Teaching computer game development with Unity engine: a case study

Natalia V. Moiseienko[1], Mykhailo V. Moiseienko[1], Vladyslav S. Kuznetsov[1], Bohdan A. Rostalny[1] and Arnold E. Kiv[2,3]

[1]Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine
[2]Ben-Gurion University of the Negev, P.O.B. 653, Beer Sheva, 8410501, Israel
[3]South Ukrainian National Pedagogical University named after K. D. Ushynsky, 26 Staroportofrankivska Str., Odesa, 65020, Ukraine

Abstract

Computer game development is a popular and engaging topic that can motivate students to learn various aspects of software engineering, such as design, programming, testing, and teamwork. However, there is a lack of research on how to effectively teach this topic in the context of secondary education. In this paper, we present our experience of designing and delivering a course on computer game development for master's students in the specialty 014.09 Secondary education (Informatics) at the Kryvyi Rih State Pedagogical University. We describe the objectives, content, software tools, and teaching methods of the course, as well as the challenges and outcomes of its implementation. We also evaluate the course using a framework proposed by Ritzhaupt [1] based on student feedback and learning outcomes. Our results show that the course was successful in achieving its goals and enhancing students' knowledge and skills in game development. We also identify some areas for improvement and provide recommendations for future iterations of the course. We conclude that Unity Engine is a suitable platform for teaching game development in secondary education, as it offers a low barrier to entry, a rich set of features, a cross-platform compatibility, and a wide adoption in the game industry. We also argue that a team-based approach is beneficial for fostering collaboration and creativity among students.

Keywords

computer game development, software engineering education, Unity Engine, secondary education

The software industry is a dynamic and market-oriented industry that requires constant innovation and adaptation to changing customer needs and technological trends [2]. One of the most prominent and lucrative segments of this industry is the video game industry, which produces interactive entertainment products that appeal to a wide range of audiences and platforms [3, 4].

According to a report by Gamesindustry.biz and Newzoo, the global games market value reached $189.3 billion in 2022, surpassing the film and music industries combined (figure 1).

3L-Person 2022: VII International Workshop on Professional Retraining and Life-Long Learning using ICT: Person-oriented Approach, October 25, 2022, Kryvyi Rih (Virtual), Ukraine
n.v.moiseenko@gmail.com (N. V. Moiseienko); seliverst17moiseenko@gmail.com (M. V. Moiseienko); kiv.arnold20@gmail.com (A. E. Kiv)
https://kdpu.edu.ua/personal/nvmoiseienko.html (N. V. Moiseienko); https://kdpu.edu.ua/personal/mvmoiseienko.html (M. V. Moiseienko); https://ieeexplore.ieee.org/author/38339185000 (A. E. Kiv)
0000-0003-0789-0272 (N. V. Moiseienko); 0000-0003-4401-0297 (M. V. Moiseienko); 0000-0002-0991-2343 (A. E. Kiv)

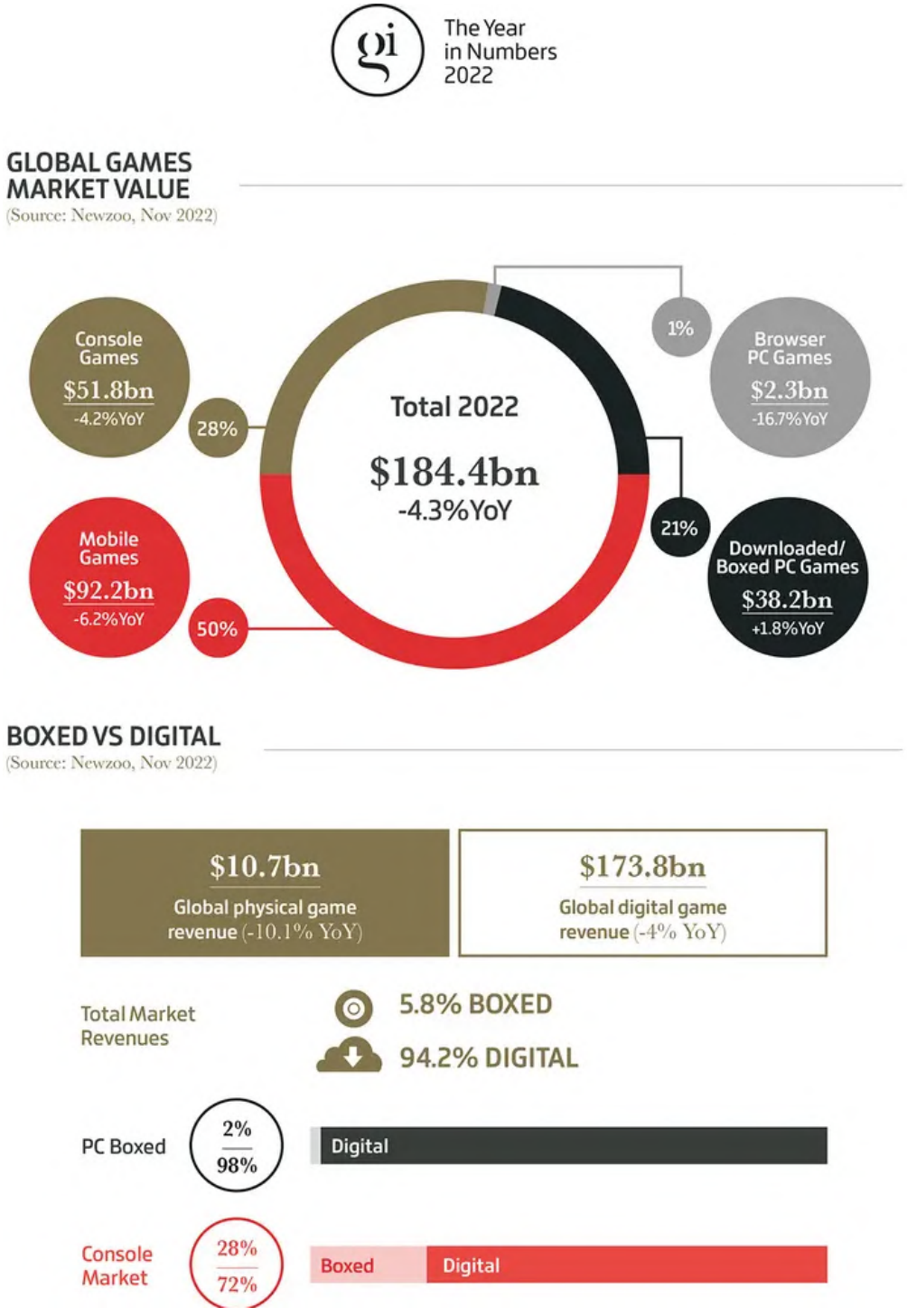

Figure 1: Global Games Market Value 2022 [5].

The report also projected that the games market will grow to $217.9 billion by 2025, driven by the increasing popularity of mobile, cloud, and streaming gaming [5].

Given the significance and potential of the video game industry, many educational institutions that train software engineers have incorporated game development as a part of their curriculum. The main motivations for teaching game development include enhancing the attractiveness and effectiveness of the curriculum [6, 7, 8, 9, 10], preparing graduates for the competitive and demanding game industry [11, 12], fostering teamwork and collaboration skills [13, 14], and developing project management and problem-solving abilities [6, 7].

However, teaching game development is not without challenges and difficulties. Some of the common barriers that hinder the integration of game development courses in higher education are the lack of interdisciplinary skills, time constraints, insufficient interest and expertise among teachers, and the perception that game development is not a serious academic topic [15, 16].

As teachers of Computer Science at the Kryvyi Rih State Pedagogical University, we believe that offering an elective course on computer game development for master's students in the specialty 014.09 Secondary Education (Informatics) is a valuable and rewarding opportunity to increase their motivation, engagement, and professional satisfaction.

The *purpose* of this paper is to share our experience and insights on designing and delivering a course on computer game development using the Unity Game Engine [17], which is one of the most widely used and powerful platforms for creating games across various genres and devices. The aim of this course is to introduce students to the principles and practices of game development using industry-standard software tools. We emphasize problem-solving, project planning, SDK work, and teamwork as essential skills for successful game development. We also view this course as a way to entertain and inspire students to pursue their passion and creativity.

The rest of this paper is organized as follows: Section 2 reviews the related work on teaching game development in higher education. Sections 3 and 4 describes the design and implementation of our course, including its objectives, content, software tools, and teaching methods. Section 5 discusses the challenges and lessons learned from our experience. Section 6 concludes the paper with some recommendations and future directions.

1. Background

The first task of the game development course was to select an approach. Defining the content, goals and objectives of game development is an important step, especially in the light of limited material and time resources.

A review of publications on the subject shows that the implementation of training programmes on game development is quite diverse. It varies from individual courses (Jones [18], Parberry et al. [19], Sweedyk and Keller [20]) and the inclusion of relevant sections in the traditional computer science program (Coleman et al. [21]) before the course sequence (Clark et al. [22], Fachada and Códices [23], Parberry et al. [24], Rocco and Yoder [25], Prokhorov et al. [26]). Content of individual courses from the use of engines developed for training (Gamemaker [7], RPG Maker [6], Alice [27]), development of own game engines (Labyrinth [28, 29], CAGE [30]), technical design [24], Flash [31] to a complete game development training course covering all aspects of

the game [18, 16].

The idea of developing a proprietary engine seems tempting at first, but, in experience, does not pay for itself by the time it takes, and eventually students will never see it again after the course [32]. The real game engine should simplify and speed up the development process and allow students to create interesting games in a short period of time. The problem of finding the most suitable game engine for this course is not very simple, and there are different opinions on this issue from the XNA Game Studio library to Unity and Unreal [32, 33, 34, 35, 36, 19, 37]. Dickson [32] offers to use the Unity game engine [17] to teach game development. Given its widespread use in the industry (de Macedo and Rodrigues [38], Toftedahl and Engström [39]) and even for teaching game development in the middle school [40], this seems logical.

There are also several important CS sections directly used in the development of computer games: the basics of physics, multimedia, network basics, computer graphics, and the basics of game artificial intelligence (Ahlquist and Novak [41], Millington [42], Yannakakis and Togelius [43]).

Game design usually refers to the design of the game and focuses on story, mechanics, character modelling, environment, process content generation, etc., which is enough material to take a whole semester without going into too much detail. There are many textbooks covering these broad topics, such as Adams [44], Ahlquist and Novak [41], Saulter [45], Bond [46]. These areas are compulsory for the course.

2. Selecting the software

Once the approach to the gaming course was defined, the next question we faced was what tools to use to create games.

More recently, developers have made widely available many powerful game engines and development environments that provide functionality for video game development. An overview of some of the best known is presented below.

Godot Engine [47, 48]

Cost and Licensing: Completely free and open source under the permissive MIT license.

System Requirements (minimum): Memory: 4 GB, Graphics Card: NVIDIA GeForce 6200, CPU: Intel Core 2 Duo E8400, OS: Windows 7.

Platforms: Linux, Windows, OS X, Wii, Nintendo 3DS, PlayStation 3, PS Vita, Android, iOS, BBX, web-games with asm.js, NativeClient.

Overview and Features: Godot Engine is a feature-packed, cross-platform game engine to create 2D and 3D games from a unified interface. It provides a comprehensive set of common tools, so users can focus on making games without having to reinvent the wheel. Games can be exported in one click to a number of platforms, including the major desktop platforms (Linux, macOS, Windows) as well as mobile (Android, iOS) and web-based (HTML5) platforms.

Unity Engine [17]

Cost and Licensing: Personal Free version (your project revenue or funding cannot exceed $100,000 a year), Unity Pro package $125 per month (includes an impressive amount of services not included in the free version).

System Requirements (minimum): Graphics Card: DX10, DX11, and DX12-capable GPUs,

CPU: X64 architecture with SSE2 instruction set support, Windows 7 (SP1+) and Windows 10, 64-bit versions only.

Platforms: Android, iOS, Windows Phone 8, BlackBerry, PS3, Xbox360, Wii U and web-browsers.

Overview: Unity is a cross-platform game engine. The engine can be used to create 2D/3D, virtual reality, and augmented reality games, as well as simulations and other experiences (Axon [49], Takahashi [50]). The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering and construction.

Features: Creating and Destroying GameObjects, Access the Components, Events for GameObject, Dealing with Vector Variables and Timing Variables, Physics Oriented Events, Coroutine and Return Types.

Unreal Engin [34]

Cost and Licensing: Free (5% royalty on gross revenue more than $1,000,000),

System Requirements (minimum): CPU: Quad-core Intel or AMD processor, 2.5 GHz or faster, Graphics Card: NVIDIA GeForce 470 GTX or AMD Radeon 6870 HD series card or higher, RAM: 8 GB Windows 7 64-bit or Mac OS X 10.9.2 or later.

Platforms: iOS, Android, Windows Phone 8, Xbox360, PS 3, PlayStation Vita, Wii U.

Overview and Features: Unreal Engine is a complete suite of development tools for anyone working with real-time technology. From design visualizations and cinematic experiences to high-quality games across PC, console, mobile, VR, and AR, Unreal Engine gives you everything you need to start, ship, grow, and stand out from the crowd.

XNA Game Studio [35, 36, 51]

Cost and Licensing: Free download from Microsoft site.

System Requirements (minimum): Graphics Card Shader Model 1.1 support, DirectX 9.0 support, Operating System: Windows Vista SP2, Windows 7 (All editions except Starter).

Platforms: Windows, Xbox 360, Zune.

Overview and Features: XNA Game Studio 2.0 – application framework, integrated development environment. Features: Game component models, New framework library designed to support Microsoft Windows, XBOX 360, and Zune game development, Integration with XNA Framework Content Pipeline.

From an analysis of the capabilities of the video game development tools described, it can be concluded that they are all quite powerful. The choice of a specific tool is determined by the characteristics of the project being developed. Their use for educational purposes is almost equal, although the choice may be influenced by the size of the proposed course.

The second parameter to choose the instrument was its cost. All the tools described are free of charge for educational purposes and thus meet our needs.

The third, perhaps most essential, requirement is compliance with the minimum system requirements of the equipment and associated software. State educational institutions are at a disadvantage in this respect. Therefore, for the first version of the course "Computer game development" in our university was chosen Microsoft XNA Game Studio, which has a narrower range of possibilities.

We assumed that the experience of our students in C/C++ and C# programming would allow them to easily learn XNA. However, we were wrong. By the end of the course, many of them were halfway to the games. The greatest success was achieved by the group of students

who developed the Tower Defence class game, but it was completed as part of the bachelor's qualification work.

The problem with this approach is that in order for students to feel the process of developing games, they need an environment that they can easily use to create games. The focus of the course was to make the game good, not just work at all. We wanted our students to have experience working with a real engine, real skills if they decided to develop games.

The situation improved after the computers at our university were upgraded. We were able to work with a serious game engine. We decided to use the Unity Engine because it has a less steep learning curve than Unreal. It can be used to develop games for any platform, including the Web, for real games, not just training games for learning. Unity scripting can be done in C# or JavaScript, with which our students have already had experience.

3. Organization of the course

We wanted to build the course in such a way that students could learn the basics of Unity quickly enough and focus on creating the game for most of the semester.

After studying Paul E. Dickson's works (Dickson [32], Dickson et al. [33]), our first thought was to build a course based on a book with examples that could guide both us and our students, for example, Unity 3.x Game Development Essentials [52]. One game is built throughout the book, each chapter introduces a new concept and aspect of the game. All examples of code are written in JavaScript and C#. This book quickly gives an idea of colliders, particle systems, etc. for anyone with no experience in game development. The work on the book provides enough information to study the basics of Unity.

One of the problems is the rapid development of Unity and the need to find relevant materials for work. Unity has an active online community that helps to find textbooks to cope with the new features and changes in Unity and could base the course on one of the online textbook series. However, since the duration of the course was only one semester, it was necessary to develop a manual sufficient to carry out the laboratory tasks in order to use the books only as an additional source of information.

Our goal in this course is to give students a sense of the game development process with a focus on project management, teamwork, and problem solving. The first part of the course focuses on teaching students to use Unity, and the second part focuses on developing real play by groups of students. Classes were held for 3 hours per week: 1 hour of lectures and 2 hours of laboratory work. The basic structure of the course is shown in table 1, 2.

The method that we used in the first part of the course, to organize the study of Unity students, was to combine work on the assignments in the classroom with the performance of additional creative tasks by ourselves. In each work, students had to understand in detail what had been done in the classroom in order to determine how to complete the extra assignment. During the first part of the semester, students sought to learn how to solve various problems with Unity before they began working on their final game projects that required these skills. During this work, students built a basic game in which the player could control the movement and actions of the character in their environment.

Table 1
Course part 1. Basics of work in Unity.

Topics	Duration, hours	Course Materials	Deliverables
1. Introduction to Unity	3	Unity features. Examples of games created on Unity. Unity installation. The difference between 2d and 3d design. Overview of the main elements of the scene: Camera, GameObject, Direction Light. Moving the scene. Camera object. Location of objects on a 3d scene.	Laboratory work 1
2. Textures, materials and elements of the scene	3	Adding new textures to the project. Creation and use of materials. Shaders and their use. Work with aggregated characters and their components. Creating a Terrain. Terrain Landscape Editor. Trees, grass and surroundings. Placement of a player on Terrain.	Laboratory work 2
3. Scripts and object movement	3	Install Visual Studio Plug-in for Unity3d. Creating scripts. Apply a script to an object on the stage. The structure of the automatically generated script. Creating a character movement using a script.	Laboratory work 3
4. Player management	3	Using the Asset store. Download unitypackage. Use ready-made unitypackage. Creating unitypackage. The structure of projects created by other developers. Use of ready-made asset. Character Controller and its application. Move the object with the keyboard. Dynamic object creation.	Laboratory work 4
5. User interface	3	User interface and its application. Examples of basic controls. Bindings and orientation of controls relative to the working area of the screen. Creating elementary events. Customize Canvas to different screen resolution properties	Laboratory work 5
6. Animation	3	Using ready-made character animations. Create your own animation. Editing curves. Structure and main properties of the Animation component. Animator component	Laboratory work 6

4. Results

It's hard to measure success when students are building different games. By calling the game playable, we mean that the students have created a mechanic for the game (possibly with minor errors), combined the art assets with the mechanics and made some introduction (history, list of game items) that enters into the game. In order to evaluate the results of our course "Computer game development" we used some parameters offered by Ritzhaupt [1] to evaluate its such course.

Table 2
Course part 2. Game development based on Unity.

Topics	Duration, hours	Course Materials	Deliverables
1. Game Development Basics	1	Game development life-cycle. Game terminology. Overview of game industry	Game Concept plan
2. Creating a character	3	Uploading models to the project. Features of creating game characters. Customize avatars for models that use humanoid animations. Working with the Animator component. Animator controller settings. Retargeting of humanoid animated clips.	Characters modelling and animation
3. Finding a way	3	Creating a game scene. Navigation grid settings. Add and adjust obstacles. Implementation of the movement of the character on the navigation grid.	Group projects element
4. Inverse kinematics	5	Animation settings. Attaching skeletal parts to objects. Creating a script to work with inverse kinematics. Fixation of skeleton points. LineRender component.	Group projects element
5. Characters not controlled by the player	6	Creating a slider and stylizing it. Move the coordinates of the slider to the position above the target. Creating goal health scripts. Using Raycast.	Group projects element
6. Construction of game levels	12	Creating a game level. Overlay post effects on the main camera. Set up bots to search for enemies. Game level layout. Creating multiple teams. Configuration and error correction. Possibility of application of scattering of bullets at shooting.	Final Game

4.1. Usefulness of course elements for students

For studying the elements of the course that proved successful, we asked the students to indicate which elements of the course were useful for learning in the range from 1 – "not useful" to 5 – "very useful" (table 3). Of particular interest are the highly rated elements: teamwork in labs ($M = 4.05$; $SD = 0.71$), working with peers inside and outside of class ($M = 3.9$; $SD = 0.9$), and the hands-on labs activities ($M = 4.03$; $SD = 0.92$). These results underline the importance of sufficient work in the computer laboratory and cooperative training in the game development course.

4.2. Student assessment of gains

Students were asked to evaluate their post-graduate achievements in a number of areas related to the development of games on a scale of 1 to 5 (table 4). The results showed that they made the most progress in understanding the game's development ($M = 4.03$; $SD = 0.83$) and the ability to use the Unity Engine ($M = 4.18$; $SD = 0.87$). In all other areas, progress has also been above average.

Table 3
Useful course element percentages, mean, and standard deviation.

Useful Elements	1	2	3	4	5	M	SD
The way in which the material was approached	0	5	40	32,5	22,5	3,73	0,88
The pace at which we worked	2,5	10	45	30	12,5	3,4	0,93
Working with peers inside and outside of class	0	7,5	22,5	42,5	27,5	3,9	0,9
Viber discussion group	2,5	7,5	32,5	40	17,5	3,63	0,95
Teamwork in labs	0	2,5	15	57,5	25	4,05	0,71
The presentation of the final group project	0	10	20	45	25	3,85	0,92
The hands-on labs activities	0	5	25	32,5	37,5	4,03	0,92

Table 4
Student learning gains percentages, and mean.

Student Gains from Course	1	2	3	4	5	M	SD
Understanding the main concepts in game development	0	5	25	42,5	27,5	3,93	0,86
Understanding the game development process	2,5	2,5	10	60	25	4,03	0,83
Understanding Unity Engine using in game development	0	5	15	37,5	42,5	4,18	0,87
Ability to think through a problems in game development	0	1	10	20	9	3,93	0,76
Confidence in your ability to work in game development	0	5	25	47,5	22,5	3,88	0,82
Feeling comfortable with complex game development	0	2,5	35	45	17,5	3,78	0,77

4.3. Final project game

In the second part of the course, students worked in groups (3–4) to create final game projects. We allow students to decide for themselves which games they want to develop and how to split into groups. Each group decided who would play what roles and what they would need to do to finish the game. Lectures on this part of the course covered a wide range of topics. Some specific aspects of game development that students are likely to need were discussed. All practical tasks for this part of the course are related to keeping students on their way to finishing the final project games. These include students presenting game ideas, project plans, vertical slices, usability tests, a final game, and weekly reports on who has achieved what.

Most of the groups were able to successfully build a playable game for the final project, which is significantly better than the previous version of the course. Students created RPG games (figure 2), quest games (figure 3), logical games (figure 4) and action games (figure 5). The variety of these games shows that students are free to create games of their choice instead of being limited to the genre and content given by the teacher.

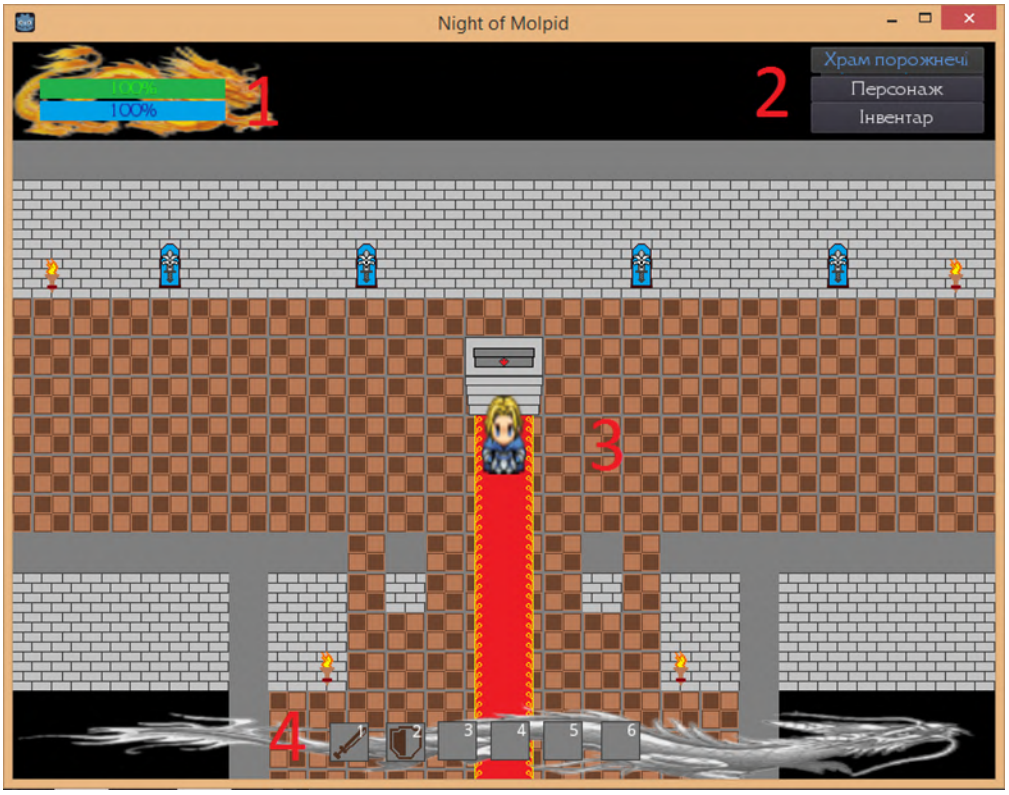

Figure 2: RPG game.

Figure 3: Quest game.

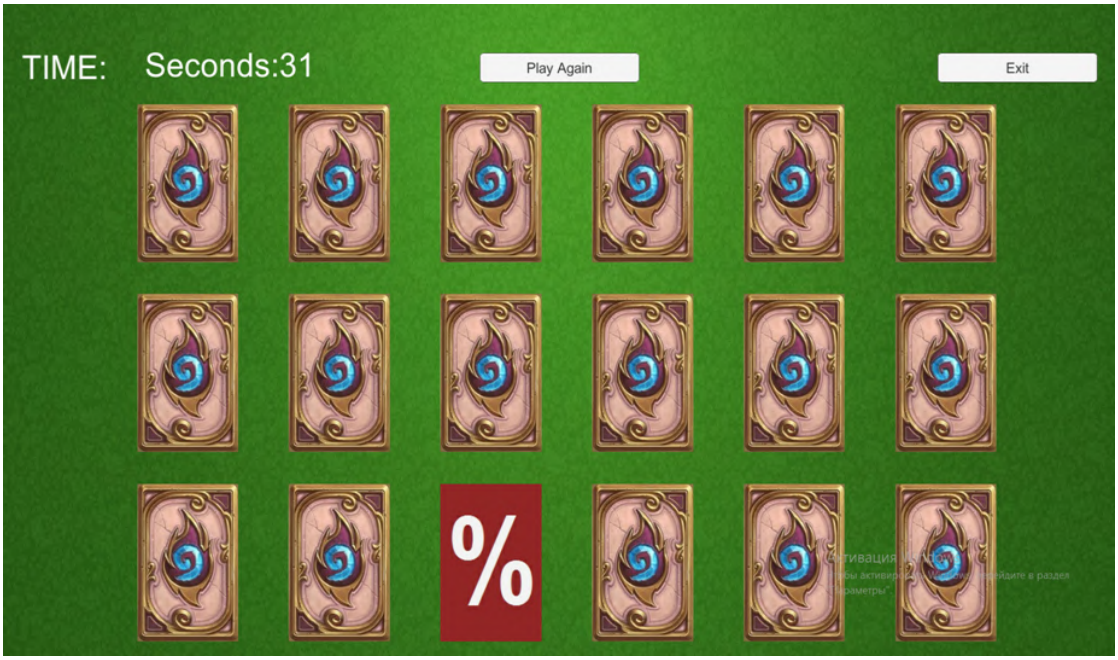

Figure 4: Logical game.

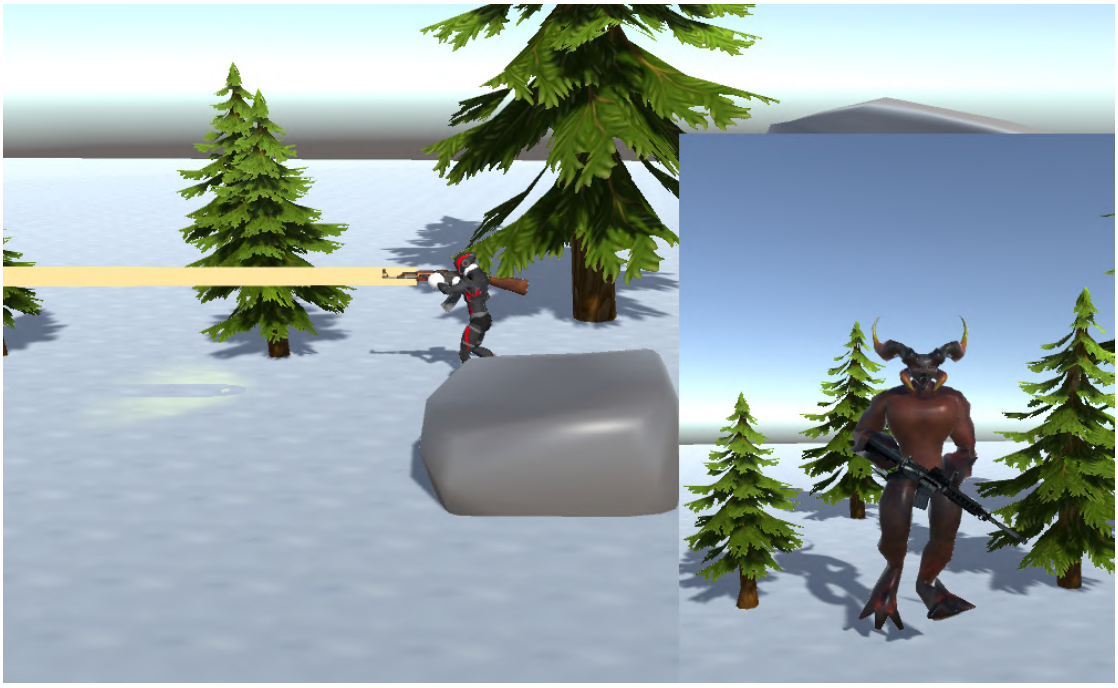

Figure 5: Action game.

5. Conclusions

In this paper, we have presented our experience and evaluation of teaching a course on computer game development using the Unity Game Engine for master's students in the specialty 014.09 Secondary Education (Informatics) at the Kryvyi Rih State Pedagogical University. We have described the design and implementation of the course, as well as the challenges and outcomes of its delivery. We have also assessed the course using a framework proposed by Ritzhaupt [1] based on student feedback and learning outcomes.

We have found that our course was successful in achieving its objectives and enhancing students' knowledge and skills in game development. We have also observed that students were highly motivated, engaged, and satisfied with the course. We have identified some areas for improvement, such as providing more guidance and feedback, balancing the workload and difficulty, and diversifying the assessment methods.

We have concluded that the Unity Game Engine is a suitable platform for teaching game development in secondary education, as it offers a low barrier to entry, a rich set of features, a cross-platform compatibility, and a wide adoption in the game industry. We have also argued that a team-based approach is beneficial for fostering collaboration and creativity among students.

We have also reflected on the pedagogical implications of teaching game development in secondary education. We have suggested that teaching game development requires a shift from a teacher-centred to a learner-centred environment, where students have more autonomy and control over their learning process and teachers act as facilitators and mentors.

We hope that our paper will inspire and inform other teachers who are interested in teaching game development in secondary education. We also hope that our paper will contribute to the growing body of research on game development education and its impact on student learning and motivation.

References

[1] A. D. Ritzhaupt, Creating a game development course with limited resources, ACM Transactions on Computing Education 9 (2009) 1–16. doi:10.1145/1513593.1513596.

[2] T. A. Vakaliuk, V. V. Kontsedailo, D. S. Antoniuk, O. V. Korotun, I. S. Mintii, A. V. Pikilnyak, Using game simulator Software Inc in the Software Engineering education, in: A. E. Kiv, M. P. Shyshkina (Eds.), Proceedings of the 2nd International Workshop on Augmented Reality in Education, Kryvyi Rih, Ukraine, March 22, 2019, volume 2547 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 66–80. URL: https://ceur-ws.org/Vol-2547/paper05.pdf.

[3] O. M. Haranin, N. V. Moiseienko, Adaptive artificial intelligence in RPG-game on the Unity game engine, CEUR Workshop Proceedings 2292 (2018) 143–150.

[4] O. O. Katsko, N. V. Moiseienko, Development computer games on the Unity game engine for research of elements of the cognitive thinking in the playing process, CEUR Workshop Proceedings 2292 (2018) 151–155.

[5] J. Batchelor, GamesIndustry.biz presents... The Year in Numbers 2022, 2022. URL: https://www.gamesindustry.biz/gamesindustrybiz-presents-the-year-in-numbers-2022.

[6] T. Barnes, H. Richter, E. Powell, A. Chaffin, A. Godwin, Game2learn: Building cs1 learning games for retention, ACM SIGCSE Bulletin 39 (2007) 121–125. doi:10.1145/1269900.1268821.

[7] K. Claypool, M. Claypool, Teaching software engineering through game design, ACM SIGCSE Bulletin 37 (2005) 123–127. doi:10.1145/1151954.1067482.

[8] B. B. Morrison, J. A. Preston, Engagement, ACM SIGCSE Bulletin 41 (2009) 342–346. doi:10.1145/1539024.1508990.

[9] T. E. Roden, R. LeGrand, Growing a computer science program with a focus on game development, in: Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE'13, ACM Press, 2013. doi:10.1145/2445196.2445362.

[10] K. Sung, Computer games and traditional CS courses, Communications of the ACM 52 (2009) 74–78. doi:10.1145/1610252.1610273.

[11] O. M. Haranin, O. O. Katsko, N. V. Moiseienko, Developer software tools in a course "Development of computer games", New computer technology 15 (2017) 160–163.

[12] T. A. Vakaliuk, V. Kontsedailo, D. Antoniuk, O. Korotun, S. Semerikov, I. S. Mintii, Using Game Dev Tycoon to Create Professional Soft Competencies for Future Engineers-Programmers, in: O. Sokolov, G. Zholtkevych, V. Yakovyna, Y. Tarasich, V. Kharchenko, V. Kobets, O. Burov, S. Semerikov, H. Kravtsov (Eds.), Proceedings of the 16th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. Volume II: Workshops, Kharkiv, Ukraine, October 06-10, 2020, volume 2732 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 808–822. URL: https://ceur-ws.org/Vol-2732/20200808.pdf.

[13] Q. Brown, F. Lee, S. Alejandre, Emphasizing soft skills and team development in an educational digital game design course, in: Proceedings of the 4th International Conference on Foundations of Digital Games, FDG '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 240–247. doi:10.1145/1536513.1536557.

[14] Y. Rankin, A. Gooch, B. Gooch, The impact of game design on students' interest in CS, in: Proceedings of the 3rd international conference on Game development in computer science education - GDCSE'08, ACM Press, 2008. doi:10.1145/1463673.1463680.

[15] K. Becker, J. R. Parker, Serious Games + Computer Science = Serious CS, Journal of Computing Sciences in Colleges 23 (2007) 40–46. doi:10.5555/1292428.1292436.

[16] J. Martin, C. Smith, A cross-curricular team based approach to game development, Journal of Computing Sciences in Colleges 17 (2002) 39–45. doi:10.5555/775009.775019.

[17] U. Technologies, Unity Real-Time Development Platform | 3D, 2D VR & AR Engine, 2021. URL: https://unity.com.

[18] R. M. Jones, Design and implementation of computer games, ACM SIGCSE Bulletin 32 (2000) 260–264. doi:10.1145/331795.331866.

[19] I. Parberry, T. Roden, M. B. Kazemzadeh, Experience with an industry-driven capstone course on game programming, in: Proceedings of the 36th SIGCSE technical symposium on Computer science education - SIGCSE'05, ACM Press, 2005. doi:10.1145/1047344.1047387.

[20] E. Sweedyk, R. M. Keller, Fun and games, ACM SIGCSE Bulletin 37 (2005) 138–142. doi:10.1145/1151954.1067485.

[21] R. Coleman, M. Krembs, A. Labouseur, J. Weir, Game design & programming concentration

within the computer science curriculum, ACM SIGCSE Bulletin 37 (2005) 545–550. doi:10.1145/1047124.1047514.

[22] B. Clark, J. Rosenberg, T. Smith, S. Steiner, S. Wallace, G. Orr, Game development courses in the computer science curriculum, Journal of Computing Sciences in Colleges 23 (2007) 65–66. doi:10.5555/1292428.1292440.

[23] N. Fachada, N. Códices, Top-down design of a CS curriculum for a computer games BA, in: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ACM, 2020. doi:10.1145/3341525.3387378.

[24] I. Parberry, M. B. Kazemzadeh, T. Roden, The art and science of game programming, in: Proceedings of the 37th SIGCSE technical symposium on Computer science education - SIGCSE'06, ACM Press, 2006. doi:10.1145/1121341.1121500.

[25] D. Rocco, D. Yoder, Design of a media and gaming sequence for graduates in applied CS, Journal of Computing Sciences in Colleges 22 (2007) 131–137.

[26] O. V. Prokhorov, V. O. Lisovichenko, M. S. Mazorchuk, O. H. Kuzminska, Implementation of digital technology for student involvement based on a 3D quest game for career guidance and assessing students' digital competences, Educational Technology Quarterly 2022 (2022) 366–387. doi:10.55056/etq.430.

[27] L. Werner, S. Campe, J. Denner, Children learning computer science concepts via Alice game-programming, in: Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE'12, ACM Press, 2012. doi:10.1145/2157136.2157263.

[28] J. Distasio, T. Way, Inclusive computer science education using a ready-made computer game framework, in: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE'07, ACM Press, 2007. doi:10.1145/1268784.1268820.

[29] G. A. Shultz, The story engine concept in CS education, Journal of Computing Sciences in Colleges 20 (2004) 241–247. doi:10.5555/1040231.1040263.

[30] J.-M. Vanhatupa, Game engines in game programming education, in: Proceedings of the 11th Koli Calling International Conference on Computing Education Research - Koli Calling'11, ACM Press, 2011. doi:10.1145/2094131.2094156.

[31] A. Estey, J. Long, B. Gooch, A. A. Gooch, Investigating studio-based learning in a course on game design, in: Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG'10, ACM Press, 2010. doi:10.1145/1822348.1822357.

[32] P. E. Dickson, Using Unity to teach game development, in: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ACM, 2015. doi:10.1145/2729094.2742591.

[33] P. E. Dickson, J. E. Block, G. N. Echevarria, K. C. Keenan, An experience-based comparison of Unity and Unreal for a stand-alone 3D game development course, in: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, ACM, 2017. doi:10.1145/3059009.3059013.

[34] The most powerful real-time 3D creation platform - Unreal Engine, 2021. URL: https://www.unrealengine.com.

[35] J. Harris, Teaching Game Programming Using XNA: What Works and What Doesn't, Journal of Computing Sciences in Colleges 27 (2011) 174–181.

[36] J. Linhoff, A. Settle, Teaching game programming using XNA, in: Proceedings of the

13th annual conference on Innovation and technology in computer science education - ITiCSE'08, ACM Press, 2008. doi:10.1145/1384271.1384338.

[37] C. Peng, Introductory game development course: A mix of programming and art, in: 2015 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2015. doi:10.1109/csci.2015.152.

[38] D. V. de Macedo, M. A. F. Rodrigues, Experiences with rapid mobile game development using Unity engine, Computers in Entertainment 9 (2011) 1–12. doi:10.1145/2027456.2027460.

[39] M. Toftedahl, H. Engström, A taxonomy of game engines and the tools that drive the industry, in: DiGRA 2019, The 12th Digital Games Research Association Conference, Kyoto, Japan, August, 6-10, 2019, Digital Games Research Association (DiGRA), DiGRA, 2019. URL: http://www.digra.org/wp-content/uploads/digital-library/DiGRA_2019_paper_164.pdf.

[40] O. Comber, R. Motschnig, H. Mayer, D. Haselberger, Engaging students in computer science education through game development with Unity, in: 2019 IEEE Global Engineering Education Conference (EDUCON), IEEE, 2019. doi:10.1109/educon.2019.8725135.

[41] J. B. Ahlquist, J. Novak, Game Development Essentials: Game Artificial Intelligence, Cengage Learning, 2007.

[42] I. Millington, AI for Games, CRC Press, 2019. doi:10.1201/9781351053303.

[43] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, Springer International Publishing, 2018. doi:10.1007/978-3-319-63519-4.

[44] E. Adams, Fundamentals of game design, 4rd ed., New Riders, 2013.

[45] J. Saulter, Introduction to video game design and development, McGraw-Hill, New York, 2007.

[46] J. G. Bond, Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#, Addison-Wesley Professional, 2014.

[47] C. Bradfield, Godot Engine Game Development Projects: Build five cross-platform 2D and 3D games with Godot 3.0, Packt Publishing, Birmingham, 2018.

[48] A. Manzur, G. Marques, Godot Engine Game Development in 24 Hours, Sams Publishing, Indianopolis, 2018.

[49] S. Axon, Unity at 10: For better—or worse—game development has never been easier, 2016. URL: https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/.

[50] D. Takahashi, John riccitiello q&a: How unity ceo views epic's fortnite success, 2018. URL: https://venturebeat.com/2018/09/15/john-riccitiello-interview-how-unity-ceo-views-epics-fortnite-success/.

[51] R. Miles, Microsoft XNA Game Studio 4.0: Learn Programming Now!, Pearson Education, 2011.

[52] W. Goldstone, Unity game development essentials, Packt Publishing Ltd, 2009.