Design and development of a game application for learning Python

Vasyl P. Oleksiuk[1,2], Dmytro V. Verbovetskyi[2] and Ivan A. Hrytsai[1]

[1]Ternopil Volodymyr Hnatiuk National Pedagogical University, 2 Maxyma Kryvonosa Str., Ternopil, 46027, Ukraine
[2]Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine

Abstract

This article explores the design of a Python learning game application, presenting outcomes in meeting its objectives. The study involves an analysis of various experiences in game-based learning, leading to the substantiation of requirements and the development of a model for the application. The authors identified the basic requirements for a game application, such as relevance to educational objectives, incorporation of game mechanics, the ability to write and debug code in the application, individualized passing of the game, and the ability to report and rate learning outcomes of students. The authors identified the basic requirements for a game application, such as relevance to educational objectives, incorporation of game mechanics, the ability to write and debug code in the application, individualized passing of the game, and the ability to report and rate learning outcomes of students. Unity3D emerges as the chosen engine following comparative analysis, considering graphic development, interaction, and user-friendliness. The authors of the article described their own experience in developing the PythonLeaner game. The designing process of this game emphasises the incorporation of game elements with both test and open-ended tasks to enhance programming education. The paper contains some fragments of the game workspace and code of a gaming application. The PythonLeaner's game limitations of the game and further directions for overcoming them for improving the application are indicated.

Keywords

game-based learning, game engine, Unity, Unreal Engine, CryEngine, game, game development,

1. The problem statement

Today, in the digital age, artificial intelligence game-based educational tools are widely used. Teachers use them as desktop and mobile applications, cloud services, LMS modules, etc. [1, 2]

The design of game applications to teach many science subjects especially computer science offers significant benefits, making it highly relevant for several reasons. First of all by utilising game applications for educational purposes, instructors are able to capture and sustain pupils' interest and motivation, simplifying the process of immersion into the subject matter [3]. Furthermore, game-based learning promotes active engagement, providing an effective method to

CS&SE@SW 2023: 6th Workshop for Young Scientists in Computer Science & Software Engineering, February 2, 2024, Kryvyi Rih, Ukraine

oleksyuk@fizmat.tnpu.edu.ua (V. P. Oleksiuk); verbovetskyj.dv@gmail.com (D. V. Verbovetskyi); grytsaj_ia@fizmat.tnpu.edu.ua (I. A. Hrytsai)
https://tnpu.edu.ua/faculty/fizmat/oleksyuk-vasil-petrovich.php (V. P. Oleksiuk)
0000-0003-2206-8447 (V. P. Oleksiuk); 0000-0002-4716-9968 (D. V. Verbovetskyi); 0000-0001-6261-8429 (I. A. Hrytsai)

encourage student participation [4]. Students should participate in the game, solve problems and make decisions to encourage hands-on, experiential learning, which is a key aspect of mastering programming concepts. Games commonly require problem-solving, logic, algorithms, and critical thinking [5]. They are fundamental skills in science learning and programming. Game applications have the ability to simulate real-life scenarios and challenges that programmers are likely to face in their future careers. This feature helps students to connect theoretical knowledge with practical application. Games offer prompt feedback, allowing students to observe the consequences of their choices and make necessary modifications. This iterative learning process is in line with the problem-solving nature of programming. Many games feature a gradual difficulty curve, enabling students to begin with simple tasks and gradually advance to more intricate challenges. This approach mirrors the incremental learning method often used in programming [6].

Multiplayer modes of modern games can facilitate the development of teamwork and communication skills, which are vital in software development and real-world projects. Competitive games can inspire productive competition, motivating players to enhance their abilities [7]. Self-paced learning is also a key component of this approach. Game-based learning enables self-paced learning, enabling learners to progress through the content at their own speed, thus accommodating diverse learning styles and abilities. Some game-based platforms include coding elements, which allow learners to script or modify game behaviour. This hands-on exposure to coding strengthens programming concepts and syntax. Game-based learning can offer increased accessibility as it caters to a range of learning styles and can be tailored to suit different abilities and skill levels [8]. Moreover, it can provide better retention of information over the long-term. When students derive enjoyment from the learning experience, they tend to remember and employ what they have learned more effectively.

Although numerous ready-made tools are available on the internet [9], the issue of creating new instances remains relevant. So the aim of this research is to develop a model of a game application for learning Python. To achieve this, the following tasks need to be solved:

1. Analyse the experience of using game applications in teaching programming.
2. Describe the requirements for the application and design its model.
3. Select tools for application development.
4. Describe and analyse the key points of application development.

2. The model of the game application for learning Python

Based on extensive research by Priyaadharshini et al. [10], Kroustalli and Xinogalos [11], Papadakis et al. [12], Osadcha and Osadchyi [13], Yatsenyak et al. [14] it can be concluded that powerful learning game applications must meet specific, key requirements.

- The application should possess well-defined educational objectives, including imparting knowledge of programming concepts, problem-solving abilities, and coding languages, in accordance with established academic standards.
- The application should have elements of gamification. It is necessary to incorporate game mechanics, including challenges, rewards and progression systems, to enhance the

learning experience by making it engaging and enjoyable. Therefore, the game should be based on an engaging narrative or storyline that contextualises the learning content in the game.

- The application should be in a Hands-on style. For example offers chances for learners to write and debug code within the application. It is advisable to integrate coding challenges, puzzles, or issues that necessitate learners to put programming principles into practice.
- The game should be played individually by each student. To develop programming skills, the game should be played over several levels of increasing complexity. The tasks should comprehensively cover a range of programming concepts, from basic syntax to more advanced ideas, catering to the intended audience.
- Additionally, the learning application utilizes analytics and reporting to provide insightful measures of progress. Provide educators and learners with performance data and identify areas for improvement.
- It would be good if the app is available across multiple platforms (web, mobile, and desktop) to maximize the potential audience. In particular, the game must perform effectively on various screen sizes and devices.
- The application should permit users to access and learn from the application offline, allowing it to be utilized in regions with limited or no internet connectivity.

To develop the game model, the content, goal, objective, tasks and scheme of passing this game were developed. The aim of the game is to facilitate game-based learning [11, 15, 16] for first-year students taking the Python programming course. The PythonLeaner game we are developing should provide the student with several modes. They are implemented by menu items such as:

1. New game.
2. Continue.
3. Shop.
4. Exit.

The model of the game is shown in figure 1.

The item “New game” displays to the user a greeting. It introduces the player to the purpose of the game and introduces its rules. The next screen shows an assistant who will accompany the player throughout the game. The assistant has the appearance of a stylized snake and familiarizes the player with the game process in more detail. As can be seen from figure 1 the PythonLeaner game provides completing by users several levels (stages). There are 3 tasks in each level. After successful completion of each of the tasks points are awarded to the student. At any moment of the game, the player can open a “shop” purchase a computer upgrade and stylize its appearance.

If the player gives an incorrect answer during the tasks, the assistant will provide a hint that will help to get to the correct answer. The game process consists of theory and practice, the incentive element is that the player receives points after correct answers, which can be spent on the graphic design of the working environment.

During the game, the player has some options such as improving the appearance of his/her own computer. In particular, he/she can buy PC peripherals of a different appearance and can

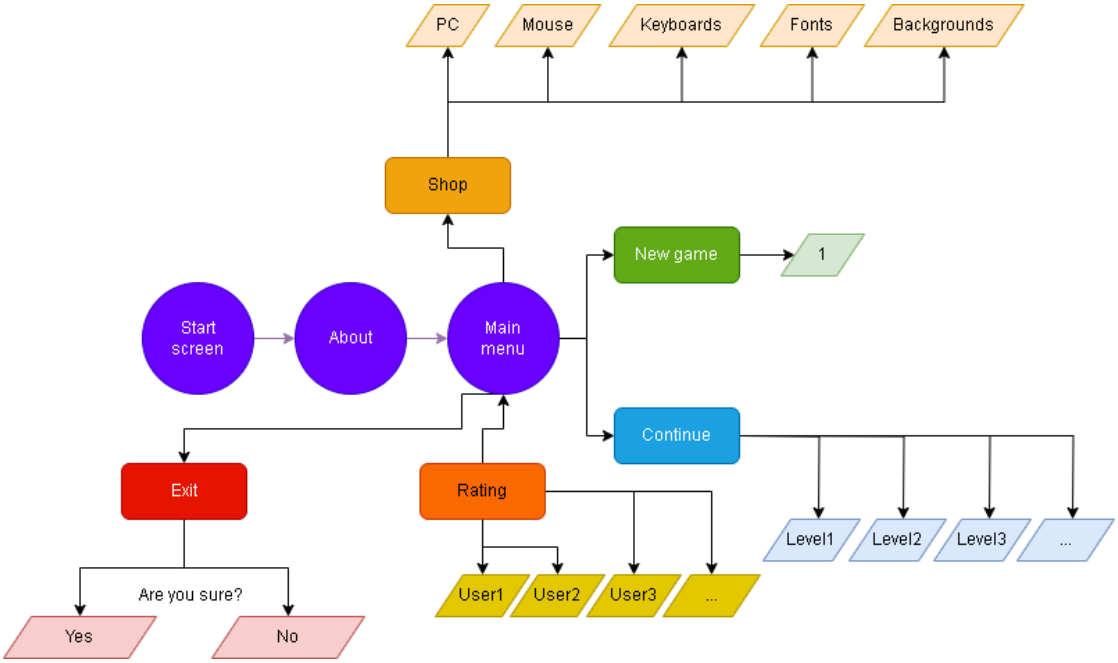

Figure 1: The model of the PythonLeaner game.

change the output devices (monitor, mouse, keyboard) the terminal font and the background of the working environment.

Moreover, it implemented an element of computer freezing. At the start levels, the user's computer freezes when processing the answer to a test question. The performance of the computer can be increased in the store by using the collected points. Finally, after completing all the levels, the player can get an infinite number of points and can switch between all the allowed interfaces. All player results are stored in the cloud. They are used to generate a ranking (figure 2), which is also available to players.

This approach makes it possible to implement a competitive approach to learning programming.

3. Programming environments for developing game applications

Each game is developed on a specific game engine and unified according to specific tasks. Action games are among the most popular [17]. This genre has a lot of subgenres such as platform games, fighting games, and, of course, shooting or shooters, which have not lost their popularity since the 1990s. This genre also includes arcade games, survival, sports games, simulators, and racing. The next most popular voluminous genre with a large number of subgenres is strategies. Strategies are economic, defensive towers, military, card. And the last of the most popular genres of games are role-playing games or RPGs [18]. These include such subgenres as educational, adventure, quests, puzzles, browser games, network text and network role-playing games. The game we are designing can be categorised as an educational quest.

Figure 2: The rating of PythonLeaner's gamers.

Let's consider in more detail some engines for creating games such as Unity, Unity3D, and CryEngine [19]. *Unity* environment makes it very convenient to use a 2D template to quickly start working on a project. This template is accessible via the Unity Hub and features default configurations, such as [20, 21].

1. A default scene that uses 2D mapping and contains the camera in orthographic mode with a single-colour sweep of the previous frame (not a Skybox, as is typical for 3D projects).
2. The editor is set to 2D mode by default, so new texture assets are imported as Sprites.
3. The global lighting mechanism is disabled.
4. A number of installed packages, including 2D Animation, 2D Pixel Perfect, 2D PSD Importer and 2D SpriteShape, as well as required dependencies.

Unity allows you to develop games for various platforms such as Windows, macOS, Linux, Android, iOS, Xbox and PlayStation consoles, virtual reality and others. This engine supports several programming languages, including C# and JavaScript. Engine has a large and active developer community, and there are plenty of resources such as forums, tutorials, and documentation to help you solve problems and learn new things. Unity provides a free version for small teams and independent developers. This promotes the spread of the engine and allows startups and individual developers to effectively enter the game development industry.

The platform is constantly being updated. Its developers adding new features and improvements. The regular update releases allow programmers to take advantage of the latest features. Unity provides powerful tools for working with graphics and animation, including realistic lighting effects, a particle system, and tools for modeling 3D scenes. Unity supports virtual reality and augmented reality well, allowing developers to create immersive games and applications for a variety of devices. The platform has become one of the most popular game engines. It is so due to its affordability, powerful tools, and extensive opportunities for developers. The advantages of this engine are:

- profitable solution from a financial point of view, since there is a free version of the program;

- easy mastering of the engine thanks to a wide selection of video lessons on the Internet;
- a fairly wide community between engine developers and users, so developers quickly correct errors and shortcomings in its operation.

A drawback of Unity is the amount of space that any game written using this engine takes up. For example, even a simple pixel computer game takes up quite a lot of computer RAM. The Cities Skylines, Hearthstone, Call of Duty and others well known games have been developed using Unity.

Unreal Engine is a powerful game engine used to develop video games and interactive virtual environments. Drivers are trying to make the most efficient use of computer architecture to optimize their driver's performance. Since 2015, Unreal Engine 4 has become free for game developers. Known for its powerful graphics and photorealistic rendering quality, this engine uses its own rendering engine that supports various lighting effects and shaders. Unreal Engine uses the C++ for game logic and development. Users can use either C++ or the built-in scripting language Blueprint. The second one is a visual programming language. The platform allows developers to create games for various platforms, including Windows, macOS, Linux, Android, iOS, Xbox and PlayStation consoles. There is extensive documentation and tutorials to help developers learn and improve their skills. Unreal Engine supports virtual and augmented reality well. It provides tools to create impressive VR games and applications [22]. Unreal Engine is free to use. Game developers only pay if their product is successful in the marketplace. The advantages of this engine are:

- a large community of developers and, as a result, a considerable number of video materials to illustrate the development process;
- item technical support at a fairly high level;
- debugged update mechanism;
- at least one new tool for work is added during each update;
- a large set of tools for creating animations.

However, there are also drawbacks to using Unreal Engine, including its unsuitability for developing simplistic games and the requirement to pay a five percent tax after the game becomes profitable.

CryEngine is a commercial game engine from Crytek. This engine offers a large set of tools for creating a real-time PC game. Planform contains many advanced graphics, physics and animation technologies, as well as many gameplay improvements. CryEngine is famous for its impressive graphics and detailing of large open worlds. It uses a powerful rendering engine and supports a wide range of impressive effects such as shadows and lighting. Platform supports virtual reality, allowing developers to create immersive VR projects and games [23]. Engine specializes in creating large, open and immersive worlds that allow players to freely explore the game. CryEngine supports game development for various platforms such as PC, Xbox and PlayStation consoles. Engine has a large developer community and documentation to help beginners and professionals use the engine. CryEngine has been used to create famous games such as Far Cry, Crysis, and the Robinson: The Journey series. The engine includes a powerful physics system that allows you to realistically simulate the movement and interaction of objects in the game. CryEngine is defined by its ability to create stunning open worlds and realistic

graphics. This makes it popular with developers looking to create the most immersive gameplay experiences. Game developers and researchers highlight the following advantages of this engine [24, 25]:

- some functions such as Flowgraph and Fmod provide to developers with excellent graphics;
- the platform provides simple mechanisms for creating artificial intelligence;
- this engine is more advantageous in terms of water and weather effects, which can make it suitable for games with realistic atmosphere;
- support for realistic modeling of destruction and object physics, which can be useful for creating interactive and dynamic game worlds.

Among the drawbacks, significant ones include enduring prolonged wait times for technical support with the free version, a community of users growing at a sluggish pace, and obsolete or absent documentation for numerous modules [26].

Table 1
Comparison of game engines Unity3D, Unreal Engine, CryEngine.

Criterion	Unity3D	Unreal Engine	CryEngine
Ease of learning	High	Average	High
Cost	Free	Free	License
Community and support	Big and active	Big and active	Present
Extensions and modding	Yes	Yes	Yes
Compatibility and ecosystem	Universal	Wide functionality	Universal
Multi-platform	Yes	Yes	Yes
Productivity and optimization	Depends on the developer	Depends on the developer	Capable of optimization
Multiplayer and network support	Yes	Yes	Yes
Artificial Intelligence	Yes	Yes	Yes

Based on the data from table 1, we have selected the Unity3D platform for developing the PythonLeaner game.

4. Analysis of key points of the game development

The development of the game took place in several stages, including the design of scenes (greetings, shop, selection of levels, etc.), filling of scenes, creation of scripts for transition between scenes, selection of levels, and creation of level scenes. When logging into the game, the user has the option to register and start a new game. Players who are already registered can log in with their login and password and resume the game from the level they have previously completed. User data, including login and password, as well as their rating, are stored using Firebase web hosting. Firebase is Google's mobile and web development platform that assists

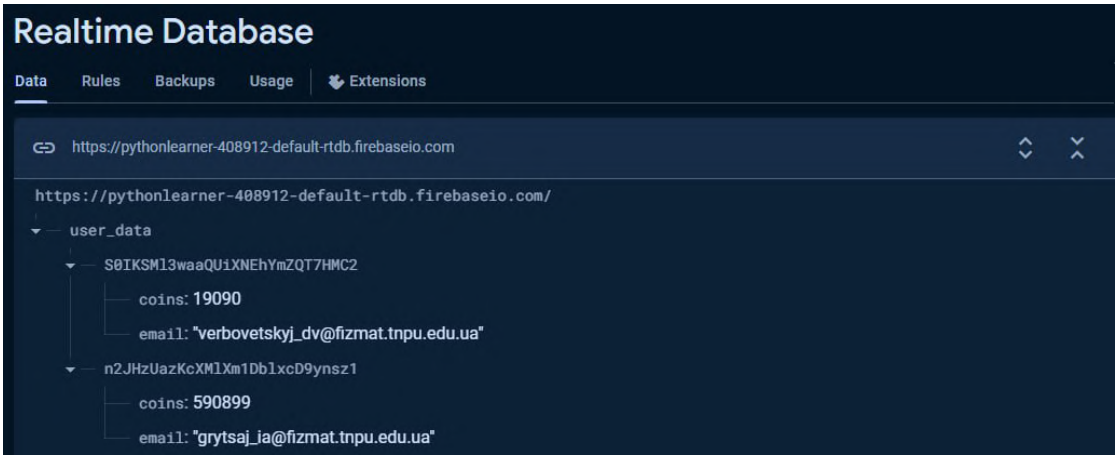

Figure 3: Rating saved in Firebase.

developers in building and scaling software products. The database for storing the rating required the installation of the SDK (FirebaseDatabase.unitypackage), as shown in the figure 3.

The game also has an option to choose the language of the interface. Users can switch between Ukrainian and English interface languages.

The IdleAnimation(), LikeAnimation(), DislikeAnimation() methods were used to implement the action of assistant (figure 4).

Figure 4: Reaction to the player's response.

These functions implement animation in three states of the assistant: "calm", "like", and "dislike". Each state changes the sprite. The Update() method is a main component in the code and is called with every frame. The method processes player keystrokes and updates the displayed text accordingly. This applies to deleting a character, wrapping a line, and other keys that may cause changes to the text. The following method, OnButtonClick(), is triggered when the player presses a button in the game. It compares the text entered by the player with the expected result. If the text is correct, the "like" animation is triggered. Otherwise, the "dislike" animation is displayed. The method performs changes in the game such as going to the next

scene.

To create the game, static variables are also used. One of them is coins (a static variable that stores the number of coins in the game and is available globally to all other objects and scripts). Others variables are pcPositions, pcSizes, promptPositions, promptSizes. These static lists of vectors and sizes define positions and sizes of objects in the game, such as computers. The selectedPCIndex is a static variable that stores the index of the selected computer in the list. The TextMeshProUGUI:mesh is a text field used to display the number of coins in the game.

An example of using the SetPCSizeAndPositionScript method is shown in Listing 1.

Listing 1: SetPCSizeAndPositionScript method

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class SetPCSizeAndPositionScript : MonoBehaviour {
    void Start()  {
        RectTransform t = GetComponent<RectTransform>();
        t.sizeDelta = SingletonScript.pcSizes
                      [SingletonScript.selectedPCIndex];
        t.localPosition = SingletonScript.pcPositions
                          [SingletonScript.selectedPCIndex];
    }
    // Update is called once per frame
    void Update() {
    }
}
```

The project and gameplay have been depicted in figure 5. The user receives a default computer in the first level. The figure illustrates the default keyboard, mouse, font, terminal and workplace backgrounds, and an assistant (snake) located at the top right corner of the workplace. The player can request a hint from the assistant, and the snake's response depends on the student's answer. As you can see from the figure 5, the PythonLearner game has not only a desktop but also a mobile interface.

Tasks for the PyntonLeaner game are of several types such as tests (with one and several answers), entering short answers and writing Python's code. The essence of the game is that the player sees the screen of an old computer and an "improvised programming environment/terminal" is opened on it. During the first level, the player has an old computer a normal room background and a delay of 2.5 seconds when pressing the answer key, however, when completing the following levels, there is an opportunity to upgrade it by changing the appearance and reducing the delay when pressing the keys on the keyboard. The delay is implemented in such a way that when choosing each subsequent computer with a better design, its performance also increases. That is, each subsequent one from the computer by default has a delay of 0.5 seconds less than the previous one. The implementation of the key delay is described in listing 2. Each level consists of three types of tasks, including writing code. During the development of the project, it was decided to adapt the application so that new tasks could be created for

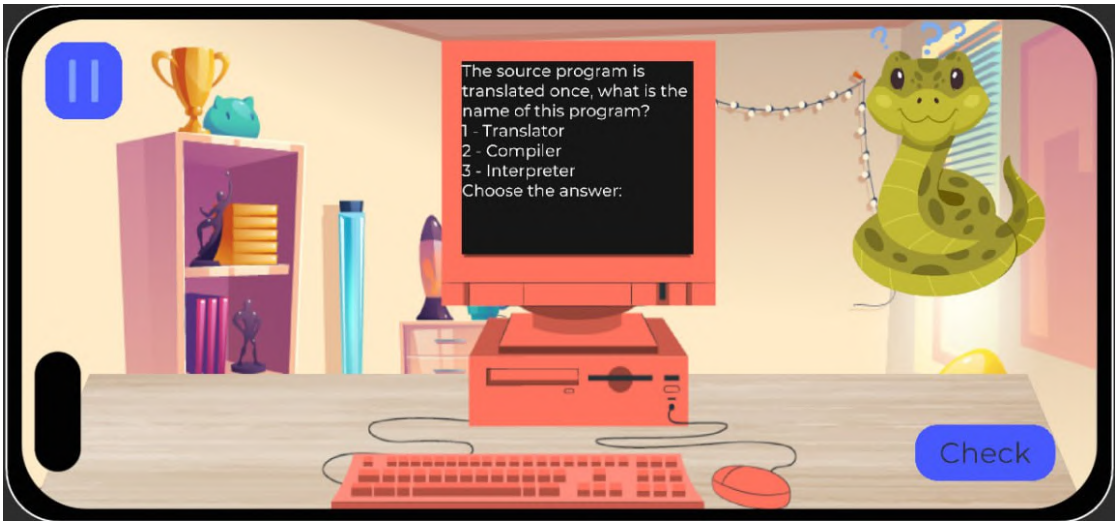

Figure 5: Gameplay of PythonLeaner.

learning other popular programming languages without changing the code. To check the code entered by the player, binding to each level of answers in the game engine was implemented, but this option is not very good, therefore, in the future of the game, we plan to implement an interpreter-based or artificial intelligence system in the game to automatically check the player's code. This innovative technology will allow us to effectively detect errors, optimize the work of players and ensure a high level of security in the game. The compiler or artificial intelligence will be responsible for analyzing the syntax, structure, and efficiency of the code, thus ensuring a higher quality of game software and reducing the possibility of vulnerabilities or anomalies.

Listing 2: Implementation of delay of input keys

```
public void OnButtonClick()      {
    Task.Delay(Convert.ToInt32
            (SingletonScript.pause * 1000)).Wait();
    if (text == result) {
        StopCoroutine(coroutine);
        StartCoroutine(LikeAnimation());
        SingletonScript.coins += 50;
    }
    else {
        StopCoroutine(coroutine);
        StartCoroutine(DislikeAnimation());
    }
}
```

The player learns the following from the assistant:

- the game consists of several levels, at each of them the player will be expected to have

theory and several practical tasks;

- after completing each level, the player receives a specific number of points. These points can be exchanged in the future for the graphic design of the game environment. The elements can be purchased at any time in the store (figure 6).

Figure 6: PythonLeaner Game Store.

After completing all the levels, the player gets an infinite number of points and can buy himself the best computer worth shown in figure 6. It costs 850 points and looks like a modern computer with a curved screen, backlit keyboard and cooling system. The player can choose the best background for the workplace after completing all levels (figure 7). This background

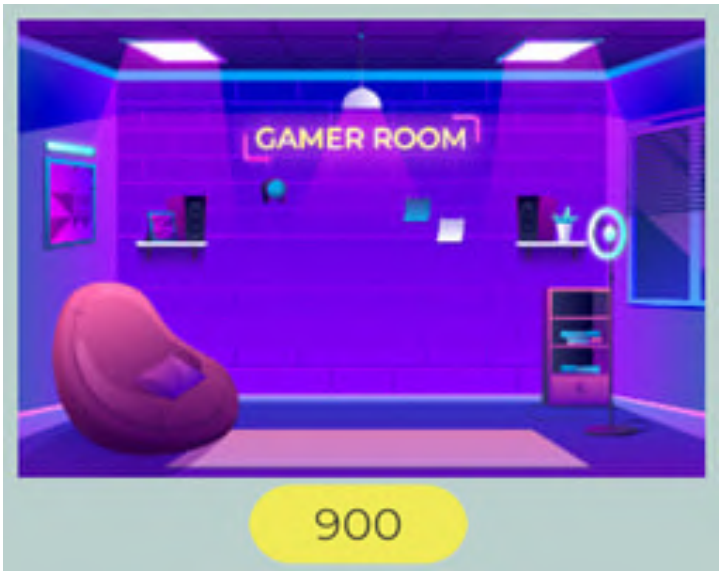

Figure 7: The best PythonLeaner desktop background.

is the most expensive of the ones offered and immerses the player in the real atmosphere of a programmer. There is no delay for entering characters from the keyboard during the passage of levels.

5. Conclusions

The results of this study show that its objectives were met. We analysed some experiences of creating and using game-based learning. On this basis, the requirements for a game application were substantiated and its model was developed. As a result of the comparative analysis, Unity3D was chosen as the main engine for the development of the PythonLeaner game. The chosen set of tools should take into account the needs of graphic development, interaction and ease of use. The author's experience in creating the game was systematised and the key points of its development were described. As a result of understanding the peculiarities of teaching programming, game elements with test and open-ended tasks were developed. It is important to emphasise the adaptability of the application. This implies the ability to adapt it to add tasks or learn other programming languages without editing the game code. Prospects for further research include the development of multiplayer, the integration of an interpreter to check the player's code, and the introduction of artificial intelligence. It is also planned to complete the game by creating a mobile application.

References

[1] 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.

[2] E. Polat, Gamification implementation for educational purposes: a scoping review (2013-2018), Educational Technology Quarterly 2023 (2023) 367–400. doi:10.55056/etq.589.

[3] F. S. Breien, B. Wasson, Narrative categorization in digital game-based learning: Engagement, motivation & learning, British Journal of Educational Technology 52 (2021) 91–111. doi:10.1111/bjet.13004.

[4] A. I. A. Jabbar, P. Felicia, Gameplay Engagement and Learning in Game-Based Learning: A Systematic Review, Review of Educational Research 85 (2015) 740–779. doi:10.3102/0034654315577210.

[5] D. B. Clark, E. E. Tanner-Smith, S. S. Killingsworth, Digital Games, Design, and Learning: A Systematic Review and Meta-Analysis, Review of Educational Research 86 (2016) 79–122. doi:10.3102/0034654315582065.

[6] M. Priyaadharshini, N. Natha-Mayil, R. Dakshina, S. Sandhya, R. Bettina Shirley, Learning Analytics: Game-based Learning for Programming Course in Higher Education, Procedia Computer Science 172 (2020) 468–472. doi:10.1016/j.procs.2020.05.143.

[7] M.-T. Cheng, W.-Y. Huang, M.-E. Hsu, Does emotion matter? An investigation into the relationship between emotions and science learning outcomes in a game-based learning

environment, British Journal of Educational Technology 51 (2020) 2233–2251. doi:10.1111/bjet.12896.

[8] M. Barr, Student attitudes to games-based skills development: Learning from video games in higher education, Computers in Human Behavior 80 (2018) 283–294. doi:10.1016/j.chb.2017.11.030.

[9] J. Díaz, J. A. López, S. Sepúlveda, G. M. Ramírez Villegas, D. Ahumada, F. Moreira, Evaluating Aspects of Usability in Video Game-Based Programming Learning Platforms, Procedia Computer Science 181 (2021) 247–254. doi:https://doi.org/10.1016/j.procs.2021.01.141.

[10] M. Priyaadharshini, M. Natha, R. Dakshina, S. Sandhya, B. Shirley, Learning Analytics: Game-based Learning for Programming Course in Higher Education, Procedia Computer Science 172 (2020) 468–472. doi:10.1016/j.procs.2020.05.143.

[11] C. Kroustalli, S. Xinogalos, Studying the effects of teaching programming to lower secondary school students with a serious game: a case study with Python and CodeCombat, Education and Information Technologies 26 (2021) 6069–6095. doi:10.1007/s10639-021-10596-y.

[12] S. Papadakis, A. E. Kiv, H. M. Kravtsov, V. V. Osadchyi, M. V. Marienko, O. P. Pinchuk, M. P. Shyshkina, O. M. Sokolyuk, I. S. Mintii, T. A. Vakaliuk, L. E. Azarova, L. S. Kolgatina, S. M. Amelina, N. P. Volkova, V. Y. Velychko, A. M. Striuk, S. O. Semerikov, Unlocking the power of synergy: the joint force of cloud technologies and augmented reality in education, CEUR Workshop Proceedings 3364 (2023) 1–23. URL: https://ceur-ws.org/Vol-3364/paper00.pdf.

[13] K. P. Osadcha, V. V. Osadchyi, The use of cloud computing technology in professional training of future programmers, CTE Workshop Proceedings 8 (2021) 155–164. doi:10.55056/cte.229.

[14] D. V. Yatsenyak, V. P. Oleksiuk, N. R. Balyk, Study of ergonomic criteria for evaluating the software user interface, Journal of Physics: Conference Series 2288 (2022) 012005. doi:10.1088/1742-6596/2288/1/012005.

[15] S. Schez-Sobrino, D. Vallejo, C. Glez-Morcillo, M. A. Redondo, J. J. Castro-Sanchez, RoboTIC: A serious game based on augmented reality for learning programming, Multimedia Tools and Applications 79 (2020) 34079–34099. doi:10.1007/s11042-020-09202-z.

[16] N. Balyk, Y. Vasylenko, G. Shmyger, V. Oleksiuk, A. Skaskiv, Design of Approaches to the Development of Teacher's Digital Competencies in the Process of Their Lifelong Learning, CEUR Workshop Proceedings 2393 (2019) 204–219. URL: https://ceur-ws.org/Vol-2393/paper_237.pdf.

[17] M. Hidalgo, H. Astudillo, L. M. Castro, Challenges to Use Role Playing in Software Engineering Education: A Rapid Review, in: H. Florez, M. Leon (Eds.), Applied Informatics, Springer Nature Switzerland, Cham, 2024, pp. 245–260. doi:10.1007/978-3-031-46813-1_17.

[18] M. Frania, Let's Open the Locker of Creativity - How the Traditional Educational Escape Room Changed into a Virtual Puzzle Game During the COVID-19 Pandemic, in: Ł. Tomczyk (Ed.), New Media Pedagogy: Research Trends, Methodological Challenges and Successful Implementations, Springer Nature Switzerland, Cham, 2023, pp. 102–114. doi:10.1007/978-3-031-44581-1_8.

[19] M. I. M. Nadziman, H. F. M. Hanum, N. A. K. Adnan, N. M. Diah, Z. A. Bakar, Game-Based Mobile Application for Tarannum Learning, in: H. Badioze Zaman, P. Robinson, A. F.

Smeaton, R. L. De Oliveira, B. N. Jørgensen, T. K. Shih, R. Abdul Kadir, U. H. Mohamad, M. N. Ahmad (Eds.), Advances in Visual Informatics, Springer Nature Singapore, Singapore, 2024, pp. 223–233. doi:10.1007/978-981-99-7339-2_20.

[20] V. H. Andaluz, J. S. Sánchez, C. R. Sánchez, W. X. Quevedo, J. Varela, J. L. Morales, G. Cuzco, Multi-user Industrial Training and Education Environment, in: L. T. De Paolis, P. Bourdot (Eds.), Augmented Reality, Virtual Reality, and Computer Graphics, Springer International Publishing, Cham, 2018, pp. 533–546. doi:10.1007/978-3-319-95282-6_38.

[21] F. Horn, S. Vogt, S. P. Göbel, GameTULearn: An Interactive Educational Game Authoring Tool for 3D Environments, in: M. Haahr, A. Rojas-Salazar, S. Göbel (Eds.), Serious Games, Springer Nature Switzerland, Cham, 2023, pp. 384–390. doi:10.1007/978-3-031-44751-8_32.

[22] C. M. Torres-Ferreyros, M. A. Festini-Wendorff, P. N. Shiguihara-Juárez, Developing a videogame using unreal engine based on a four stages methodology, in: 2016 IEEE ANDESCON, 2016, pp. 1–4. doi:10.1109/ANDESCON.2016.7836249.

[23] T. A. Vakaliuk, O. M. Spirin, N. M. Lobanchykova, L. A. Martseva, I. V. Novitska, V. V. Kontsedailo, Features of distance learning of cloud technologies for the organization educational process in quarantine, Journal of Physics: Conference Series 1840 (2021). doi:10.1088/1742-6596/1840/1/012051.

[24] H. Żukowski, Comparison of 3D games' efficiency with use of CRYENGINE and Unity game engines, J. Comput. Sci. Inst. 13 (2019) 345–348. URL: https://ph.pollub.pl/index.php/jcsi/article/view/1330. doi:10.35784/jcsi.1330.

[25] J. Pirker, M. Bertini, M. Lux, Open source for video games: a shortlist of game engines, SIGMultimedia Rec. 12 (2022). doi:10.1145/3548570.3548579.

[26] B. Xu, T. Jing, H. Lin, J. Du, G. Zhu, CryEngine based virtual geographic environments construction, Wuhan Daxue Xuebao (Xinxi Kexue Ban)/Geomatics and Information Science of Wuhan University 42 (2017) 28 – 34. doi:10.13203/j.whugis20140768.