almost finished game with camera implementation, gestures, and AI in around 12th week (right image).

CREATING GAME ASSETS

An important step in the creation of a game is the creation of game assets such as graphics and 3D-models. A game designer or a game engine programmer may not necessary be skilled in the creation of graphics. One possibility is to download free graphics from the internet, but it can be problematic to find suitable graphics that all fit together. In the first phases of development the graphics consists mostly of placeholder graphics (see first image in Fig. 2), which the students can find on the internet or make on their own. A lot of students also take the optional third semester undergraduate course *Graphic design* and can therefore utilize their knowledge in creating their own graphics (see second image in Fig. 2). Those who did not take the course could cooperate with external artists, which is also an opportunity to improve their communication skills.

4. XNI framework

When writing the game code, we often stumble upon a code that is present in every game, but is independent of it, for example texture rendering or sound playback. We do not wish to write that code every time anew. On the other side, we have different platforms and different APIs. During the game development we would like to focus on the development of the game logic and not bother with, for example, how to render a texture in OpenGL or DirectX. To bridge the gap between the platform and the game and to avoid repeated writing of the same code, we introduced a middle layer which also acts as our starting framework, XNI. XNI is XNA for iOS and is a subset of the Microsoft XNA Framework in the sense that it mirrors XNA's API. XNI is written in Objective-C and can be freely used under MIT Licence. XNI was developed by Matej Jan for the needs of our course.

One of the questions that we face when teaching game development is which level do we start on. We can let the student program with the most basic OpenGL or DirectX function calls, however, as we have a computer graphics course, which is a precondition for taking the game development course, we assume that they know the basics and, thus, XNI allows them to skip writing that kind of code. On the other hand, it may as well be compelling to give the students a full-featured game engine with a toolset to work with. The downside of this approach is that they may be trained for a specific tool and do not understand the inner mechanisms of an engine. XNI is in this way a compromise as it hides the low-level functions and speeds-up the programing of the game. However, XNI is not a game engine, but a foundation to build the game engine upon, so that the students learn the internals of a game engine in practice.

5. Results and conclusion

At the end of the course we had ten games suitable for distribution. Eight of the students decided to publish the games on Apple App

Figure 3 Screenshots of games developed during our computer game development course in the academic year 2010/11.

Store, where the games were approved and are available for download free of charge. All the descriptions of the games are available online.

The game genres chosen by students ranged from arcade style short games, which were the most common type, top-down driving games, a side-scroll action shooter to a strategy game.

Fig. 3 features screenshots from some of the games developed during the course. The games are predominantly short due to workload constraints. The goal i.e. the publishing of the game was a good motivator, as the students spent additional free time on polishing the game before the launch. The games published on App Store had all together around 9.000 downloads in the first 2 months. From the student perspective, the success with the game demonstrates their ability to complete a project and this can be included in the resume.

The two short show-case videos were prepared to summarise the work done within the semester from the teachers and the student's point of view.

The student feedback on the course was generally very positive. Their remarks on forums outside the faculty, such as that the course was probably one of the best executed courses they attended gave all the teachers involved a feeling of great satisfaction.

In the next years we want to streamline the flow of the course by adjusting the topics and the time they are allocated. There may be some corrections of the scoring system. As the platforms are constantly developing, we will continue to update the course so it reflects these developments.

Acknowledgement

Matej Jan contributed substantially to the success of the course. He is one of the leading game developers in Slovenia and he developed the XNI Framework.

References

ANDERSON, L.W. ET AL., 2000. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Abridged Edition* 2nd ed., Allyn & Bacon.

ASSOCIATION FOR COMPUTING MACHINERY. Computer Science Curriculum 2008: An Interim Revision of CS 2001. http://www.acm.org/education/curricula-recommendations.

ENTERTAINMENT SOFTWARE ASSOCIATION. 2011. Essential Facts About the Computer and Video Game Industry. http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf.

COSTANTINI, G., MAGGIORE, G. & CORTESI, A., 2009. Learning by fixing and extending games. *Proceedings of Eurographics: Education paper*.

DENNINGER, O. & SCHIMMEL, J., 2008. Game Programming and XNA in Software Engineering Education. *Proceedings of Computer Games and Allied Technology (CGAT08)*.

EDERY, D. & MOLLICK, E., 2009. *Changing the game: how video games are transforming the future of business*, FT Press.

HALLER, S. ET AL., 2008. Games: good/evil. In *Proceedings of the 39 th ACM SIGCSE technical symposium on Computer Science Education*, pp. 219–220.

HUTCHINS-KORTE, L., 2008. *Learning by Game-Building in Theoretical Computer Science Education*. Scotland, UK: University of Edinburgh.

KURKOVSKY, S., 2009. Engaging students through mobile game development. *SIGCSE Bull.*, 41(1), pp. 44–48.

LINHOFF, J. & SETTLE, A., 2008. Teaching game programming using XNA. *Proceedings of the 13th annual ACM conference on Innovation and technology in computer science education (*ITiCSE), pp. 250–254.

MORRISON, B.B. & PRESTON, J.A., 2009. Engagement: gaming throughout the curriculum. *Proceedings of the 40th ACM SIGCSE technical symposium on Computer science education*, pp. 342–346.

RABIN, S., 2009. *Introduction to Game Development,* 2nd ed., Charles River Media.

SHEN, Y., 2009. Teaching game development using microsoft XNA game studio. *Proceedings of the 2009 Spring Simulation Multiconference* (SpringSim), pp. 1–6.

SUNG, K., 2009. Computer games and traditional CS courses. *Communications of the ACM*, 52(12), pp. 74–78.

WALKER, H.M., 2003. Do computer games have a role in the computing classroom? *SIGCSE Bull.*, 35(4), pp. 18–20.

WU, B. ET AL., 2009. An evaluation of using a game development framework in higher education. Proceedings of the 22nd Conference on Software Engineering Education and Training (CSEET), pp. 41–44.