Project Éden: platform for introductory programming concepts

Yure Pablo do Nascimento Oliveira
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, Brasil
yurepablo@usp.br

Claudio Fabiano Motta Toledo
Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, Brasil
claudio@icmc.usp.br

Leonardo Tórtoro Pereira
Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, Brasil
leonardo.t.pereira13@gmail.com

***Abstract*—The area of programming is perceived as complex and requires challenging skills, and programming courses have some of the highest dropout rates. Educational games are good solutions to these problems as they promote student engagement and an immersive learning experience. In this context, the present paper discuss the development and the validation of a more complete and more playful version of the Project Éden game, which aims to improve the learning process of coding. The player needs to instantiate variables and modify their values, and also must interpret, choose, and adjust in code form to advance through the stages and reach the final challenge. The game was developed with a methodology based on the evolutionary prototype concept. A set of users evaluated the game, and the results reported a satisfactory user experience. The game was considered fun with valuable tips for solving problems with content approached playfully and understandably by undergraduate, graduate students, and professors.**

***Index Terms*—serious game, learning process, programming tool, videogame**

I. Introduction

The authors in [1] identify two issues in the area of programming. First, the area is perceived as complex and requires challenging skills, in which it takes about a decade to become an expert. Second, programming courses have some of the highest dropout rates, which indicates the inadequacy of traditional teaching methods.

Educational games can be part of the solution to these problems as they promote student engagement and an immersive learning experience. If the narrative, gameplay, and game elements are appealing, they will release endorphins into the player's body, leading to the fun. In this way, learning will be a second thought, which makes the individual retain more information, learn more efficiently and store the knowledge in your long-term memory [2].

In a questionnaire applied to 55 Higher Education students in [1], they evaluated ten skills generating a ranking of the most important to be a successful programmer. In order of importance, the first three are problem-solving skills, analytical reasoning, and critical thinking. In contrast, the three most minor essential skills for them are math, leadership, and memorization.

Project Eden hopes to help the learning process of coding through a platform game with an educational purpose. The player needs to instantiate variables and modify their values to advance through the stages and reach the final challenge. The first version of this platform is reported in [12]–[14].

The present paper presents a new version of Project Eden[1] to make it even more complete and playful. The game was developed in *engine* Unity, with the addition of new content related to the conditional structure and the iterative structure and new mechanics to support learning. In the game, the player must interpret, choose and modify in code form to meet and reach the final challenge.

Section II presents a short literature review, and the game development methodology is in Section III. Section IV discusses the Project Eden based on interface, mechanics, and game levels. In section V, the game is validated based on Bloom's taxonomy [15], on the technical and pedagogical criteria proposed by [10] and on the user experience. Finally, section VI presents our conclusions.

II. Literature Review

A systematic mapping reported by [3] in 2007 found only 43 serious games in scientific publications for learning programming, which they consider a small number. A total of only six studies obtained more than 70% of points about the quality criteria established by the authors, based on methodology, description of educational objectives, tests with students, and comparison with other works.

[1] https://yurepablo.itch.io/projetoeden

The research in [4] reported feedbacks from 19 participants enrolled in programming courses. Most of them listed three elements as more effective for learning: reward for completing a challenge, the chance to repeat a phase unsuccessful without losing current progress, and level indication to generate progress. The least effective elements, according to the study, were to limit the player's action within a period and the *multiplayer* approach. In addition, the article extracts the keywords most commented on by subscribers in essay responses about serious games. The first four words are self-learning, attracting attention, guiding, and staying focused.

According to [7], education is a sensitive area, and if serious game design is not well defined, it will not be able to solve all its challenges. In this context, they classify serious gaming as exogenous or endogenous. The exogenous approach reuses other games and adds educational content, e.g., "Angry Birds" to understand Physics events. Ther endogenous depends on the educational content addressed and is intrinsically linked to it. For the authors, the endogenous approach is considered better for producing serious games.

Regarding works related to videogames of learning programming, we have the mobile games 2D Maze and 3D Aventure in [5]. In 2D Maze, a spider mother teaches her child to walk on the web, and in 3D Adventure, a girl must make it to the other side of the island. Both use sequential, conditional, and iterative commands with the drag and drop technique. A total of 20 Higher Education students rated these games as fun and learning facilitators, with an overall average of 4.27 on a 5-point scale.

The authors in [6] compare 20 commercial games available on Steam and Google Play. They have approached conditional and repetition structures, most feature functions, parameter passing, and debug strategies. A small number features variables and problem-solving algorithms. Also, the majority of these games are puzzle types. However, these videogames lack phases involving planning and strategy selection before writing code, which experienced programmers usually must do for better code organization and reduce chances of error [6].

It is worth mentioning the previous version of the Project Eden [12], a platform game in which a computer scientist must save the world from an Artificial Intelligence (AI) that got out of control by fulfilling coded missions. The videogame explores the entire composition of a variable: name, type, and content and brings the concept of constant, in a context where only two games about variables were found, focused on data types and memory allocation.

The Project Eden update enabled better graphics, the creation of a responsive design, and the application of classic game mechanics with significant improvements from an interactive coding screen. We also added instant feedbacks for more programming contents such as conditional and iterative structures. In this way, it is possible to cover all the elementary topics necessary to understand an algorithm. Furthermore, the project was improved by including more current programming languages: C, C#, Java, and Python.

III. Development Methodology

The present section describes how we update the previous version of Project Eden. The development cycle shown in Fig. 1 foresees five stages: Conception, Relation, Analysis, Production, and Testing.

Fig. 1. Game's development cycle [12]

In Conception, we defined more game objectives based on the Elementary Tetrad of [9], which divides the game into four elements: aesthetics, narrative, mechanics, and technology. We added new content, decided about technologies to be employed, explored new subjects, and modified old ones. At this stage, the development changed from Construct 2 engine to Unity for the reasons listed below.

- Unity is a more flexible and adaptable engine, allowing work to improve design responsiveness, resulting in an equivalent experience across multiple devices and easing the transition to mobile devices — the game design in Construct 2 was static;
- Unity is the second most used game engine in Steam games and the first in Itch.io [11], which indicates a greater range of documentation and information on forums;
- Construct 2 has some limitations in its free version, while Unity gives Premium equivalent access to students;
- Scripts in Unity are programmable using C#, an object-oriented language, which facilitates code modularization and expands the range of possibilities in the tool.

In the Relation stage, a literature review searches for serious games in programming that could serve as a basis for the Project Eden. The Analysis sought to identify gaps in related work and good practices already implemented to improve the game.

During Production, the game was developed, following Object Oriented Programming (OOP) practices, the ten quality principles for game development, determined by [9], and the technical and pedagogical criteria proposed by [10] for building serious games.

According to [8], it is desirable, when designing software to standardize the code, reuse components, and a short development time. Changing the engine caused development delay because the two engines are not integrable, and it was not possible to reuse functions. Despite this, reuse the entire visual identity of the game, and the OOP principles allow to scale the game quickly and efficiently.

Finally, the Testing stage evaluated each mission added to this new version. At this stage, we realize the need to make the tips more dynamic and add more support phases in the sub-themes to fix the content better. Before the release, an evaluation was carried out with students and teachers to collect

feedbacks. Based on the feedback, we reduced the difficulty level for some phases and added instructions to clarify the mission goals better. The Analysis, Production, and Testing steps took place iteratively throughout the cycle.

IV. Project Eden: Description

Project Eden is a 2D platform game that builds on successful elements in commercial games, such as Super Mario to create an immersive and fun learning experience of primary content in the schedule. In this game, educational content is treated as an intrinsic part of the plot and mechanics, allowing anyone to play it for the simple desire of entertainment voluntarily.

The game was tested and adapted for four programming languages: C, C#, Java, and Python. Two other languages, Pascal and Visualg, were removed from the project because they are less used in Higher Education and have a syntax that is more out of place than most existing programming languages.

The game encourages the player to declare sentences in code form to modify the game state and advance through the stages. These sentences include declaring variables and assigning values to them, condition structures, and repeating structures. In addition, it uses playful and intuitive situations, such as defining nickname, solving puzzles, opening blocked passages, and facing enemies.

It is expected that when playing Project Eden, the player will develop the following skills: i) name variables according to agreed programming standards; ii) create sentences involving variables; iii) assign values to variables; iv) identify the most suitable type for a variable; v) create and interpret logical sentences *if* and *if-else*; vi) identify repetitions and choose the appropriate representation structure (*for* or *while*); vii) solve logical reasoning problems;

A. Plot

A computer scientist is sent to an island paradise to destroy an Artificial Intelligence (AI), which controls all life forms through commands in a period of fierce war. However, the AI got out of control: it didn't distinguish humans from objects, discarding them to generate improved things to keep the peace. So the scientist must carry out missions, reprogramming the AI in his favor, to achieve his goal and restore humanity in a secret project entitled "Project Eden".

B. User Interface

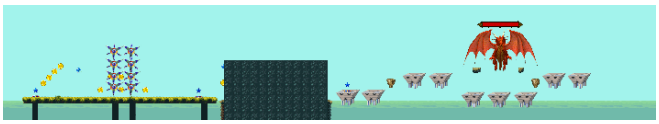
The main scenario of the game is a paradise island full of strange enemies and with a variety of platforms presented in the Fig. 2. The last phase of the game takes place in the Command Center with a more technological aspect in Fig. 3.

We introduce new enemies during the current development: flying chicken, rabid bee, carnivorous flower, colored slugs, soldiers with ax and spear, warrior, crawling guinea pig, personified cloud, monster, and dragon. Fig. 4 shows them.

In this new version, we introduce a general guide (Fig. 5), representing the commander in the war that sent the scientist to accomplish the mission. The general provides hints, introduces new skills, and guides the player in the game tutorial.

(a) Phases 1 to 6

(b) Phases 7 to 10



(c) Phases 11 to 13

Fig. 2. Main scenarios: paradise island

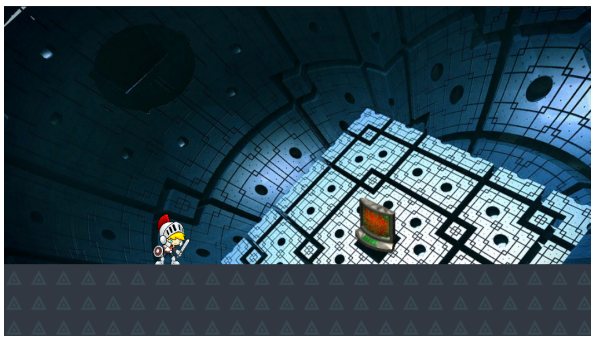


Fig. 3. Last phase: command center

Fig. 4. Enemies

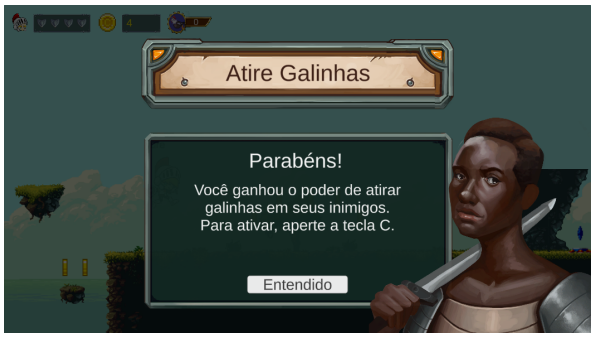


Fig. 5. General introducing a new skill

In the game, when seeing a tip shown by the general tab, the letters appear with a delay of time, giving the impression of a dialogue between the player and the actual character. Furthermore, at the top of the screen, there are health and coin quantity indicators. In addition, four other indicators appear as soon as the player gains a new power. These indicators are essential from a didactic point of view, as they characterize the change or not of the variables instantiated during the game.

The first indicator shows the number of chickens, and the

second shows the values *true*, when the flame is activated by the player, or *false* when deactivated. The third indicator shows the code that does 50% more damage to the sword. It is a value that doesn't change during gameplay, spelling out the meaning of a constant. Lastly, the player's current height indicator varies as it increases or decreases with the given commands. Fig. 6 illustrates all of them.

Fig. 6. Game indicators

There is a pause option, as shown in Fig. 7, which allows accessing the description of commands related to the three programming topics: variables, conditional structures, and iterative structures. This step describes basic commands in the programming language such as variable types, acceptable naming forms, if and for details, among others. The menu also allows you to access game controls, i.e., activated by some keys. Finally, it is possible to go back from the last checkpoint at any time, restarting the game or following by clicking the corresponding buttons, as illustrated in Fig. 7.

Fig. 7. Pause Menu

C. Game Mechanics

The player can use commands common in platform games, such as using the horizontal arrows to move right or left and the upper vertical arrow to jump. The arrow keys can be replaced by the W, E, A, and S keys, as in most games. The player also gains powers throughout the game, such as: throwing chickens, pressing the C key, activating the flame, pressing E, increasing height, pressing Q, and decreasing height, pressing Z. Finally, the X key lets you attack with the sword, both in the air and on the ground.

The main difference of the game is the interaction with a screen of encoding, which has retained its visual look and textual style but has undergone changes to the tooltip system since the first version. Instead of three fixed hints, the player can buy multiple clues by pressing a single button. Tips increase their price by two coins with each use. When the player acts, from choosing an option in dropdown to possible typing values in the text box, hints are generated based on the most common mistakes. In the case of dropdowns, the guidance explains why the selected option is incorrect and leads to the correct path, increasing its didactic potential.

The common mistakes are seen from text boxes, such as forgetting the semicolon (;), not putting double quotes in textual values, putting comma and non-period in decimal values, memory overflow, among others, and provides tips directed to the possible player error. The hints are stored in a stack structure, so the direction for the most recent error is always shown first.

The coding screen improved the player interaction replacing text boxes by dropdowns when choosing the type and name of the variable. The same happens for sentences in conditional and iterative structures. The idea was becoming easier for beginning students with problems to memorize all languages' types and rules for variable names, among other issues. The modifications are in the Fig. 8.

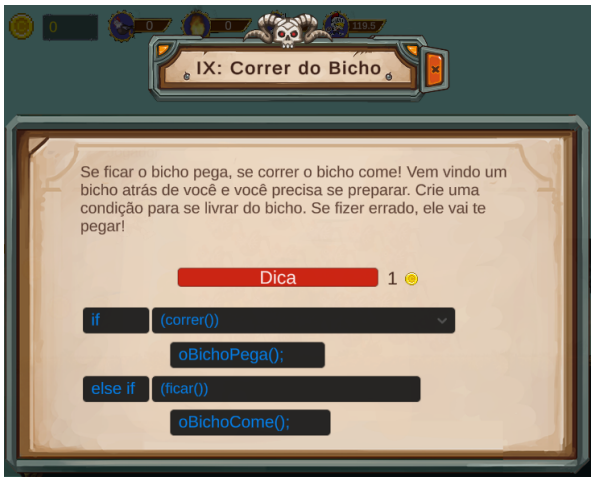


Fig. 8. New screen for coding on phase 9 in C language

In the last phase of the game, the encoding screen takes on a more technological aspect, as can be seen in the Fig. 9. The player has no hints available and interacts with the last mission via a scroll bar and a large text box for entering the password. When the password is discovered, the AI is destroyed once and for all.

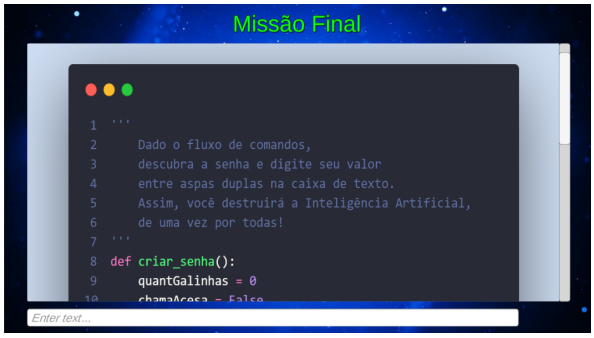


Fig. 9. New screen of coding on final phase in Python language

Finally, you can activate the pause menu by pressing ESC or P. As browsers often use ESC to disable full screen, for a better browsing experience, you can use the P key.

D. Phases

The game consists of fifteen stages. Each of them contains a mission that must be accomplished to advance, most involving coding. In this way, educational content is placed naturally as an indispensable resource for winning the game. The first stage features a step-by-step tutorial to familiarize the player with the mechanics of the coding screen. After the tutorial, the following six phases are about variables and data types, three about conditional structures, and three about repeating structures. The last two missions in the game represent more difficult challenges to consolidate all the knowledge acquired. The stages have progressive difficulty in each sub-theme so as not to bore or discourage the player.

Table I presents the fifteen phases of the game, citing the mission's difficulty, the task, and the pedagogical objectives.

TABLE I
PHASES OF THE GAME

Phase	Difficulty	Mission	Pedagogical Objectives
Understanding the Game	Easy	Create Nickname	Understand the concept of constants and the dynamics of the coding screen
Phase 1	Easy	Reduce Enemies	Declare a variable of integer type and understand its behavior
Phase 2	Easy	Extinguish the Flame	Declare a variable of logical type and differentiate *true* and *false* values
Phase 3	Medium	Destroy Soldiers	Declare a variable of char-type and solve logical reasoning problems
Phase 4	Hard	Pass the barrier	Declare a variable of real type and assign value to a variable using arithmetic operations
Phase 5	Hard	Crush the Clouds	Change the value of a variable and manipulate variables of real type
Phase 6	Medium	Cheat the Cameras	Manipulate variables of string type and logical thinking to solve problems
Phase 7	Easy	Stop Blocks	Declare conditional structures, differentiate their common operators and understand their behavior
Phase 8	Medium	Free Yourself	Declare conditional structures, differentiate their common operators and understand their behavior
Phase 9	Hard	Run Away from the Monster	Declare linked conditional structures, differentiate their common operators and understand their behavior
Phase 10	Easy	Build Path	Declare iterative structure with *while* and understand its behavior
Phase 11	Easy	Eliminate Blades	Declare iterative structure with *for* and understand its behavior
Phase 12	Medium	Break the 4th wall	Declare iterative structure with with nested *fors* and understanding its behavior
Phase 13	Hard	Kill the Dragon (*boss*)	Develop quick and strategic thinking
Phase 14	Very Hard	Destroy the Artificial Intelligence	Read an algorithm and analyze it; Combine several data into a variable of string type; Code without the dependency on hints

V. GAME VALIDATION

A. Assessment based on Bloom's Taxonomy

First, we will analyze the game based on Bloom's Taxonomy in the context of the introductory programming [15], composed of six categories: remember, understand, apply, analyze, evaluate and create.

The action to remember concerns the knowledge present in long-term memory. In project Eden, we added new phases to favor this feature; in this way, using previous commands in later missions, it is possible to exercise memorization. Variable identifiers also act as mental triggers for fixing related content. It is noteworthy that there are always instructions available to describe the main commands in the pause menu.

The understanding category already requires the player's skills of interpretation, classification, inference, among others. The mechanics of selecting the missing parts and filling the text boxes, based on the mission description, validate the skill mentioned earlier. The player must understand what the mission asks for, sort between the options to execute the mission more efficiently, and often infers the result with the knowledge acquired previously.

In the apply category, project Eden presents the resources available to the player. However, it is up to him to manipulate the commands to produce results in the game world. Once the problem is understood, it must be resolved soon afterward. Many times, several commands are correct but do not produce the expected result, which exercises through trial and error the action of actually completing the mission.

In the analyze category, the game guides challenges by a relationship between one command and the other and their differences, when each one should or can be used. When relating data types to find which one solves the problem or, for example, differentiating between two repetition structures, *for* and *while*, the player will need to exercise analysis skills.

As for evaluating, the game does not have error checking or efficiency phases in pre-designed codes. However, there is

more than one correct command in many steps, so no matter how much any one of them works if executed correctly, the player can perceive conditions that are more difficult to be reached during execution. It is also possible to encounter logic errors that result in infinite *loops* or instructions that never happen.

In the last category, since the game focuses on pre-processed responses and is limited to finite and premeditated options, the player has no compilation of freely typed code. This aspect compromises the proposition of alternative algorithms to solve the same problem or construct code segments based on examples. Project Eden is expected to help the player create codes in other situations in their real-life course with skills acquired in the game.

B. Assessment according to technical and pedagogical criteria

We will also evaluate according to the technical and pedagogical criteria proposed by [10]. Next, a pedagogical assessment is carried out considering the aspects described below by the authors.

Behavioral approach: the game presents brief information, that is, without a heavy load of text at a time; there are tests with visual *feedbacks* on each mission and results right after in the game world. The results act as a reward for the correctness, allowing the player to continue their journey. Given the nature of coding, as an optionally collectible power, there are stages where it is possible to proceed without coding. However, they become much more difficult. In case of errors, the game forces the player to return to the previous point using *game over*.

Constructivist approach: the game proposes problem situations that involve the formulation of hypotheses, investigation, and comparison; it is often possible to go through different paths, even if limited, to solve the problem. Thus, Eden project encourages the search for information either in the menu or in other sources.

Socio-Interactionist Approach: since it is a *single-player* game, there are no communication tools between participants. In the classroom, the teacher can promote the discussion of content after the game.

Ability to Adapt: the game did not pass rigorous criteria to include all learning styles. It is inferred that it is a good tool for students who like visual stimuli, who are comfortable with reading and writing that are not too long, and who like practical activities and/or those that require logic and intuition. The Éden project is aimed at beginning students, so the difficulty is adapted for this audience. Next, a technical evaluation considers the aspects below described by the authors.

Robustness: it is possible to finish the game without apparent errors; in case something unexpected happens during the game, you can return to the last *checkpoint* using the pause menu.

Portability: it is possible to play from any operating system, and computers with different configurations since the game is available *online*. In the current version, there is no adaptation for mobile devices.

Image use: the images in each mission illustrate the user's *feedback*, so there is no imagery overload; the indicators that are always present in the game were developed compactly, not getting in the way of the experience.

Information Presentation: there was concern about the contrast between font and background, being possible to read without much difficulty; there is no user control of the font size. However, the displayed size is believed to be adequate for most people. The texts are aligned to the left, making them easier to read on different devices, and each mission has a title, which facilitates the immediate understanding of the phase.

Orientation and Navigation: the objects with which the player interacts are official *engine*, therefore, developed in a very intuitive way; it's possible to know where it's at since the mission titles include its number; the *checkpoints* in each mission help to separate one from the other.

Interactivity: you can play with the game world with different keys, buttons, *dropdowns* and text boxes, as well as get other answers according to your actions.

Aesthetics: the game employs several graphical resources, such as transitions and animations, making the interface more pleasant than traditional compilers;

Affection: there is the expression of various affective states through static and animated characters and sounds and soundtracks to create immersion in the game world.

C. Assessment with players

In this research, 22 people participated: 15 undergraduate students, four graduate students, and three undergraduate professors. The applied form contained six questions and aimed to measure the technical and pedagogical characteristics of the game. In table II you will find an overview of the questionnaire.

TABLE II
QUESTIONNAIRE TO MEASURE TECHNICAL AND PEDAGOGICAL CHARACTERISTICS OF THE GAME USING THE 5-POINT LIKERT SCALE

Question	Type
Who are you?	Multiple choice: Undergraduate Student; Undergraduate Teacher; Graduate Student
What is your experience with platform games?	1 = none 5 = high
The game is fun and has immersive features	Likert 1-5
The tips offered are useful for solving the problems	Likert 1-5
The level of challenges is adequate and progresses evenly	Likert 1-5
The listed content, with its specifics, is approached in the game in a playful and understandable way: 1. Variables and Constants; 2. Conditional Structure; 3. Repetition Structure.	Likert 1-5

The public responding to the survey has experience with platform games in general. For example, in the Fig. 10, it can be seen that 86.4% of respondents have medium to high background and that only 13.6% of them have little experience.

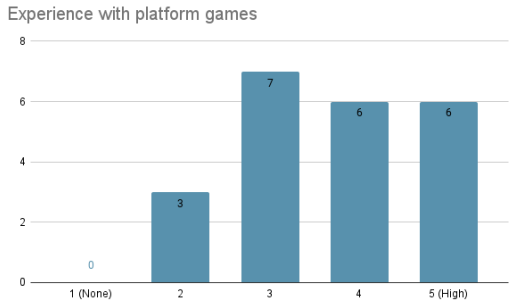


Fig. 10. Participants' level of experience in platform games

The first technical question was about the immersive and fun features of the game. It is concluded, analyzing the Fig. 11, that the game provided a fun and immersive experience for more than 60% of people. Only one participant disagreed with the statement.

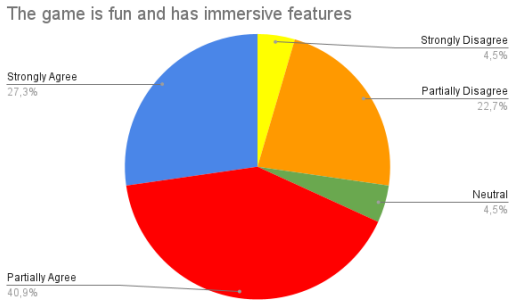


Fig. 11. Graph showing the participant's degree of agreement as to whether the game is fun and has immersive characteristics

The next question aimed to measure the tips for solving missions that involved coding to ensure that the student's independence in pursuit of their knowledge is being safeguarded. In the Fig. 12, we observe more than 50% of the interviewees agree that the recommendations are helpful in the game, attesting as valid the effort to make the information more dynamic and personalized. From the teachers' point of view, two agree with the statement, and one is indifferent.

The penultimate question aimed to measure the degree of balance between the phases. As the graph in the Fig. 13 shows, more than 60% of respondents fully or partially agree that the level of challenges is adequate and progresses evenly. The only 13.6% of the participants partially or totally disagreed with the statement. From the teachers' point of view, two of them partially agree and one is indifferent.

The last question aimed to compare the three sub-themes covered in the game and verify if we approached these contents playfully and understandably. As illustrated in Fig. 14, the subtopic closed to those aspects was conditional structure. It obtained a higher degree of total agreement, followed by variables and repetition structure.

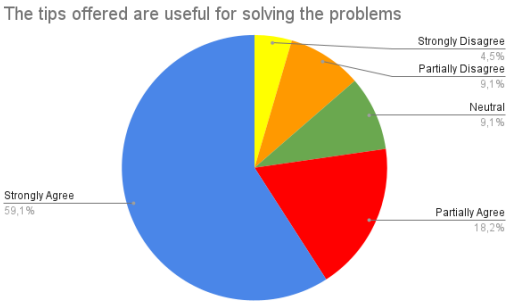


Fig. 12. Graph showing the participant's degree of agreement as to whether the tips are useful for solving the proposed problems

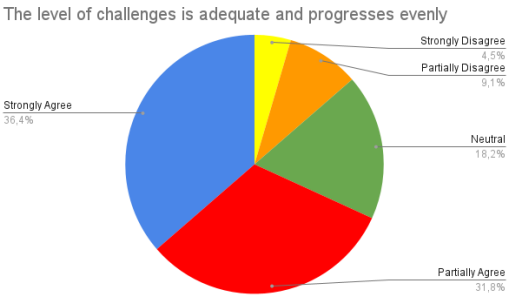


Fig. 13. Graph showing the participant's degree of agreement as to whether the level of challenges is adequate and progresses in a balanced way

There is also a very low degree of disagreement, only 4.5% of the participants partially disagreed on each content, and there was no total disagreement. From the teachers' point of view, one of them partially agrees with the statement in both contents. Another fully agrees concerning the variables and partially to the conditional and repetition structures, and the latter is indifferent.

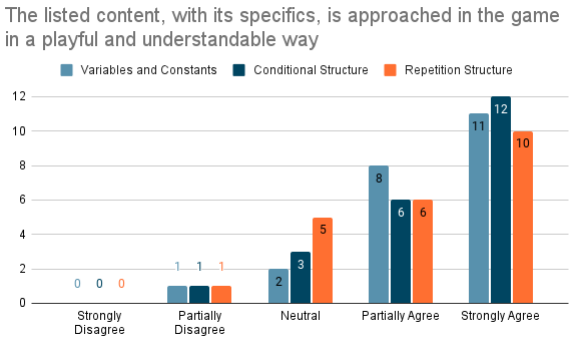


Fig. 14. Graph showing the participants' degree of agreement as to whether each content (variables, conditional structure, and repetition structure) is approached in the game in a playful and understandable way

VI. Conclusion

This paper presented the new version of the Project Eden game, which became more complete and playful. A set of players evaluated the game, and the results reported a satisfactory user experience. The game was considered fun with valuable tips for solving problems with content approached playfully and understandably by undergraduate, graduate students, and professors.

The new version of the game is complete, covering all the initial topics necessary to understand an algorithm: variables and data types, conditional structures, and repetition structures. Furthermore, there is a considerable expansion of the game world, with new enemies and battles. It is worth mentioning the addition of a general guide that helps the player during his journey, a pause menu with the description of the main commands of the selected programming language, and the dynamic hint system that offers immediate feedback according to the most recent mistake made by the player.

With the change of engine to Unity, the game also significantly improved technical interface, speed, and responsiveness, ensuring a better experience for the player. In a context in which educational games are still not widespread in the industry and little used in the classroom, the Project Eden is seen as a possibility to bring quality playful content that complements the classes and helps students consolidate essential concepts for career follow-up in Computing.

As future work, the game will be validated by beginners students with pre and post-tests. Also, attentive to the growth in the use of cell phones, the idea is to create an adaptation of the game for Android and iOS platforms. In addition, you can use the game's successful mechanics in other more specific contents of the area, such as Object-Oriented Programming, ordering algorithms, algorithm complexity, data structures, among others.

Acknowledgment

We thank the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) for funding this project. The grant follows Process 2019/24451-8.

References

[1] T. Hainey, G. Baxter, and A. Stanton, "A Serious Game to Teach Rudimentary Programming: investigating content integration", presented at the 12th European Conference On Game Based Learning, France, Oct 4-5, 2019.

[2] K. Borna and H. M. Rad, "Serious Games in Computer Science Learning Goals", presented at the 2nd National and 1st International Digital Games Research Conference, 2018, doi: 10.1109/dgrc.2018.8712030.

[3] A. L. Santos, M. R. A. Souza, E. Figueiredo, and M. Dayrell, "Game Elements for Learning Programming: a mapping study", presented at the 10th International Conference On Computer Supported Education, 2018, pp. 89-101, doi: 10.5220/0006682200890101.

[4] A. L. Santos, M. R. A. Souza, E. Figueiredo, and M. Dayrell, "Exploring Game Elements in Learning Programming: an empirical evaluation", presented at the IEEE Frontiers In Education Conference (FIE), 2018, doi: 10.1109/fie.2018.8658505.

[5] A. Kurniawati, N. H. Akbar, and D. Prasetyo, "Visual Learning on Mobile Phone for Introduction Basic Programming in Vocational High School", presented at the 2018 International Conference On Computer Engineering, Network And Intelligent Multimedia (cenim), 2018, doi: 10.1109/cenim.2018.8710873.

[6] A. Areizaga, "PROGRAMMING LEARNING GAMES: identification of game design patterns in programming learning games", M.S. thesis, Escola de Informática, Universidade de Skövde, 2019.

[7] W. Mestadi, K. Nafil, R. Touahni, and R. Messoussi, "An Assessment of Serious Games Technology: toward an architecture for serious games design", presented at the 2018 International Journal Of Computer Games Technology, 2018, pp. 1-16, doi: 10.1155/2018/9834565.

[8] A.N. Domingues, M. L. Lotufo, A.F.S. Silva, A.C.P Guimarães, J.G.S.F. Esteves, J.L. Otsuka, D.M. Beder, and S.H. Zem-mascarenhas, "Uso de protótipo em papel no design de um jogo educacional acessível", presented at the XIII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), 2014.

[9] J. Schell, *The Art of Game Design: A book of lenses*, Amsterdam, Boston: Elsevier, 2008.

[10] E. Reategui, E. Boff, and M.D. Finco, "Proposta de Diretrizes para Avaliação de Objetos de Aprendizagem Considerando Aspectos Pedagógicos e Técnicos" in *CINTED-UFRGS Novas Tecnologias na Educação*, *vol. 8*, 2010.

[11] M. Toftedahl, "Which are the most commonly used Game Engines?". Gamasutra.com. https://www.gamasutra.com/blogs/MarcusToftedahl/20190930/350830/Which_are_the_most_commonly_used_Game_Engines.php (accessed Jun. 9, 2020).

[12] Y. Oliveira and C. Farias, "projetoÉden: Jogo Sério sobre Variáveis e Tipos de Dados", presented at the XVII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), 2018.

[13] Y. Oliveira and C. Farias, "Desenvolvimento e avaliação do jogo Sério projeto Éden sobre variáveis e tipos de dados", in Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe, Ilhéus, 2019, pp. 615-624.

[14] Y. Oliveira and C. Farias, "Desenvolvimento e Avaliação do projetoÉden, um jogo educacional sobre Variáveis e Tipos de Dados" in Revista de Sistemas e Computação, Salvador, *vol. 10*, 2020, pp. 140-151.

[15] E. A. Jesus and A. L. A Raabe, "Interpretações da Taxonomia de Bloom no Contexto da Programação Introdutória" in Anais do XX Simpósio Brasileiro de Informática na Educação (SBIE), 2009, doi: 10.5753/cbie.sbie.2009.%25p.