Teaching database concepts to video game design and development students

Nuno Fachada

Abstract

A video game design and development degree is a very specific choice for students, who are mainly interested in making games or taking part in the game development process. Databases are not an apparent requirement in order to pursue these goals, often leading to a lack of motivation and interest in the subject. Nonetheless, a number of Bachelor degrees in video game design and development acknowledge the importance of Databases, and offer it as a mandatory course in their curricula. However, the subject exposition is often done in the context of typical database areas of application, such as business settings, websites or library and university management. In this paper we describe and propose four classroom database problems specifically designed for video game design and development students. It is our belief that using a context-aware approach yields not only more motivated students, but students with a better understanding of database concepts so often necessary to design and develop computer games.

Keywords:

programming curriculum; video game development; databases; SQL

Ensinar conceitos de bases de dados a alunos de concepção e desenvolvimento de vídeo jogos

Resumo: Um curso superior sobre design e desenvolvimento de vídeo jogos é uma escolha bastante específica para alunos que estão sobretudo interessados em criar jogos ou em fazer parte do processo de desenvolvimento desses jogos. À primeira vista, a disciplina de Bases de Dados não é um pré-requisito na persecução destes objetivos, levando muitas vezes à desmotivação e desinteresse dos alunos que a frequentam. No entanto, boa parte dos cursos superiores de ***design*** e desenvolvimento de vídeo jogos incluem a disciplina de Bases de Dados na sua componente curricular obrigatória, reconhecendo de forma clara a importância da temática. Por outro lado, a exposição do assunto é geralmente realizada no contexto das áreas clássicas das Bases de Dados, como, por exemplo, cenários empresariais, websites ou gestão de bibliotecas e universidades. Neste artigo descrevemos e propomos quatro problemas de sala de aula especificamente elaborados para alunos de vídeo jogos. Acreditamos que o uso de uma abordagem contextualizada na criação destes conteúdos não só motivará mais os alunos como também os ajudará a compreender melhor os conceitos de Bases de Dados, muitas vezes necessários para desenvolver jogos de computador.

Palavras-chave: currículo de programação; desenvolvimento de vídeo jogos; bases de dados; SQL

Enseigner des concepts de bases de données aux étudiants en conception et développement de jeux vidéo

Résumé: Un diplôme de conception et de développement de jeux vidéo est un choix très spécifique pour les étudiants, qui sont principalement intéressés à développer des jeux ou à prendre part à son processus. Les bases de données ne sont pas une exigence évident pour poursuivre ces objectifs, conduisant souvent à un manque de motivation et d'intérêt pour le sujet. Cependant, un certain nombre de baccalauréats en conception et développement de jeux vidéo reconnaissent l'importance des bases de données, et l'offrent comme un cours obligatoire dans leurs programmes d'études. Malheureusement, l'exposition du sujet est souvent faite dans le contexte des domaines d'application typiques de la base de données, comme les scénarios des entreprises, les sites Web ou la gestion des bibliothèques et des universités. Dans cet article, nous décrivons et proposons quatre problèmes en classe de base de données, spécialement conçus pour les étudiants en conception et développement de jeux vidéo. Nous sommes d'avis que l'utilisation d'une approche contextuelle permet non seulement motive davantage les étudiants, mais également fournit une meilleure compréhension des concepts de base de données, habituellement nécessaires au développement de jeux informatiques.

Mots-clés: programme de programmation; développement de jeux vidéo; bases de données; SQL

Enseñar conceptos de bases de datos a estudiantes de diseño y desarrollo de videojuegos

Resumen: El Grado en Diseño y Desarrollo de Videojuegos es un curso específicamente destinado para aquellos alumnos interesados en crear videojuegos o formar parte del proceso de creación. La asignatura Base de Datos no es considerada un pre-requisito evidente para tal fin, lo que genera muchas veces desmotivación y desinterés por parte de los alumnos que la cursan. Sin embargo, grande parte de los cursos de Grado en esta especialidad incluyen esta materia en su oferta curricular obligatoria, reconociendo de esta forma la importancia de la misma. A su vez, la forma en que se expone el contenido de la materia es frecuentemente en contextos típicos en los que se suelen utilizar bases de datos, tales como ambientes empresariales, bibliotecas, websites y gestión universitaria. En este trabajo, proponemos y explicamos cuatro casos prácticos de bases de datos especialmente pensados para alumnos de diseño y desarrollo de videojuegos. Creemos que un abordaje que tenga en cuenta el contexto en el que se aplican las bases de datos puede motivar a los alumnos, así como también ayudarlos a comprender mejor los conceptos relacionados con bases de datos, muchas veces necesarios a la hora de diseñar y desarrollar videojuegos.

Palabras-clave: programa de programación; desarrollo de videojuegos; base de datos; SQL

Introduction

A video game design and development degree is a very specific choice for students. Sitting between the broader areas of Computer Science and Art, with the specific balance defined by the University in question, such degrees are chosen by students interested in making games or taking part in the game development process. Part of this process involves visual feedback, seeing things moving, appearing on screen, as a result of player interaction. Conversely, databases (DB) and database-management systems (DBMS) seem distant from these goals, leading, according to our experience, to a lack of motivation and interest among these students. In the educational context, teaching DBs and DBMSs is not a priority in many of these degrees, and the subject is often not offered as a separate course International Student (2017; Instituto Politécnico de Bragança, 2017; Instituto Politécnico de Leiria, 2017; Media Design School, 2017; Study.com, 2017). In the video game design and development degrees that do offer a separate DB course Universidade Europeia (2017; Escola Superior de Tecnologia, Instituto Politécnico do Cávado e do Ave, 2017; Universidade Lusófona de Humanidades e Tecnologias, 2017; Vilniaus Verslo kolegija, 2017), the subject exposition can easily be done in such a way that the associated theory and problems bear little connection with video game development. This may occur due to the degrees themselves being associated with informatics and/or computer science majors, or simply due to educators having informatics and/or computer science backgrounds. In these fields of education, DBs and DBMSs are typically introduced within the context of their main areas of application, namely business settings, websites or library and university management Thakur (2016; Damas, 2017).

Nonetheless, databases can be an essential aspect of game development. Their most obvious application is in multiplayer games, which are commonly required to keep track of players and many of their interactions with the game world, such as non-aggregate match statistics in a first-person shooter (FPS). Role-playing games (RPG), multiplayer or otherwise, benefit from being able to keep player objects, weapons, spells and player level-up details in a dedicated data store. The use of a DBMS can be quite convenient even in purely single-player games, for example in keeping track of objects and events in the game world, saving game progress (e.g. in save games) or game configurations. In the single-player case, local DBMSs such as SQLite Owens (2006) or LiteDB David (2017) are able to provide complete DB functionality without the overhead of having a dedicated server process.

Considering the importance of databases in video games, the approach used in teaching DBs and DBMSs should be adapted to the video game design and development context. More specifically, the associated topics should be presented using video game related exercises, problems and examples. Although these are not as simple to

forge as in the areas where DBs and DBMSs are most associated with, some perseverance and imagination can lead to interesting and motivating outcomes. In the current paper we describe and propose four classroom DB problems and possible variants for video game design and development students. It is our belief that using a more customized approach as the one presented here, yields not only more motivated students, but students with a better understanding of DB concepts so often necessary to design and develop computer games.

The paper is organized as follows. The proposed exercises, including their pedagogical goals and possible variants, are presented in Methodology. A preliminary assessment of how the proposed problems are faring, and a discussion of how to evaluate the final results, is undertaken in Results and Future Work. Conclusions of what was accomplished with this paper are presented in the final section.

Methodology

In this Section four different database problems are proposed. Each proposal is organized into five subsections, namely, the pedagogical goals, problem statement, instructor preparation, class activities and possible variants.

An engaging introduction to databases

This example aims to be a preliminary introduction to DBs, at a stage where students have yet to learn anything concrete about the topic. It is best presented in the first class of the Databases course.

Goals

- Introduce students to basic DB concepts, such as entities and relations, and DB modeling techniques such as Entity–Relationship diagrams.
- Motivate students on how DBs can be an important component in a video game.
- Highlight the potential of DBs in yielding crucial information about how a game is being played.
- Engage students by using their names in the player list and then querying the DB to check who played more and who played better.

Problem statement

A moderately successful first person shooter (FPS) game was developed by University students last semester. This semester, and in order to increase their game's market share, the same students decided to implement a multiplayer variant into their shooter. To do that, they needed to create a DB to keep track of the multiple players, the different maps, and the scores and time played by individual players on each map. Figure 1 shows the main entities and relations in the database.

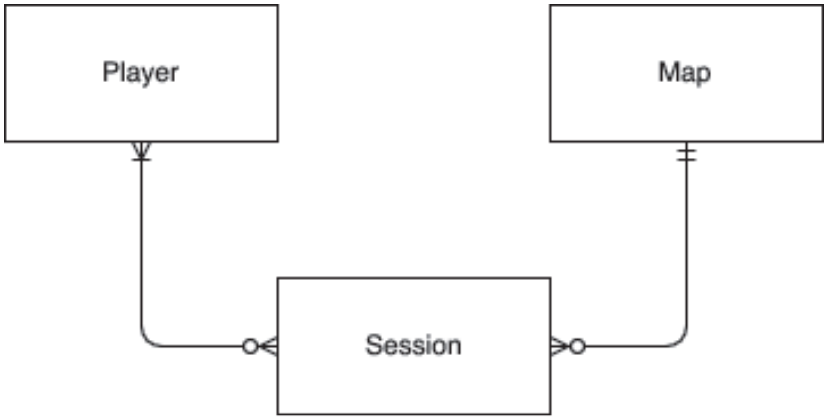

Figure 1. Entity–Relationship (ER) diagram (using Crow's Foot notation) of a conceptual model for the introductory problem.

Instructor preparation

The instructor should prepare a physical schema based on the conceptualization presented in Figure 1. An intermediate logical model, shown in Figure 2, provides additional details which can be used in the DBMS-dependent physical schema. For example, the *Player* and *Map* tables are only required to have ID (primary key) and name fields, although additional columns can be added for scenario enrichment purposes (e.g., a *Map* can have a filename where it is stored). In turn, the *Session* table needs columns for its ID (primary key), the *Map* ID (foreign key), and possibly the date/time the session started and ended. An additional table is required for the many-to-many relation between *Player* and *Session* (e.g., *Player_Session*). Besides having the primary keys of *Player* and *Session* as foreign keys, this table should also hold the player's score and time played.

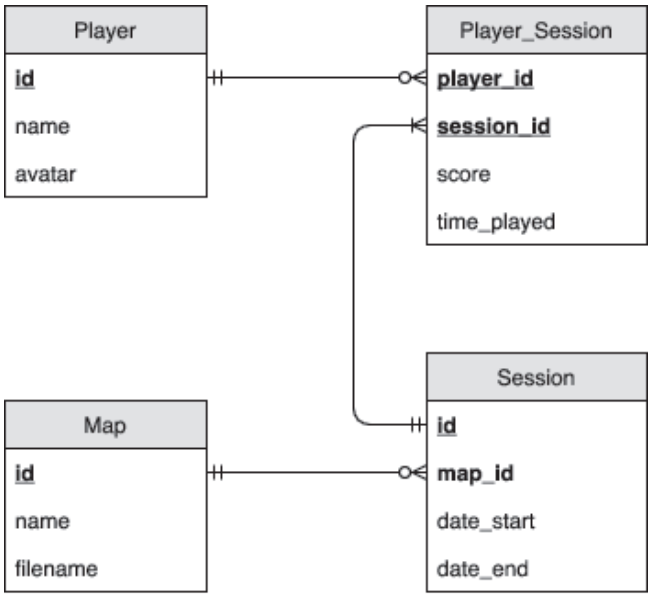

Figure 2. Entity–Relationship (ER) diagram (using Crow's Foot notation) of a logical model for the introductory problem. The main difference with respect to the conceptual model (Figure 1) concerns the use of an additional table for mapping the many-to-many relation between Players and Sessions. Furthermore, a number of fields, as well as primary and foreign keys, are proposed.

The tables should be populated as follows:

- The *Player* table should be populated with the class students names in order to engage the students.
- The *Map* table should be populated with suggestive map names, e.g. prison, graveyard, playground, and so on.
- The *Session* table should be populated with a large number (i.e. > 200) of game sessions involving most or all of the players in the various maps, with randomly generated scores and session durations.

The DB should be read-only accessible to students via a front-end such as phpMyAdmin Delisle (2012) for MySQL Widenius (2002), SQLiteStudio Salawa (2016) for SQLite, or a DBMS-independent program such as DbVisualizer DbVis Software AB. (2017). Online alternatives such as http://sqlfiddle.com/ or sqliteonline.com are also an interesting alternative. In any case, the front-end should be simple as possible, such that they can experiment with and observe how the DB works without obstacles.

The instructor should also prepare a few SELECT statements for students experiment with. Some possible queries include, but are not limited to:

- Total time played by player.
- Total time played by player for a specific map.
- Top 5 total time played by player for a specific map with customized fields (AS keyword).
- Total time a specific player (a student, preferably one present in the class) has played.
- Top 10 scores on a specific map.
- Best scores for a specific player (a student, preferably one present in the class) at a specific map.
- Number of sessions and time played per map, sorted by number of sessions.

Class activities

The proposed class activities for this exercise are as follows:

1. The problem is presented and discussed with the students.
2. Students are given a file containing the SQL code which will create and populate the DB.
3. The SQL queries are given, one-by-one, to the students, which in turn will:
    - Copy and paste the query to the DB front-end.
    - Execute the query.
    - Observe, analyze and discuss the results among themselves and with the instructor.

Variants

As stated, the problem is clearly oriented towards relational DBs. If such DBs are not the primary focus of the course, variants of the problem can be developed for document, key-value or object DBs, for example.

The reputation problem

This problem aims to help students form an idea of the several issues and pitfalls associated with designing a DB schema. Consequently, it is best presented when students are learning DB modeling rules and techniques.

Goals

- Understanding that problem descriptions are often vague and ambiguous.
- Learn to use defensive modeling approaches while maintaining the schema scalable and adaptable to future changes and minimizing potentially bad design decisions.

- Learn to keep non-aggregate information in the DB, since aggregate statistics can be retrieved later by querying the DB and possibly manipulating returned results in a general-purpose programming language.

Problem statement

The company behind a well-known multiplayer game wants to implement a player reputation system. After a preliminary analysis, it has been determined that player reputation should depend on:

- Rating provided by other players.
- Number of game sessions played.
- Average time per session.

Instructor preparation

Not much preparation is needed on behalf of the instructor, which should essentially be aware of the several design pitfalls and be ready to discuss them with the students.

Class activities

The proposed class activities for this problem are as follows:

1. Design DB schema while discussing possible modeling issues, namely what is meant by “rating”? Is this a unique, once in a lifetime value provided by other players, or is this a score players give others for individual game sessions? What range of values defines a rating?
2. Determine a simple mathematical formula for determining overall player reputation, discussing how to weigh and possibly define a ceiling on individual items which affect it. For example, if reputation is to be an absolute value (e.g., between 0–100), how to properly scale the average time played into the final reputation value.
3. In any case, it should be concluded that reputation is not adequate for being stored in a DB, since it can be calculated using other information which should be in the DB.

Variants

Add, remove or change some of the variables that determine reputation.

The critics example

This example is oriented towards relational DBMSs and shows how to model schemas when data comes from various sources, possibly in different formats. Its textual description is considerably detailed so that students can focus on this issue, and proceed with table creation and query building. Thus, the example is best presented when students already know how to create physical schemas and query the database.

Goals

- Understand how to model schemas when data comes from different sources.
- Introduce the concepts of mapping of relations in object-oriented programming to relational DBs, using an inheritance relation as an example.
- Query the DB having multiple sources, possibly mapped to different tables, into account.

Problem statement

In order to help players buy the best games, a company wishes to improve the game evaluation system in their games platform. Games are subject to criticism, and such evaluations can come from two sources[1]:

- Professional reviews (game magazines, websites, renown ***youtubers***, and so on).
- The game players themselves.

Individual game evaluations or criticisms should have:

- A score in the range 0–100, which might have to be normalized for professional reviewers who offer scores in different scales (e.g. 0–5 or 1–10).
- A text description, which is the full review in the case of players, and a summary of the full review performed by professional reviewers.

Furthermore: a) only players who own the game can write reviews and provide the respective score; and, b) professional reviews should have an associated link to the original review.

Finally, it might be interesting in the future to give more weight to reviews of players who have played the game the most.

Instructor preparation

No special preparation is needed by the instructor, other than being prepared to discuss the issue of modeling and manipulating a relational DB when data can come from disparate sources with specific particularities.

Class activities

1. Design DB schema focusing the discussion on how to model evaluations from players and professional reviewers, namely looking at this relation as an object-oriented inheritance relationship, as shown in Figure 3. More specifically, the discussion should center itself on how to best model this inheritance relationship, with at least two competing possibilities:
   - Create an "super" *Evaluation* entity containing all the fields required by the different types of evaluations, which will most likely have a high count of undesirable `NULL` values.
   - Create tables for the super class and extending classes, where the primary key of the latter must also be a foreign key to the former. Additionally: a) discuss the possibility of distinguishing the type of evaluation just by looking at the table associated with the super class; and, b) discuss how to guarantee that keys in one of the extending classes does not appear in the other.
2. Create a physical schema with the appropriate SQL statements, and fill the DB with test data.
3. Build the required `SELECT` statements for obtaining a weighted criticism on a specific game. If the total time each player played the game is saved in the DB, then construct queries to have this into account when weighing individual player evaluations.

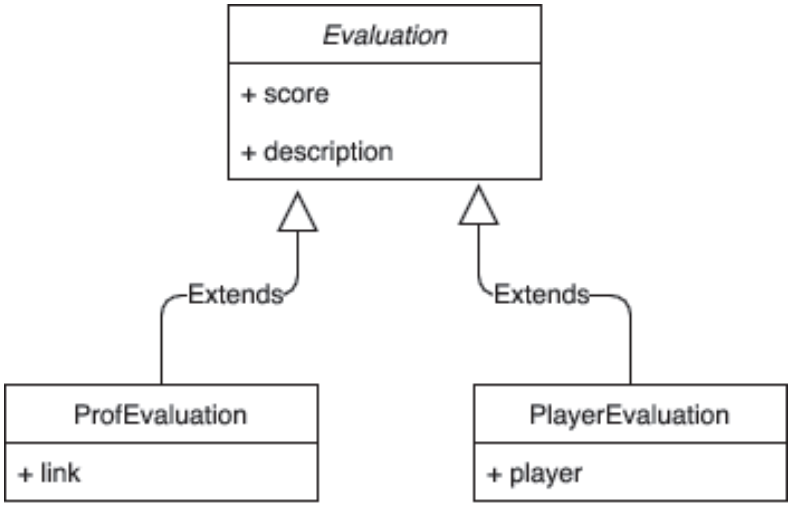

Figure 3. UML diagram of the relation between evaluations, professional evaluations and player evaluations.

Variants

We propose two variants of this example:

1. Allow for even more types of reviews or differentiate between evaluations from game magazines, websites, ***youtubers***, and so on.
2. Instead of being video game-oriented, the example could follow the Metacritic approach more directly, and be more general, allowing the user and professional reviews of movies, TV shows, music and video games. In this case it would probably be preferable to not require that video game evaluations be made by players who own or have played the game.

RTS with rules: a new game approach

Goals

This problem can be used as the base for a final course project. It encompasses different aspects and steps of DB design, implementation and experimentation. Additionally, it the problem description directly involves the development of a video game with a few "novel" game ideas, further motivating and engaging students.

Problem statement

A group of students pursuing the same degree is developing a multiplayer real-time strategy (RTS) game with a twist. Every player plays against everyone, similar to Total War: Warhammer 2 all-against-all four player mode Andrews (2017). The twist is that every battle or session takes place under a set of rules (*ruleset*) that players have to voluntarily follow. As is typical of RTS games, each player commands its own army, and battles can take place on different maps.

Each *ruleset* is composed of individual rules, such as restrictions on what units or weapons may be used, "sacred" terrain types where armies should not pass or where players should not build structures, limitations to temporary alliances that can be made, and so on. Individual rules have at least one name and one unique code that identifies them in the game engine.

The interesting part of the game is that it does not impose the rules on players. In fact, players should obey the rules voluntarily. Rule compliance is monitored by the players themselves. If a player spots a "cheater" in the field vision of its army, he can denounce the offending player within the game itself. If the game engine verifies that this was in fact a violation of the rules, the "cheater" immediately loses the battle. Furthermore, an entry is created on a blacklist of transgressors, specifying who was

the offending player, who was the "snitch", in which battle the offense took place, and what rule was broken. On the other hand, if a player reports an occurrence that was not a violation of the rules, he also immediately loses the current battle, and it is his ID who gets a place in the transgressors blacklist. In this case the "snitch" field is set to NULL, with the broken rule field set to a special rule (which can not be used in *rulesets* – or can it?) indicative of false reporting.

Each battle is associated not only with a map and a *ruleset*, but also with the final standing of all players involved. The best ranked player in each battle is the one that lasted longer on the field, since it wiped out the armies of the remaining players.

Instructor preparation

Since this problem is to be offered as a final or intermediate project, there is no specific class preparation for instructors.

Project activities

It is intended that students propose a solution taking into account the various problem elements, but also possible future developments of the game. For example, the students who are developing the game are considering the implementation of a player ranking, updated daily. The position of a player in this ranking might positively depend on the number of wins, or depend negatively on the number of times the player violated the established *ruleset*s for each battle he played. In addition, it would be interesting to obtain all kinds of statistics regarding the game sessions, average time played per map per winner, distribution of cheats per map, mean number of cheats per *ruleset*, and so on. The database should be designed taking into account these and other types of future needs.

Students should submit two elements to be evaluated by the instructors:

1. A properly structured technical report, containing at least the following:
   - Conceptual and/or logical model with an ER or UML diagram showing only the main entities and relations in the problem.
   - Physical DB schema, with an ER or UML diagram explicitly showing all schema entities and respective fields, keys and types, as well as a short text describing the main design differences with respect with the previous point.
   - In the case of relational DBs, a demonstration that all tables are at some level of DB normalization.
2. A text file containing the DBMS-specific statements to create the DB, insert information into the DB, and query the DB. These statements should be properly tested and work without error. In the case of relational DBs, this would be a text file with an

`.sql` extension containing the following SQL instructions:

- A number of `CREATE TABLE` statements that reflect the physical model defined in the technical report.
- A number of `INSERT` statements that validate and test the tables created in the previous point.
- A number of `SELECT` statements demonstrating that the proposed database schema contains the necessary information for determining player rankings, and is able to cope with some of the described future needs.

Variants

This problem can easily be modified. For example:

- Instead of the "novel" rule-breaking system, a different twist can be used, as long as it is also amenable for mapping to a DB context.
- Allow co-op teams, penalizing only the rule-breaking player in the adversary team (i.e., the team itself would not immediately lose the game).
- Apply the proposed idea of players policing each other to other game genres, such as turn-based games or massively multiplayer online RPGs.

Preliminary results

The proposed methodology is currently being implemented in the Databases course of the Bachelor degree in Video Games and Multimedia Applications at Universidade Lusófona de Humanidades e Tecnologias, in Lisbon, Portugal. Preliminary assessment, based on informal conversations, shows that students are motivated and engaged with the course, contrary to what seemed to be the case in previous years, where a non-context aware teaching methodology was used. These conversations involved students enrolled in the Databases course only once (experimenting either the non-context aware or context aware approach), and students who were enrolled in two consecutive years, thus experiencing the two contrasting methodologies.

Conclusions

In this paper we proposed a context-aware DB and DBMS teaching methodology for video game design and development students. The approach is based on presenting students with exercises, problems and examples concerning the direct use of DBs in video games. Four problems were presented, along with possible variants, the associated pedagogical goals, class activities and instructor preparation. It is our belief this methodology is both motivating and engaging for video game design and development

students, helping them to gain a better understanding of DB concepts so often necessary to design and develop computer games. The proposed problems can be used directly or serve as a first step or inspiration in building better contextualized DB teaching materials. Preliminary results, based on informal conversations with video game design and development students, show high levels of motivation and engagement with a DB course implementing the proposed methodology.

Notes

[1] following the pattern observed in services such as Metacritic (http://www.metacritic.com/).

References

Andrews, S. (2017). Total war: Warhammer 2. Trusted Reviews. [Available from http://www.trustedreviews.com/reviews/total-war-warhammer-2, accessed on 14/10/2017].

Damas, L. (2017). *SQL – structured query language (14th ed.).* FCA.

David, M. (2017). LiteDB – embedded NoSQL database for NET. [Available from http://www.litedb.org/, accessed on 10/10/2017]

DbVis Software AB. (2017). DbVisualizer. [Available from https://www.dbvis.com/, accessed on 08/10/2017].

Delisle, M. (2012). *phpMyAdmin starter*. Packt Publishing.

Escola Superior de Tecnologia, Instituto Politécnico do Cávado e do Ave. (2017). Licenciatura em Engenharia em desenvolvimento de jogos digitais – estrutura curricular. [Available from https://est.ipca.pt/curso/desenvolvimento-de-jogos-digitais/, accessed on 20/10/2017].

Instituto Politécnico de Bragança. (2017). Design de jogos digitais – o essencial da licenciatura. [Available from http://djd.esact.ipb.pt/docs/brochura_pt.pdf, accessed on 20/10/2017].

Instituto Politécnico de Leiria. (2017). Licenciatura em jogos digitais e multimédia – plano curricular. [Available from http://www.jogos.ipleiria.pt/#planoCurricular, accessed on 20/10/2017].

International Student. (2017). Study video game development in the uSA guide. [Available from https://www.internationalstudent.com/study-video-game-development/, accessed on 14/10/2017].

Media Design School. (2017). Game programming – bachelor of software engineering. New Zealand. [Available from https://www.mediadesignschool.com/courses/game-programming, accessed on 19/10/2017].

Owens, M. (2006). *The definitive guide to SQLite*. Apress.

Salawa, P. (2016). SQLiteStudio 3.1.1. [Available from https://sqlitestudio.pl/, accessed on 21/10/2017].

Study.com. (2017). Associate of video game design and development: Degree overview. [Available from http://study.com/articles/Associate_of_Video_Game_Design_and_Development_Degree_Overview.html, accessed on 14/10/2017].

Thakur, S. (2016). Application and uses of database management system (DBMS). [Available from http://whatisdbms.com/application-and-uses-of-database-management-system-dbms/, accessed on 17/10/2017].

Universidade Europeia. (2017). Licenciatura em desenvolvimento de jogos e aplicações – plano de estudos. [Available from https://www.europeia.pt/oferta-formativa/licenciaturas/licenciaturas-globais/games-apps-development#module-tabs1, accessed on 20/10/2017].

Universidade Lusófona de Humanidades e Tecnologias. (2017). Licenciatura em aplicações multimédia e videojogos. [Available from https://www.ulusofona.pt/licenciatura/aplicacoes-multimedia-e-videojogos, accessed on 10/10/2017].

Vilniaus Verslo kolegija. (2017). Media and computer games, professional bachelor's degree in applied informatics. [Available from http://www.kolegija.lt/en/studiju-programos/media-and-computer-games/96, accessed on 18/10/2017].

Widenius, M. (2002). *MySQL reference manual*. O'Reilly Media.

Nuno Fachada
Sub-director e docente do Curso de Videojogos da ULHT, Lisboa, Portugal.
Investigador integrado no Hei-Lab/ULHT e colaborador no ISR/IST, Lisboa, Portugal.
Email: nuno.fachada@ulusofona.pt
ORCID: 0000-0002-8487-5837

Correspondência:
Nuno Fachada
ECATI – Escola de comunicação, arquitetura, artes e tecnologia de informação
Universidade Lusófona de Humanidades e Tecnologias
Campo Grande, 376, 1749-024 Lisboa

Data de submissão: Janeiro 2018
Data de avaliação: Março 2018
Data de publicação: Julho 2018