Games for coding to attract new students to STEM

Antonio Calvo-Morata
Universidad Complutense de Madrid
Madrid, Spain.
acmorata@ucm.es

Niklas Humble
Mid Sweden University, Sweden
Sweden
niklas.humble@miun.se

Peter Mozelius
Mid Sweden University, Sweden
Sweden
peter.mozelius@miun.se

Rasmus Pechuel
Ingenious Knowledge
Cologne, Germany
rasmus.pechuel@ingeniousknowledge.com

Baltasar Fernández-Manjón
Universidad Complutense de Madrid
Madrid, Spain.
balta@ucm.es

***Abstract*— There is a need to increase the number of students, especially women, choosing programming and STEM disciplines. We need innovative approaches in schools to better engage students and awake their interest in computer science. This paper addresses the need to create tools that effectively support the learning of programming and the development of computational thinking, highlighting why video games can be an effective educational tool for it and also attract new students to STEM. The Game4Coding Erasmus+ project proposes the design of a video game called CodeQuest, using a game genre that has not been frequently used before to address the teaching of programming, the monster tamer genre. We consider that video games have a number of benefits such as that stimulate active learning, are engaging for a wide range of students, and present information in a way that is attractive to learners. We want to explore this kind of game's effectiveness as a learning tool as well as its effect on the perception of STEM disciplines and programming to attract new public to coding (especially girls).**

Keywords*— *Game-based learning, Programming learning games, Computational thinking.

I. Introduction

The basic understanding of computer science at early stages is a key element for the digital transformation of our society and economy. With the generalization of technology in all facets of daily life, a trend that has been accelerated recently (e.g. teleworking or remote education due to COVID), and the emergence of new and increasingly sophisticated computer applications (e.g. those based on artificial intelligence), a minimum knowledge of computer science and programming is becoming more and more necessary. For this reason, computational thinking and programming have become highly relevant in the compulsory education program [1]. However, achieving this digital literacy is complex as it requires not only trained teachers and a coherent curriculum approach but also to engage students and awaken their interest in computer science. Many students, especially girls, have no interest in the subject and still consider computer science to be a difficult and arduous subject that they dismiss out of hand. Therefore, we need new approaches to teaching and learning programming that are more motivating and attractive to a wider audience.

Until now, programming and computational thinking has usually been a supplementary content in schools and its assessment has not usually been very systematic. For example, in the OECD's Programme for International Student Assessment (PISA 2022), which evaluates education systems worldwide by testing the skills and knowledge of a sampling of 15-year-old students, computational thinking appears within the mathematics assessment framework. Furthermore, little attention is paid to coding as a way of developing computational thinking. But this is beginning to change and computer science is now more frequently becoming a core subject in schools. And this will be accelerated as the new PISA assessment for 2025 will include a new test called "Learning in the Digital World" [1], which will evaluate "students' capacity to engage in an iterative process of knowledge building and problem solving using computational tools". It seems that this test will include questions that should be solved using a visual block-based programming language (Scratch-like) (Fig. 1). Schools will therefore need to improve the coding teaching process.

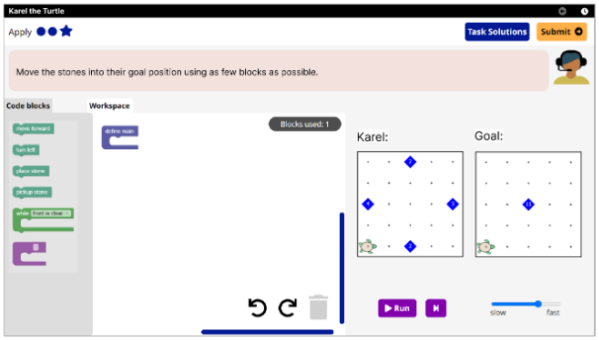

Fig. 1. Complex programming task in the Karel the Turtle unit. PISA 2025 Learning in the world framework.

On the other hand, it has been shown that videogames can be an effective learning and motivational approach for engaging students in difficult STEM subjects (e.g. mathematics). This is in line with the literature that reflects that gamification techniques and the use of games, Game-Based Learning (GBL), are becoming more common as an educational methodology. But videogames are not only for learning. Videogames can also be used to address other issues such as raising students' awareness or change attitudes towards complex social problems (e.g., cyberbullying, gender discrimination). In addition, according to videogameseurope.eu, some experiments suggest that girls who play video games choose 3 times more STEM disciplines than those who do not play (even if this is a topic that need more systematic research to be confirmed).

Therefore, we consider that video games can be a natural learning tool to both learn basic programming concepts and to increase interest in computational thinking in new public (e.g.,

This work has been partially funded by the Ministry of Education (PID2020-119620RB-I00), by the European Commission (Gaming4Coding Erasmus+ project 2021-1-SE01-KA220-SCH-000023932) and by the Telefónica-Complutense Chair on Digital Education and Serious Games.

[1] https://www.oecd.org/pisa/innovation/learning-digital-world/

girls). We want to explore a more natural way of presenting programming not so much as something compulsory to learn but as something beneficial to the player. Programming as a kind of superpower that will "power-up" your avatar making it easier for you to win in a game (i.e. in the same line that in Minecraft programming can be a way to automatize and speed up building construction avoiding doing repetitive tasks such as laying bricks one by one). Through playing with the avatars students will better understand the computational concepts through active engagement, all while learning at their own pace. But we want to do it in a way that helps the teacher in teaching programming without requiring a great deal of added effort or complexity. That means that the game should be easy to deploy in a classroom. The game should be motivational but simple enough so that learning to play does not take too much time. And finally, the videogame should be flexible and powerful enough to introduce all the required programming concepts effectively so they can be transferred to a visual language (e.g. Scratch) or a text language (e.g., Python).

The Gaming4Codign Erasmus project aims to highlight the need to find effective and engaging ways to better teach programming at compulsory education stages and the usefulness of video games as an educational tool for this purpose. It also proposes the design of a video game to learn basic programming concepts and, also, to improve computational thinking due to the close relationship between both. We consider that a programming game is an effective way to involve more students into computational thinking in a very concrete way using some interactive and appealing scenarios.

II. Programming vs Computational Thinking

The concept of Computational Thinking (CT) [4] often appears in studies dealing with the teaching of programming. But this also happens in the opposite direction. We also see this when looking for educational games focused on learning programming. It is therefore necessary to have a better understanding of the concept of computational thinking and how it relates to programming.

There are several definitions and frameworks for CT [2]. An early definition, which also popularized the concept, stated that the characteristics of CT were that it was about thinking like a computer scientist, learning fundamental skills, solving problems in a human way, drawing on mathematics and engineering, [3] highlighting ideas rather than artifacts and that it is for everyone everywhere [4]. Later definitions often view CT as a collection of skills or aspects. Shute, Sun and Asbell-Clarke [2] divide CT key concepts in the facets of decomposition, abstraction, algorithms, debugging, iteration, and generalization. While Brennan and Resnick [5] understand CT as encompassing three key dimensions, namely computational concepts, computational practices, and computational perspectives. Grover and Pea [6] have stated that research on CT often focuses on the problem of definition when it comes to CT, and which tools that could be said to foster development in CT. In many countries CT has become an integral part of kindergarten to grade 12 (K-12) education and curriculum through the ongoing integration of programming in schools [7].

Regarding the relationship between programming and CT there are both similarities and differences between the two. Computer programming is the activity “to write a computer program” [8], and several practices associated with programming can be found in the definitions and frameworks of CT. Practices such as algorithms and debugging [3] and concepts such as sequences, loops, parallelism, and conditionals [5]are part of definitions and frameworks for CT but also closely related to computer programming. However, CT is often described in more general terms, including skills and practices that go beyond computer programming. Going back to the definition by Wing [4], it is stated that CT is not about programming but rather about conceptualizing. This is motivated by that “thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction”. A similar relationship between programming and CT can be viewed in later definitions and frameworks for CT, such as generalization in the framework by Shute, Sun and Asbell-Clarke [3] and the key dimension of computational perspectives by Brennan and Resnick [5].

III. Digital Games for Programming

GBL concept appeared at the university level in the 1970s introduced by pedagogues such as Piaget [9] and Vygotsky[10]. Jean Piaget was also the pedagogue who presented the GBL ideas of Comenius for a new target group [11]. Later the idea of using games in educational contexts got a renaissance when Mark Lepper [12] and Thomas Malone [13] first separately presented their analyses of how computer games could stimulate intrinsic motivation. Their findings on digital games and intrinsic motivation were later merged in the creation of the taxonomy of intrinsic motivation [14]. One of several links between Comenius’s ideas and Lepper and the taxonomy of intrinsic motivation is the American pedagogue and philosopher John Dewey. Both Comenius and Dewey criticized the frequent occurrence of root learning and suggested the idea of learning through activities outside the traditional classroom activities [15]. In the 21st century, there has been a rapid development of digital games with a wide variety of game genres. With the phenomenon of digital 'casual games', playing has reached new target groups in all age groups [16]. Today, when humans play more than ever, it would be desirable to increase the use of games in teaching and learning activities.

The use of video games as an educational resource in the field of programming provides a practical and motivating experience for students. These games allow students to learn to program in a playful and engaging way, eliminating some of the initial fear that some beginners may feel towards programming. By interacting with games, students can experience visible and rewarding results from their work, increasing their engagement and interest for learning.

Fig. 2. Rabbids Coding. A Ubisoft game to learn programming.

The importance of programming and computational thinking in society has led many companies to develop video games to address a current educational need. Big companies such as Google and Ubisoft have developed and explored video games that teach basic programming concepts with BlocklyGames[2] and Rabbids Coding[3] (Fig. 2) respectively. Other game developers are taking advantage of the need to create platforms that work as a service to particular people and schools. This is the case for CodeCombat[4] and Programming Hero. On[5] the other hand, researchers study the best ways and mechanics with which video games can teach programming and how effective are compared with more traditional approaches [17]. They also explore how players themselves learn, the differences between novice and expert programmers, and what difficulties players encounter during their learning [18].

The different video games that can be accessed can be classified into those that are free like ToolboX academy[6] (Fig. 3) and those that require some kind of subscription or payment to access like CodeSpark[7]. On the other hand, we have those that focus on text-based programming, sometimes with real language syntax and others with pseudocode (CodeCombat), and others that use visual programming (Lightbot[8]). However, not all the games created over the last few years are still available. Many of them stops receiving maintenance, disappear from the different download platforms like Grasshopper and May's Journey, or are simply developed for research purposes and are never published [19]. Usually, those games are not distributed as open code therefore it is not possible to update them or to modify them to specific purposes or contexts (e.g. localization).

Other aspects to consider are the programming concepts involved in the gameplay and their sequencing in the game. What are the key concepts required to learn programming and how and in which order they should be introduced to simplify its learning is still an open question. According to some studies, variable initialization is more difficult to understand than variable updating or testing. Novice programmers also have problems with loops and conditionals, as well as actions performed within loops or conditional statements. Students also often have misconceptions about the operation of recursive functions [19]. Most developed video games to teach programming are focused on fundamental programming concepts such as variables, syntax, data types, output and input, conditions, loops, and functions. And in some cases, they even address recursion. However, not all games introduce the same concepts or in the same order. Other concepts that are also covered for some games include simple data structures such as arrays and lists, program debugging, and code documentation [17]. However, there are very few games that address learning through multiplayer features, encouraging player cooperation or competition between players, their advantages over single-player games are currently being explored [20].

Finally, it should be noted that all these games are intended not only to teach the basics of learning to program but also encourage the development of deeper computational thinking. In the end, even if they are big controversy about what is more important and are usually taught in a very different way, we consider that both aspects are closely related and could be integrated to get a more comprehensive understanding of computer science.

Among all games and platforms currently available, we can use games and GBL in more ways than most other subjects due to the opportunity to make game programming a part of teaching and learning activities. There are at least four GBL approaches that can be used in programming education: 1) The use of commercial games to learn programming [17], 2) The use of teacher or student-developed educational games for programming [21], 3) Learning to program by building games [22], and 4) Teaching and learning activities in interactive game environments [23].

Firstly, existing commercial-off-the-shelf (COTS) games can be used on several difficulty levels to learn programming or computational thinking. On the easier levels for the younger target audience games such as Lightbot (Fig. 4). This easy-to-learn game that teaches the very fundamental concepts has been widely used both in primary school [24], and in introductory courses at the university level [25]. The Lightbot game is in the spirit of Seymour Papert's classic Turtle Graphics and introduces computational thinking and sequential instructions by moving objects around. On the more advanced side of COTS games for programming, there are games such as the Human Resource Machine[9], TIS-100[10],

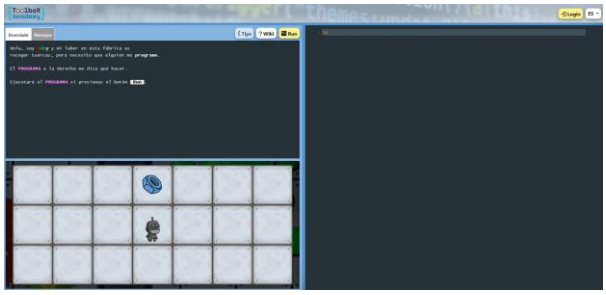

Fig. 3. Toolbox academy. Platform to teach programming.

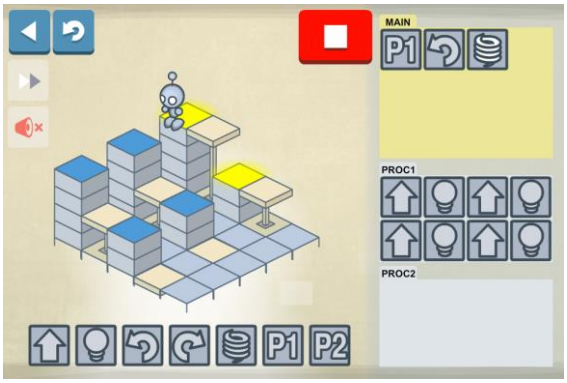

Fig. 4. Lightbot. Puzzle game based on coding that introduces programming concepts.

[2] https://blockly.games/
[3] https://store.ubisoft.com/es/rabbids-coding/5d96f9b05cdf9a2eacdf68cb.html
[4] https://codecombat.com/
[5] https://www.programming-hero.com/
[6] https://toolbox.academy/
[7] https://codespark.com/play/
[8] https://lightbot.com/
[9] https://tomorrowcorporation.com/humanresourcemachine
[10] https://www.zachtronics.com/tis-100/

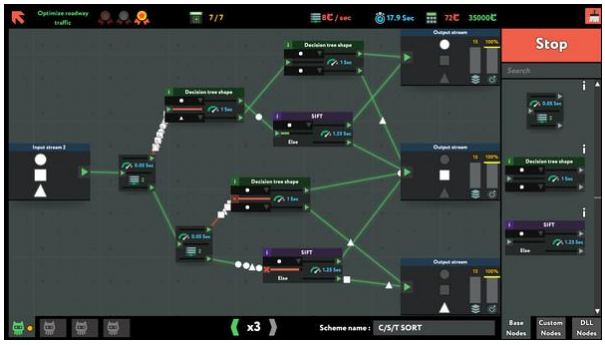

Fig. 5. while True: learn():. Puzzle game that introduces deep learning concepts.

and Shenzhen[11]. Three games that try to teach something as complex as low-level assembly programming in an enjoyable manner [26]. Some of these games not only cover programming concepts, but also with much more advanced topics related to computation. Concepts such as artificial intelligence, automata, cybersecurity and machine learning also appear. An example of this is the game while True: learn()[12] (Fig.4).

Secondly, teachers with programming skills can develop their own learning games that are tailored to the actual curricula. A university-level branch of this concept is to have assignments for students who have good programming skills, where they build learning games for training in fundamental programming. At the same time as more experienced students improve their programming skills by building games, the best-built games could be used in introductory courses [21].

Thirdly, learning to program by building games has been used for many years in computer science. Like the second approach, this can be applied in a wide range from building games by the use of block programming environments [27], to advanced low-level programming [28].

Finally, the fourth approach, which resembles the third, is to use various interactive environments to practise computational thinking and programming. An example of such a digital interactive environment is Minecraft. An environment that has been described as "Virtual Lego" [29], where the digital building blocks can be used to stimulate interest in programming [23], or for learning to program by building games [30].

In addition to these approaches, programming learning games are also starting to appear in *metaverse* environments (Minecraft environment could also be considered as a metaverse) [31]. For example, a CodeCombat Wolds [13]game has just been released within the Roblox gaming environment (which at least in some respect can be considered as one of the game metaverses with potential educational uses).

IV. CodeQuest

Gaming4Coding has led to the creation of a simple open-source game concept that has been iteratively expanded. This game aims to teach programming in a more natural and engaging way. The game focus on the age group of 10–16 year-old students and the aesthetics is designed to appeal to a broad spectrum of players (e.g. girls).

The core idea of this game called CodeQuest is that players play by collecting avatars that are cute fantasy creatures called 'critters' which can be trained by giving them instruction scripts. The training is purposefully not called programming to give the game less of a technical feeling. The main gameplay consists of collecting interesting critters, training them for racing and entering them into a race against other players. Races are held on different obstacle courses. The critters compete either against bots or against other players.

While there are games for learning programming characterized by puzzle mechanics and RPGs, there are none focused on training and collecting creatures. This game genre characterized by training and collecting creatures is called Monster-taming, one of its greatest exponents being the Pokémon saga. Another underexplored feature of games for developing computational thinking and learning programming is multiplayer. CodeQuest implements this option, enabling players to play against each other, and researchers to study the effects of competition during learning in the field of programming.

The design of the game therefore focuses on the following 4 pillars:

- Collecting: The activity of collecting things has a large appeal as a game element, especially when the player can collect creatures. The most prominent example of this is the Pokémon game. Since new creatures are awarded for winning races or for other achievements the interest in collecting critters has a strong impact on the motivation of the players to get better at coding, so that collecting new critters becomes easier.
- Rewards: Rewards are very important for player motivation. The game rewards victories in races so that players get something out of having created well working training scripts for their critters. However, the game also rewards personal achievements, even if the player doesn't end up winning the race. Thus, rewards play an important role in keeping the motivation high for any learning and gaining of experience. The biggest rewards are new collectable critters, smaller rewards are special items for the critters and trophies.

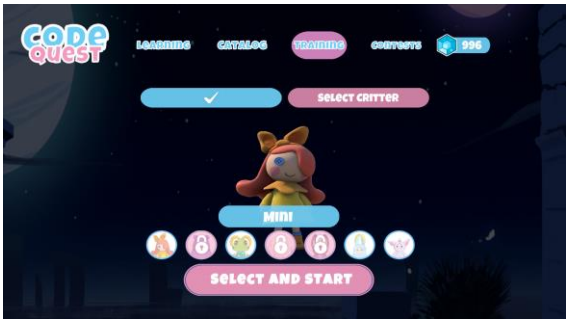

Fig. 6. Avatar selection.

[11] https://www.zachtronics.com/shenzhen-io/
[12] https://store.steampowered.com/app/619150/while_True_learn/
[13] https://www.roblox.com/games/11704713454/Pets-CodeCombat-Worlds

- Competition: The competition element is a well-established way to challenge a player to become better. The game uses a mixture of players and bots as competitors so that the player always has the opportunity to win against someone, even if it is too hard to win first place. Competition should also give players an opportunity to learn from others.
- Matching Difficulty: The game uses a smart algorithm to assign competence levels to players in order to match them with other players or bots that are performing at comparable levels. This avoids having a beginner crushed by a very advanced player.

Being able to choose between different creatures to play with, compete with other players, complete challenges and discover new blocks of code to progress with are intended to be the motivating engine that keeps players interested and thus motivates them to keep learning and improving (Fig. 6).

In the actual game the players first learn how to give critters commands to guide them through an obstacle course. Commands correspond to functions in Python but the players will be using predefined functions from the beginning (such as "Run", "Move Left", "Move Right", "Jump Forward", etc.) before learning how to create their own functions (Fig. 7). The game starts with tutorials which challenge the player to solve a situation, so they are essentially little puzzles that help the player acquire the skills needed. The tutorials gradually introduce the player to the competitive racing game in which more critters can be unlocked and rewards can be won. Of course, the tutorials can be skipped and the players can immediately start training their critters and enter them into races.

In a race the player only has limited control over the critters as they are mainly in the role of a spectator. Winning or losing a race has a lot to do with which critter is selected for the racetrack and which commands are given to the critter in the 'training script'. This is where it is most important to get new creatures and train them to fit the different circuits.

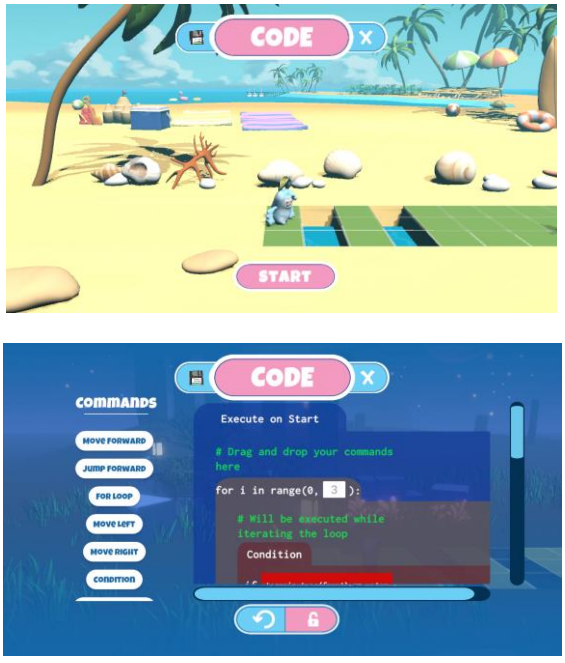

Fig. 7. Circuit and code screens in CodeQuest.

When training a critter, a player writes down a sequence of commands which the critter will execute in the race. This could start in a very simple way ("Run", "Run", "Run", "Jump Forward") and then move to more advanced coding structures with while loops and conditionals. The game never pushes the player to advance in coding skills, however the competitive aspect and the rewards for winning races are a huge motivation to improve the performance of the critters, thus leading to players trying out more advanced code.

Another feature is that the view where the code is implemented mixes both a more visual part, of predefined code blocks, with a textual part that follows the Python syntax (Fig. 7). Python syntax was chosen for the coding elements since Python is one of the most popular programming languages which is also widely recommended for school environments in Europe. It is also easy to learn. In the actual game the players first learn how to give critters commands to guide them through an obstacle course. Commands correspond to functions in Python but the players will be using predefined functions from the beginning ("Run", "Dodge Left", "Move Right", "Condition", etc.) before learning how to create their own functions.

Regarding programming concepts, the tutorial (Fig. 8) introduces some of them as in other games and the different levels are categorized by the concept being taught or practiced. the tutorial deals first with the different commands that allow the player to move, then loops, conditionals and finally variables. There is also a last advanced category with nested loops and switch cases. But the tutorial es optional, so, in CodeQuest the concepts are intrinsic, and the player develops computational thinking and learns programming concepts without being aware of it. The game seeks that the player learns in an unconscious way, practicing the concepts through finding new ways to improve, training his creatures, and completing the challenges presented to him through competition with other players.

The game also incorporates an analytics system that captures user interaction information. For the capture of this information a standard language is used, specifically xAPI (eXperience API). For this purpose, the game incorporates a tracker that supports xAPI and that takes care of communicating the relevant interaction data to a data server (which in xAPI terminology is called a Learning Record Store). This allows to have all the information of the users while they play and makes possible to make different types of analysis (e.g. average playing time, progress in the game, difficulty of the levels based on the playing time or on the number of failed attempts). This will allow to better evaluate

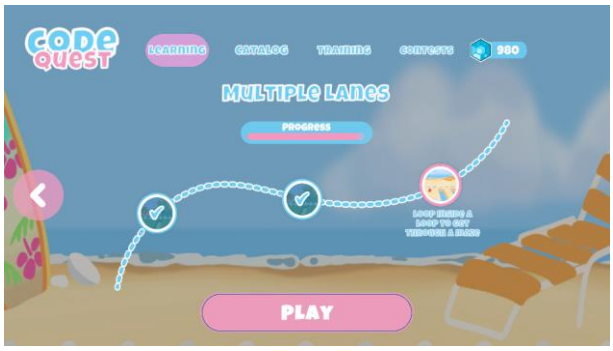

Fig. 8. Tutorial screen in CodeQuest.

the game as it is deployed in real classroom environments. To guarantee all aspects of privacy and to comply with the European GDPR regulation, all data captured for the scientific validation of the game will be pseudo-anonymized at the origin using a unique code provided by the teachers for each of the players.

V. Discussion

A digital transformation of our society is necessary, and to this end, over the last few years, computational thinking and programming have been introduced in the compulsory stages of education. This makes it necessary to create new tools adapted to these stages to enable learning in these two subjects. Video games can be a useful tool for them, either as a classroom tool to be used with game-based learning methodologies or for students themselves to learn at home and find interest in STEM careers. For instance, software developers such as Microsoft saw the potential of video games in education, buying Minecraft and releasing an educational version.

As we have seen, some studies speak of games as an activity that is intrinsically motivating, due to aspects such as challenges, progression, learning new things and being able to make decisions, all of them characteristics that attract players to play. However, many gamers also report that they play games to relax, destress or share time with friends, indicating that video games are becoming more and more a part of society, like books or films, and there is also an extrinsic motivation to play for some players. But not all of society plays video games [32]. Even though more and more people play games there are also very different profiles of players. For example, Bartle, proposes 4 profiles [33]. This makes it necessary to study different game genres and mechanics, as well as ways of approaching programming and computational thinking. To study the effectiveness of video games, their acceptance among different student profiles and to compare them. In this case, the proposed game has as its mechanics the collection and training of creatures, the critters. The game falls into the monster-tamer genre, a genre different from the more typical one used to teach programming, which is the puzzle genre.

Since not all players are the same and not all people play games, it is necessary to research and create new video games, different from each other, that address the teaching of programming and/or computational thinking to help this digital transformation of society. One of the features that has been little explored in this area is that of multiplayer games, and more specifically, competitive gaming. Adding this option, allowing students to challenge each other and compete, can help to retain players with a more "killer" profile, encouraging them to improve and continue learning new concepts and techniques related to programming that will help them to beat their opponents. Therefore, the game presented in this paper introduces the option for players to challenge each other with the scores obtained so far. This allows players to learn from the strategies used by the players they face.

On the other hand, maintaining challenges and level completion, as well as collectibles, allows reaching out to more "explorer" and "archiever" players. Finally, we must not forget those players who are more “social”, adding mechanics for sharing achievements and collaboration can help to maintain the interest of this type of player. But balancing the mechanics to maintain the interest of all these profiles is a very hard task. CodeQuest's option to complete quests and collect new creatures is intended to appeal to this type of player. In addition, while there is the option to face other players, this is optional, and students can use the game as a conventional single player.

One of the limitations of this game is that its deployment is intended for mobile devices (e.g. IOS and Android). As practically all the students have a smartphone, the initial idea was to use a BYOD (Bring Your Own Devices) approach where students provide their own device as it simplifies the deployment and reduces the requirement for schools. Therefore, its use in the classroom may be affected by the possible regulations of the center where it is to be used (for instance in Spain mobile phone use in schools is being restricted in some regions). However, this feature, on the other hand, makes it easier for players to practice and learn outside the school, seeing the activity from a more leisure point of view.

Moreover, while it is common to talk about computational thinking and programming together, it should be noted that they are two different but related things. So, it would be possible to develop games that develop computational thinking without having to deal with programming. However, it is more complex to develop programming knowledge without fostering computational thinking because these skills are necessary for programming and problem solving.

Finally, all the ethical and privacy issues involved in both game research and its application in the classroom must be taken into account. For example, this would make it very complex to use some of the new *metaverse* game proposals as there may not be sufficient clarity on the privacy and ownership aspects of student data acquired during the game. European privacy and data protection regulations GDPR must be complied with at all times and users and schools must be informed and sign consent for both data collection and data processing and the purpose of the intended data analysis.

VI. Conclusions and Future Work

As previously stated Gaming4Codign Erasmus project aims to highlight the need to find effective and engaging ways to better teach programming at compulsory education stages and the usefulness of video games as an educational tool for this purpose. This paper addresses the need to create new games focused on players to learn programming and develop computational thinking, highlighting why video games can be an effective educational tool for it. Therefore, the design of a video game called CodeQuest is also presented, using a game genre that has not been frequently used before to address the teaching of programming.

Video games stimulate active learning, are motivating and present information in a way that is attractive to learners. This can have a positive impact on motivation to learn. While we already find numerous educational games in this area of programming, studies indicate a lack of research on the use of multiplayer and how it affects learning. CodeQuest introduces multiplayer features that will allow researchers to study in the future how competition affects the motivation and learning of programming. This multiplayer feature will allow us to evaluate whether competitive mechanics can be an interesting feature when teaching programming, compared to single-player games, thus differentiating itself from other proposals such as Articoding (Fig. 9).

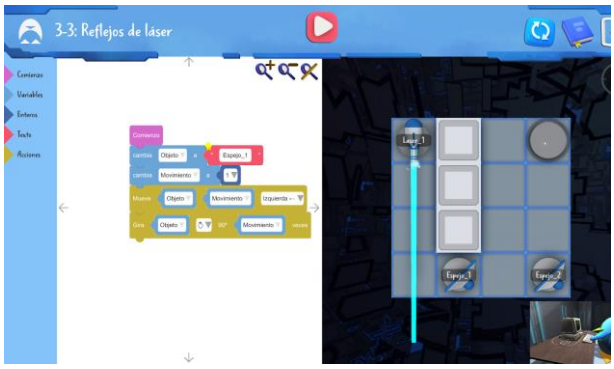

Fig. 9. Articoding, single-player game to teach programming basic concepts.

With the inclusion of computational thinking and coding aspects in the new PISA school evaluation the interest in those kinds of innovative ways to effectively include programming in schools will be more important. As previously stated, the new PISA evaluation in 2025 will add computational thinking and block coding to the list of competencies it evaluates, the need for schools to have tools to train these skills in an effective way increases.

Finally, to avoid the obsolescence of the game and to increase its impact, we plan to distribute the game as open code so other researchers can learn from this experience, reuse it and improve or adapt to their requirements. Next steps in the Gaming4Coding project are to run a case study with CodeQuest in the coming months to explore its effectiveness as an educational tool as well as the effects of competition between players on learning.

REFERENCES

[1] S. Bocconi *et al.*, "Reviwing computational thinking in compulsory education. State of play and practices from computing education.," Luxembourg, 2022. doi: https://data.europa.eu/doi/10.2760/126955.

[2] N. Humble and P. Mozelius, "Grades 7–12 teachers' perception of computational thinking for mathematics and technology," *Front Educ (Lausanne)*, vol. 8, Mar. 2023, doi: 10.3389/feduc.2023.956618.

[3] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22. Elsevier Ltd, pp. 142–158, Nov. 01, 2017. doi: 10.1016/j.edurev.2017.09.003.

[4] J. M. Wing, "Computational thinking," *Commun ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006, doi: 10.1145/1118178.1118215.

[5] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," *Proceedings of the 2012 annual meeting of the American educational research association (AERA)*, 2012, Accessed: Sep. 12, 2023. [Online]. Available: https://www.media.mit.edu/publications/new-frameworks-for-studying-and-assessing-the-development-of-computational-thinking/

[6] S. Grover and R. Pea, "Computational Thinking in K–12," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, Jan. 2013, doi: 10.3102/0013189X12463051.

[7] N. Humble and P. Mozelius, "Teacher perception of obstacles and opportunities in the integration of programming in K-12 settings," Jul. 2019, pp. 350–356. doi: 10.21125/edulearn.2019.0125.

[8] IEEE Computer Society. Standards Coordinating Committee., *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries, 610*. Institute of Electrical and Electronics Engineers, 1991. doi: https://doi.org/10.1109/IEEESTD.1991.106963.

[9] Jean Piaget, *To understand is to invent: the future of education*. New York: Penguin Books, 1973.

[10] L. S. VYGOTSKY, *Mind in Society*. Harvard University Press, 1980. doi: 10.2307/j.ctvjf9vz4.

[11] J. Piaget, *Play, Dreams And Imitation In Childhood*. Routledge, 2013. doi: 10.4324/9781315009698.

[12] M. R. Lepper and D. Greene, "Turning play into work: Effects of adult surveillance and extrinsic rewards on children's intrinsic motivation.," *J Pers Soc Psychol*, vol. 31, no. 3, pp. 479–486, Mar. 1975, doi: 10.1037/h0076484.

[13] T. W. Malone, "Toward a Theory of Intrinsically Motivating Instruction*," *Cogn Sci*, vol. 5, no. 4, pp. 333–369, Oct. 1981, doi: 10.1207/s15516709cog0504_2.

[14] Thomas Malone and Mark Leeper, "Making Learning Fun: A Taxonomy Of Intrinsic Motivations For Learning, Aptitude, learning and instruction," 1987, pp. 223–253.

[15] V. Zoric, "The importance of the impact on the formation of pedagogical ideas of the greatest minds in pedagogy: The origin and development of John Dewey's pragmatistic pedagogy Educational policy in socialist Montenegro View project Teacher Education in Europe-History, Structure and Reform View project," 2019. [Online]. Available: https://www.researchgate.net/publication/336288172

[16] J. Juul, *A casual revolution : reinventing video games and their players*, MIT press. MIT Press, 2010.

[17] M. A. Miljanovic and J. S. Bradbury, "A Review of Serious Games for Programming," vol. 11243, S. Göbel, A. Garcia-Agundez, T. Tregel, M. Ma, J. Baalsrud Hauge, M. Oliveira, T. Marsh, and P. Caserman, Eds., in Lecture Notes in Computer Science, vol. 11243. , Cham: Springer International Publishing, 2018, pp. 204–216. doi: 10.1007/978-3-030-02762-9_21.

[18] C. Delozier and J. Shey, "Using Visual Programming Games to Study Novice Programmers," *International Journal of Serious Games*, vol. 10, no. 2, pp. 115–136, Jun. 2023, doi: 10.17083/ijsg.v10i2.577.

[19] J. Díaz, J. A. López, S. Sepúlveda, G. M. Ramírez Villegas, D. Ahumada, and F. Moreira, "Evaluating Aspects of Usability in Video Game-Based Programming Learning Platforms," *Procedia*

Comput Sci, vol. 181, pp. 247–254, 2021, doi: 10.1016/j.procs.2021.01.141.

[20] A. Wynn, J. Wang, R. Han, and T.-C. Hsu, "Multiplayer Serious Games Supporting Programming Learning," *European Conference on Games Based Learning*, vol. 17, no. 1, pp. 721–729, Sep. 2023, doi: 10.34190/ecgbl.17.1.1621.

[21] M. Olsson and P. Mozelius, "Learning to Program by Playing Learning Games Active Learning Classrooms View project Learning to Program by Playing Learning Games View project Learning to Program by Playing Learning Games," 2017. [Online]. Available: https://www.researchgate.net/publication/320271391

[22] A. Batista *et al.*, "A Framework for Games-Based Construction Learning: A Text-Based Programming Languages Approach 3D Me App View project Designing and Validating a Methodology for Shared Decision-making Between Patients and Healthcare Professionals View project A Framework for Games-Based Construction Learning: A Text-Based Programming Languages Approach," 2016. [Online]. Available: https://www.researchgate.net/publication/308777864

[23] C. Zorn, C. Wingrave, E. Charbonneau, and J. J. Laviola, "Exploring Minecraft as a Conduit for Increasing Interest in Programming," 2013.

[24] J. Erazo-Palacios, C. R. Jaimez-González, and B. García-Mendoza, "Towards a Web Generator of Programming Games for Primary School Children," *International Journal of Engineering Pedagogy (iJEP)*, vol. 12, no. 4, pp. 98–114, Jul. 2022, doi: 10.3991/ijep.v12i4.17335.

[25] M. A. López, E. V. Duarte, A. P. Valderrama, and E. C. Gutierrez, "Teaching abstraction, function and reuse in the first class of CS1 - A lightbot experience," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Association for Computing Machinery, Jul. 2016, pp. 256–257. doi: 10.1145/2899415.2925505.

[26] S. Cass, "Some assembly (language) required - Three games that make low-level coding fun [Resources_Geek Life]," *IEEE Spectr*, vol. 54, no. 5, pp. 19–20, May 2017, doi: 10.1109/MSPEC.2017.7906890.

[27] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine, "Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment," *Procedia Soc Behav Sci*, vol. 191, pp. 1479–1482, Jun. 2015, doi: 10.1016/j.sbspro.2015.04.224.

[28] J. Kawash and R. Collier, "Using video game development to engage undergraduate students of assembly language programming," in *Proceedings of the 14th annual ACM SIGITE conference on Information technology education*, New York, NY, USA: ACM, Oct. 2013, pp. 71–76. doi: 10.1145/2512276.2512281.

[29] A. Overby and B. L. Jones, "Virtual LEGOs: Incorporating Minecraft into the Art Education Curriculum," *Art Education, v68 n1*, pp. 21–27, 2015.

[30] A. Bile, "Development of intellectual and scientific abilities through game-programming in Minecraft," *Educ Inf Technol (Dordr)*, vol. 27, no. 5, pp. 7241–7256, Jun. 2022, doi: 10.1007/s10639-022-10894-z.

[31] J. Han, G. Liu, and Y. Gao, "Learners in the Metaverse: A Systematic Review on the Use of Roblox in Learning," *Educ Sci (Basel)*, vol. 13, no. 3, p. 296, Mar. 2023, doi: 10.3390/educsci13030296.

[32] G. Reid, "Motivation in video games: a literature review," *The Computer Games Journal*, vol. 1, no. 2, pp. 70–81, Nov. 2012, doi: 10.1007/BF03395967.

[33] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," 1996.